Xilinx is disclosing this Document and Intellectual Property (hereinafter “the Design”) to you for use in the development of designs to operate on, or interface with Xilinx FPGAs. Except as stated herein, none of the Design may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx. Any unauthorized use of the Design may violate copyright laws, trademark laws, the laws of privacy and publicity, and communications regulations and statutes.

Xilinx does not assume any liability arising out of the application or use of the Design; nor does Xilinx convey any license under its patents, copyrights, or any rights of others. You are responsible for obtaining any rights you may require for your use or implementation of the Design. Xilinx reserves the right to make changes, at any time, to the Design as deemed desirable in the sole discretion of Xilinx. Xilinx assumes no obligation to correct any errors contained herein or to advise you of any correction if such be made. Xilinx will not assume any liability for the accuracy or correctness of any engineering or technical support or assistance provided to you in connection with the Design.

THE DESIGN IS PROVIDED “AS IS” WITH ALL FAULTS, AND THE ENTIRE RISK AS TO ITS FUNCTION AND IMPLEMENTATION IS WITH YOU. YOU ACKNOWLEDGE AND AGREE THAT YOU HAVE NOT RELIED ON ANY ORAL OR WRITTEN INFORMATION OR ADVICE, WHETHER GIVEN BY XILINX, OR ITS AGENTS OR EMPLOYEES. XILINX MAKES NO OTHER WARRANTIES, WHETHER EXPRESS, IMPLIED, OR STATUTORY, REGARDING THE DESIGN, INCLUDING ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT OF THIRD-PARTY RIGHTS.

IN NO EVENT WILL XILINX BE LIABLE FOR ANY CONSEQUENTIAL, INDIRECT, EXEMPLARY, SPECIAL, OR INCIDENTAL DAMAGES, INCLUDING ANY LOST DATA AND LOST PROFITS, ARISING FROM OR RELATING TO YOUR USE OF THE DESIGN, EVEN IF YOU HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. THE TOTAL CUMULATIVE LIABILITY OF XILINX IN CONNECTION WITH YOUR USE OF THE DESIGN, WHETHER IN CONTRACT OR TORT OR OTHERWISE, WILL IN NO EVENT EXCEED THE AMOUNT OF FEES PAID BY YOU TO XILINX HEREUNDER FOR USE OF THE DESIGN. YOU ACKNOWLEDGE THAT THE FEES, IF ANY, REFLECT THE ALLOCATION OF RISK SET FORTH IN THIS AGREEMENT AND THAT XILINX WOULD NOT MAKE AVAILABLE THE DESIGN TO YOU WITHOUT THESE LIMITATIONS OF LIABILITY.

The Design is not designed or intended for use in the development of on-line control equipment in hazardous environments requiring fail-safe controls, such as in the operation of nuclear facilities, aircraft navigation or communications systems, air traffic control, life support, or weapons systems (“High-Risk Applications”). Xilinx specifically disclaims any express or implied warranties of fitness for such High-Risk Applications. You represent that use of the Design in such High-Risk Applications is fully at your risk.

©2002-2005 Xilinx, Inc. All rights reserved. XILINX, the Xilinx logo, and other designated brands included herein are trademarks of Xilinx, Inc. PowerPC is a trademark of IBM, Inc. All other trademarks are the property of their respective owners.
**Virtex-II Pro and Virtex-II Pro X FPGA User Guide**

**UG012 (v4.0) 23 March 2005**

The following table shows the revision history for this document. All revision publications are electronic only (PDF) unless otherwise noted.

<table>
<thead>
<tr>
<th>Version</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>01/31/02</td>
<td>1.0 (Print Edition.) Initial Xilinx release.</td>
</tr>
<tr>
<td>10/14/02</td>
<td>2.0 (Print Edition.) Updated and reprinted. New Virtex-II Pro family members and packages. Revised I/O support information.</td>
</tr>
<tr>
<td>12/04/02</td>
<td>2.1 Added full support for 3.3V I/O standards PCI-X, LVTTL, and LVCMOS33.</td>
</tr>
</tbody>
</table>
| 01/23/03   | 2.2 • Added wire-bond package FG676 for XC2VP20, 30, and 40. New package diagram (Figure 5-66) and pinout diagrams (Figure 5-11 through Figure 5-15). Revised Table 5-6.  
• Removed pinout diagrams and other references for XC2VP40FF1517.  
• Revised material in, and added Figure 4-4 to, section “Mixed Voltage Environments,” page 296.  
• Clarified explanation of banks 4/5 VCCO settings for configuration vs. operation in section “Special VCCO Requirements during Configuration and Readback,” page 297. |
| 04/11/03   | 2.3 • Table 3-37, Table 3-38: Corrected package type FG672 to FF672, and added package type FG676.  
• Deleted (former) Table 3-3, Power-Up Timing Characteristics, and replaced it with a hyperlink to Data Sheet Module 3.  
• Deleted (former) Table 1-5, Multiplier Switching Characteristics, and replaced it with a hyperlink to Data Sheet Module 3.  
• Changed attribute CLOCK_FEEDBACK to CLK_FEEDBACK.  
• Clarifying text added to section “Phase Shift Enable - PSEN” in Chapter 3.  
• Additional implementation rule added to section “DCI in Virtex-II Pro / Virtex-II Pro X Hardware” in Chapter 3.  
• Corrections made in command line, key file, and key file command line equivalent in section “Creating Keys” in Chapter 3.  
• Added device-specific parameters to step 15 of “Single Device Configuration Sequence” in Chapter 4.  
• Section “Master SelectMAP Data Loading” in Chapter 4: Corrected maximum frequency for SelectMAP configuration without BUSY handshaking to 5 MHz.  
• Table 4-8: Corrected maximum SelectMAP frequency (FCC_SELECTMAP) to 50 MHz.  
• Appendix A, “BitGen and PROMGen Switches and Options”: Updated all command-line options to correlate with development tools and their documentation.  
• Table 3-58: Added numerous LVDS primitives to this table, including new differential termination primitives.  
• Added new section “LVDS Input HDL Examples.”  
• Changed Figure 3-118 to show internal rather than external differential input termination.  
• Various minor edits. |
<table>
<thead>
<tr>
<th>Version</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>06/30/03</td>
<td>2.4</td>
</tr>
<tr>
<td>02/02/04</td>
<td>2.5</td>
</tr>
</tbody>
</table>

- Corrected Location Constraints syntax, multiple instances.
- Added Figure 3-23, page 91 and associated explanatory text.
- Added HDL code for resetting the DCM after configuration in section “External Feedback,” page 93.
- Modified Banking Rules in section “DCI in Virtex-II Pro / Virtex-II Pro X Hardware,” page 235.
- Added reference to Answer Record 13012 in section “SSTL2_I_DCI, SSTL2_II_DCI,” page 236.
- Corrected command-line equivalent statements for bitstream encryption key files in section “Creating Keys,” page 267.
- Added new section “SelectMAP ABORT Sequence and ABORT Recovery,” page 314, to section “Master SelectMAP Programming Mode”.
- Corrected shading property of GND pin in symbol key, all diagrams, section “Pinout Diagrams,” page 395.
- Numerous additional minor edits.

- Figure 3-58, page 157: Added missing connection between second LUT of Slice 3 and first LUT of Slice 2.
- Figure 3-71, page 179: Corrected bit numbers on inputs to adder.
- Section “Routing with BlockRAM,” page 183 added.
- Table 3-36, page 201: Added SSO parameters for HSTL18 I/O standards.
- Figure 3-89, page 214: Correct VREF from 0.75V to 0.9V.
- Section “DCI I/O Buffer Library Components,” page 230, added IBUFG_LVDS components to list.
- Section “Location Constraints,” page 249: Added constraint on locating a DDR register next to an SDR register using a different clock.
- Appendix C, “Choosing the Battery for V_BATT” added, with linked reference in section “V_BATT,” page 269.
- Page 290: added paragraph at end of Introduction to Configuration chapter specifying a fixed voltage tie for mode pins M0-M2, and warning not to toggle these pins during or after configuration.
- Table 4-2, page 290: Reduced configuration bitstream lengths for all devices by 64 bits.
- Section “Configuration Pins,” page 294: Added warning and constraints regarding application of 3.3V signals to dedicated configuration pins.
- Table 4-3, page 295: Corrected directionality of BUSY/DOUT to Output, INIT_B to Input/Output. Updated with data from XAPP659.
- Former Table 4-6, Master/Slave Serial Mode Programming Switching, and former Table 4-8, SelectMAP Write Timing Characteristics, removed. See the Virtex-II Pro / Virtex-II Pro X Data Sheet, DC and Switching Characteristics for these and all other timing parameter specifications.
- Table 4-6, page 302: Corrected FCC_SERIAL from 66 MHz to 50 MHz.
- Added footnote to Figure 4-9, page 303, and Figure 4-11, page 305, clarifying that DOUT transitions on the falling edge of CCLK.
- Second paragraph below Figure 4-13, page 309: Corrected maximum no-handshake SelectMAP configuration speed from 5 MHz to 50 MHz.
- Section “Master SelectMAP Data Loading,” page 309: Changed wording to emphasize that if RDWR_B is toggled while CS_B is still asserted, a configuration abort will occur.
2.5 (cont’d)

- Section “Express-Style Loading,” page 308: Added new text to clarify the sequencing of signals before, during, and after data loading.
- Page 320: Added text to TDO paragraph indicating that pin is open-drain and should be pulled up externally.
- Table 4-13, page 331: Updated “Total No. of Bits” column with new numbers.
- Figure 5-53; Figure 5-54; Figure 5-55; page 456; Figure 5-55, page 457; and Figure 5-56, page 458: Corrected banking assignments for pins AB12 and AB31. Now shows these pins in Banks 2 and 7 respectively.
- Table 5-3, page 479: Changed Theta-JC to 0.5°C/Watt for all FF packages.
- Section “Crosstalk,” page 486: Clarified wording of PCB trace routing guidelines to minimize crosstalk onto VREF lines.

04/19/04 2.6

- Section “Digital Clock Managers (DCMs)” in Chapter 3: Removed all referrals to CLK2X as an option in implementing a clock feedback loop for the DCM, including:
  - Removed former Figures 3-25, 3-27, and 3-39, which showed schematics of CLK2X feedback implementation.
  - Section “Attributes,” page 96: Removed section “Feedback Input” discussing the CLK_FEEDBACK attribute, which was used to designate CLK2X as the feedback clock.
  - Table 3-9, page 113: Removed CLK_FEEDBACK from the table of attributes.
  - Section “VHDL and Verilog Instantiation,” page 114: Removed BUFG_CLK2X_SUBM, BUFG_CLK2X_FB_SUBM, and BUFG_PHASE_CLK2X_SUBM from the list of available DCM submodules.
- Page 178: Added registered multiplier primitive to table Table 3-26, new schematic of both multiplier primitives (Figure 3-70), and explanatory text.
- Table 3-36, page 201: Added LVPECL_25 and BLVDS_25 to table.
- Table 3-38, page 205: Added referral to XAPP689 for managing ground bounce.
- Section “DCI I/O Buffer Library Components,” page 230: Removed all LVDS_33_DCI and LVDSEXT_33_DCI components from listing.
- Section “Creating a Bidirectional LVDS Buffer,” page 261, and “Creating an LVPECL Output Buffer,” page 264: Added referral to Table 3-36 for SSO guidelines.
- Section “Bitstream Encryption,” page 266: Added referral to Appendix C, “Choosing the Battery for VBAT.”
- Section “Creating Keys” (end): Added material discussing use of CBC initial value to safeguard bitstream security.
- Table 4-3, page 295:
  - Mode pins (M0, M1, M2) not to be connected to 3.3V unless through a 100Ω resistor.
  - VBAT pin to be connected to GND or VCCAUX when bitstream encryption not used.
- Section “Frame Length Register (FLR),” page 333: Added “minus one word” to the description of the value to be loaded into this register.
- Former Table 4-3, Power-Up Timing Configuration Signals, removed. See the Virtex-II Pro Platform FPGA Data Sheet, DC and Switching Characteristics for these and all other timing parameter specifications.
- Section “Test Access Port,” page 319: Added mention of implementation tool pull-up, pull-down, and float options on TMS and TDI pins.
- Updated DS026 in Appendix B, “Platform Flash Family PROMs” to v5.0.
- Various edits and deletions in Glossary.
- Edits throughout to synchronize/align User Guide with Data Sheet Table 1.
<table>
<thead>
<tr>
<th>Version</th>
<th>Revision</th>
</tr>
</thead>
<tbody>
<tr>
<td>06/02/04</td>
<td>2.6.1</td>
</tr>
<tr>
<td>-</td>
<td>Table 4-3, page 295: Corrected BUSY/DOUT from “Input” to “Output”. Corrected INIT_B from “Input” to “Input/Output”. Removed redundant entries for CCLK and PROG_B.</td>
</tr>
<tr>
<td>08/05/04</td>
<td>3.0</td>
</tr>
<tr>
<td>-</td>
<td>Revised throughout with updated tables, text, pinout diagrams, and hyperlinks to include two new devices, Virtex-II Pro X XC2VPX20 and XC2VPX70, which feature the RocketIO X 10.3125 Gb/s MGT.</td>
</tr>
<tr>
<td>-</td>
<td>Revised throughout with updated tables, text, and package diagrams to include new Pb-free packaging options FGG256, FGG456, and FGG676.</td>
</tr>
<tr>
<td>-</td>
<td>Section “Frequency Synthesized Clock Output - CLKFX,” page 100: Added text indicating no CLKFX phase alignment when the CLKIN frequency is below 24 Mhz.</td>
</tr>
<tr>
<td>-</td>
<td>Section “Phase Shift Characteristics,” page 104: Added substantial detailed material related to calculating and controlling the desired phase shift.</td>
</tr>
<tr>
<td>-</td>
<td>Table 3-29, page 186: Added clarifying example to Footnote (2) regarding GTL/GTLP VCCO lower limit.</td>
</tr>
<tr>
<td>-</td>
<td>Section “Daisy-Chain Configuration,” page 303: Revised text to clarify daisy-chain options for strings of different Xilinx FPGA devices.</td>
</tr>
<tr>
<td>-</td>
<td>Section “RDWR_B,” page 305: Revised text to clarify the status of the data bus when RDWR_B is High and Low.</td>
</tr>
<tr>
<td>-</td>
<td>Table 4-16, page 329: Revised step 5 to give XC2VP2 its own TDI code, and step 6 to correct the TDI code from 1 to 0. Corrected Footnote (1) to say “data and instruction”.</td>
</tr>
<tr>
<td>03/23/05</td>
<td>4.0</td>
</tr>
<tr>
<td>-</td>
<td>Chapter 4, “Configuration”: This chapter has been expanded and completely updated with new material. Since most of the material is newly written, specific changes in this chapter are not recorded in the Revision History.</td>
</tr>
<tr>
<td>-</td>
<td>Section “BUFG Exclusivity,” page 75: Corrected numbering of exclusive buffer pairs.</td>
</tr>
<tr>
<td>-</td>
<td>Figure 3-23, page 91: Added note on when to reset the DCM.</td>
</tr>
<tr>
<td>-</td>
<td>Section “Fixed-Mode Phase Shifting in Certain Virtex-II Pro Devices,” page 105: Added link to Solution Record 13349.</td>
</tr>
<tr>
<td>-</td>
<td>Table 3-9, page 113: From “Description” for CLKIN_DIVIDE_BY_2, removed text specifying an upper frequency limit for using this attribute to divide the DCM input clock by 2.</td>
</tr>
<tr>
<td>-</td>
<td>Section “Embedded Multipliers,” page 177: Deleted subsection Two Multipliers in a Single Primitive. Deleted references to DUAL_MULTI_* templates.</td>
</tr>
<tr>
<td>-</td>
<td>Section “Output Drive Source Voltage (VCCO) Pins,” page 198: Added text recommending that VCCO be powered for all I/O standards.</td>
</tr>
<tr>
<td>-</td>
<td>Table 3-36, page 201: Corrected number of SSOs for the following standards: GTL_DC, GTP_DC, HSTLII_DC, SSTL18_DC, LVPECL_25, BLVDS_25. Added the following standards: DIFF_HSTL_II, DIFF_HSTL_II_DC, DIFF_HSTL_II_18, DIFF_HSTL_II_18_DC, DIFF_SSTL_II, DIFF_SSTL_II_DC, DIFF_SSTL_II_18, DIFF_SSTL_II_18_DC.</td>
</tr>
<tr>
<td>-</td>
<td>Section “DCI in Virtex-II Pro / Virtex-II Pro X Hardware,” page 235: Added step 8.</td>
</tr>
<tr>
<td>-</td>
<td>Added Figure 3-108, page 240.</td>
</tr>
<tr>
<td>-</td>
<td>Code template “DDR_input.v,” page 251: Removed assign data_out = q1 &amp; q2 just before “endmodule”.</td>
</tr>
<tr>
<td>-</td>
<td>Section “LVDS I/O,” page 256: Replaced Table 3-58, “Available LVDS Primitives” with new Table 3-58, “Available LVDS Standards”.</td>
</tr>
<tr>
<td>-</td>
<td>Section “Loading Encrypted Bitstreams,” page 269: Corrected reference to JTAG JSTART instruction to JTAG JPROG_B.</td>
</tr>
<tr>
<td>Version</td>
<td>4.0 (cont’d)</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>03/07/05 (cont’d)</td>
<td></td>
</tr>
</tbody>
</table>

- Added new section “Temperature-Sensing Diode (DXP/DXN),” page 270.
- Corrected pinout diagram “FF672 Composite Pinout Diagram (XC2VP4)” (Figure 5-19, page 418). Previously, this figure had shown the composite pinout diagram for the FF672/XC2VP7 device.
- Section “Design Examples,” page 519 (Appendix C): Deleted text and illustration that incorrectly suggested different battery voltage compliance requirements for Virtex-II and Virtex-II Pro BBRAM power pin.
# Table of Contents

## Preface: About This Guide

<table>
<thead>
<tr>
<th>Guide Contents</th>
<th>Additional Resources</th>
<th>Conventions</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Typographical</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Online Document</td>
</tr>
</tbody>
</table>

## Chapter 1: The Virtex-II Pro / Virtex-II Pro X FPGA Family

- The Next Logical Revolution ........................................ 21
- Built for Bandwidth .................................................. 21
- Legacy of Leadership .................................................. 22
- Packets Everywhere ................................................... 23
- Bridge, Anyone? ....................................................... 23
- Simplifying Complexity ............................................... 24
- Time Is Money ........................................................... 24
- Flexibility Is Money ................................................... 25
- Not Being Discrete .................................................... 25

## Chapter 2: Timing Models

- Summary ................................................................. 27
- Introduction ............................................................ 27
- CLB / Slice Timing Models .......................................... 28
  - General Slice Timing Model and Parameters .................. 28
  - Slice Distributed RAM Timing Model and Parameters .... 31
  - Slice SRL Timing Model and Parameters ...................... 34
- Block SelectRAM Timing Model ..................................... 36
- Embedded Multiplier Timing Model ............................... 40
- IOB Timing Models .................................................... 43
  - IOB Input Timing Model and Parameters ...................... 44
  - IOB Output Timing Model and Parameters ...................... 48
  - IOB 3-State Timing Model and Parameters .................... 51
- Pin-to-Pin Timing Models ............................................ 55
  - Global Clock Input to Output .................................. 56
  - Global Clock Setup and Hold ................................... 58
- Digital Clock Manager Timing Model ............................. 60
  - Operating Frequency Ranges .................................... 60
  - Input Clock Tolerances ........................................... 62
  - Output Clock Precision ........................................... 63
  - Miscellaneous DCM Timing Parameters ....................... 63
- Additional Timing Models in Other Publications: .......... 65
  - PPC405 Processor Block and RocketIO/RocketIO X Transceivers
Chapter 3: Design Considerations

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Block SelectRAM™ Memory</td>
<td>122</td>
</tr>
<tr>
<td>Introduction</td>
<td>122</td>
</tr>
<tr>
<td>Synchronous Dual-Port and Single-Port RAM</td>
<td>122</td>
</tr>
<tr>
<td>Characteristics</td>
<td>126</td>
</tr>
<tr>
<td>Library Primitives</td>
<td>126</td>
</tr>
<tr>
<td>VHDL and Verilog Instantiation</td>
<td>129</td>
</tr>
<tr>
<td>Port Signals</td>
<td>129</td>
</tr>
<tr>
<td>Address Mapping</td>
<td>131</td>
</tr>
<tr>
<td>Attributes</td>
<td>131</td>
</tr>
<tr>
<td>Initialization in VHDL or Verilog Codes</td>
<td>133</td>
</tr>
<tr>
<td>Location Constraints</td>
<td>133</td>
</tr>
<tr>
<td>Applications</td>
<td>133</td>
</tr>
<tr>
<td>VHDL and Verilog Templates</td>
<td>134</td>
</tr>
<tr>
<td>Distributed SelectRAM Memory</td>
<td>140</td>
</tr>
<tr>
<td>Introduction</td>
<td>140</td>
</tr>
<tr>
<td>Characteristics</td>
<td>142</td>
</tr>
<tr>
<td>Library Primitives</td>
<td>142</td>
</tr>
<tr>
<td>VHDL and Verilog Instantiation</td>
<td>143</td>
</tr>
<tr>
<td>Ports Signals</td>
<td>144</td>
</tr>
<tr>
<td>Attributes</td>
<td>144</td>
</tr>
<tr>
<td>Initialization in VHDL or Verilog Codes</td>
<td>145</td>
</tr>
<tr>
<td>Location Constraints</td>
<td>145</td>
</tr>
<tr>
<td>Applications</td>
<td>147</td>
</tr>
<tr>
<td>VHDL and Verilog Templates</td>
<td>148</td>
</tr>
<tr>
<td>Look-Up Tables as Shift Registers (SRLs)</td>
<td>150</td>
</tr>
<tr>
<td>Introduction</td>
<td>150</td>
</tr>
<tr>
<td>Section</td>
<td>Page</td>
</tr>
<tr>
<td>------------------------------------------------------------------------</td>
<td>------</td>
</tr>
<tr>
<td>Shift Register Operations</td>
<td>150</td>
</tr>
<tr>
<td>Characteristics</td>
<td>152</td>
</tr>
<tr>
<td>Library Primitives and Submodules</td>
<td>152</td>
</tr>
<tr>
<td>Initialization in VHDL and Verilog Code</td>
<td>155</td>
</tr>
<tr>
<td>Port Signals</td>
<td>155</td>
</tr>
<tr>
<td>Attributes</td>
<td>156</td>
</tr>
<tr>
<td>Location Constraints</td>
<td>156</td>
</tr>
<tr>
<td>Fully Synchronous Shift Registers</td>
<td>157</td>
</tr>
<tr>
<td>Static-Length Shift Registers</td>
<td>158</td>
</tr>
<tr>
<td>VHDL and Verilog Instantiation</td>
<td>159</td>
</tr>
<tr>
<td><strong>Large Multiplexers</strong></td>
<td>161</td>
</tr>
<tr>
<td>Introduction</td>
<td>161</td>
</tr>
<tr>
<td>Virtex-II Pro / Virtex-II Pro X CLB Resources</td>
<td>161</td>
</tr>
<tr>
<td>Wide-Input Multiplexers</td>
<td>165</td>
</tr>
<tr>
<td>Characteristics</td>
<td>165</td>
</tr>
<tr>
<td>Library Primitives and Submodules</td>
<td>166</td>
</tr>
<tr>
<td>Port Signals</td>
<td>166</td>
</tr>
<tr>
<td>Applications</td>
<td>167</td>
</tr>
<tr>
<td>VHDL and Verilog Instantiation</td>
<td>167</td>
</tr>
<tr>
<td><strong>Sum of Products (SOP) Logic</strong></td>
<td>171</td>
</tr>
<tr>
<td>Introduction</td>
<td>171</td>
</tr>
<tr>
<td>Virtex-II Pro / Virtex-II Pro X CLB Resources</td>
<td>171</td>
</tr>
<tr>
<td>VHDL Parameters</td>
<td>172</td>
</tr>
<tr>
<td>Applications</td>
<td>173</td>
</tr>
<tr>
<td>VHDL and Verilog Instantiation</td>
<td>173</td>
</tr>
<tr>
<td><strong>Embedded Multipliers</strong></td>
<td>177</td>
</tr>
<tr>
<td>Introduction</td>
<td>177</td>
</tr>
<tr>
<td>Two’s-Complement Signed Multiplier</td>
<td>178</td>
</tr>
<tr>
<td>Library Primitives and Submodules</td>
<td>178</td>
</tr>
<tr>
<td>VHDL and Verilog Instantiation</td>
<td>182</td>
</tr>
<tr>
<td>Port Signals</td>
<td>182</td>
</tr>
<tr>
<td>Location Constraints</td>
<td>182</td>
</tr>
<tr>
<td>Routing with BlockRAM</td>
<td>183</td>
</tr>
<tr>
<td>VHDL and Verilog Templates</td>
<td>184</td>
</tr>
<tr>
<td><strong>Single-Ended SelectIO-Ultra Resources</strong></td>
<td>185</td>
</tr>
<tr>
<td>Summary</td>
<td>185</td>
</tr>
<tr>
<td>Introduction</td>
<td>185</td>
</tr>
<tr>
<td>Fundamentals</td>
<td>186</td>
</tr>
<tr>
<td>Overview of Supported I/O Standards</td>
<td>187</td>
</tr>
<tr>
<td>Library Symbols</td>
<td>189</td>
</tr>
<tr>
<td>Design Considerations</td>
<td>198</td>
</tr>
<tr>
<td>Application Examples</td>
<td>206</td>
</tr>
<tr>
<td><strong>Digitally Controlled Impedance (DCI)</strong></td>
<td>223</td>
</tr>
<tr>
<td>Introduction</td>
<td>223</td>
</tr>
<tr>
<td>Xilinx DCI</td>
<td>223</td>
</tr>
<tr>
<td>Software Support</td>
<td>230</td>
</tr>
<tr>
<td>DCI in Virtex-II Pro / Virtex-II Pro X Hardware</td>
<td>235</td>
</tr>
<tr>
<td><strong>Double-Data-Rate (DDR) I/O</strong></td>
<td>241</td>
</tr>
<tr>
<td>Introduction</td>
<td>241</td>
</tr>
<tr>
<td>Data Flow</td>
<td>241</td>
</tr>
<tr>
<td>Characteristics</td>
<td>247</td>
</tr>
<tr>
<td>Library Primitives</td>
<td>247</td>
</tr>
</tbody>
</table>
Chapter 4: Configuration

Summary ........................................................................... 289

Introduction ..................................................................... 289
  Configuration Modes .................................................. 290
  Configuration Process and Flow .................................. 291
  Configuration Pins ...................................................... 294
  Mixed Voltage Environments ....................................... 296

Configuration Solutions .................................................. 297
  System ACE CF (CompactFlash) Solution ..................... 297
  Configuration PROMs ................................................ 298
  Flash PROMs With a CPLD Configuration Controller .... 299
  Embedded Solutions .................................................. 300
  PROM and System ACE CF Selection Guide .................. 301

Software Support and Data Files ..................................... 302
  iMPACT .................................................................... 302
Chapter 5: PCB Design Considerations

### Summary

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>383</td>
</tr>
</tbody>
</table>

### Pinout Information

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>383</td>
</tr>
</tbody>
</table>

#### Introduction

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>383</td>
</tr>
</tbody>
</table>

#### Pin Definitions

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>384</td>
</tr>
</tbody>
</table>

#### FG256/FGG256 Fine-Pitch BGA Package

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>386</td>
</tr>
</tbody>
</table>

### Pinout Diagrams

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>395</td>
</tr>
</tbody>
</table>

#### FG256/FGG256 Fine-Pitch BGA Pinout Diagrams

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>397</td>
</tr>
</tbody>
</table>

#### FG456/FGG456 Fine-Pitch BGA Pinout Diagrams

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>401</td>
</tr>
</tbody>
</table>

#### FG676/FGG676 Fine-Pitch BGA Pinout Diagrams

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>409</td>
</tr>
</tbody>
</table>

#### FP672 Flip-Chip Fine-Pitch BGA Pinout Diagrams

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>415</td>
</tr>
</tbody>
</table>

#### FF896 Flip-Chip Fine-Pitch BGA Pinout Diagrams

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>423</td>
</tr>
</tbody>
</table>

#### FF1152 Flip-Chip Fine-Pitch BGA Pinout Diagrams

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>431</td>
</tr>
</tbody>
</table>

#### FF1148 Flip-Chip Fine-Pitch BGA Pinout Diagrams

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>443</td>
</tr>
</tbody>
</table>

#### FF1517 Flip-Chip Fine-Pitch BGA Pinout Diagrams

<table>
<thead>
<tr>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>449</td>
</tr>
</tbody>
</table>
Appendix A: BitGen and PROMGen Switches and Options

Using BitGen ................................................................. 499
BitGen Syntax ............................................................. 500
BitGen Files ............................................................... 500
BitGen Options ......................................................... 501

Using PROMGen ......................................................... 507
PROMGen Syntax ....................................................... 508
PROMGen Files ......................................................... 508
PROMGen Options ..................................................... 509
Examples ................................................................. 512
Appendix B: Platform Flash Family PROMs

PROM Package Specifications ................................................................. 513
  FS48 Package Specification ............................................................... 514
  VO48 Package Specification ............................................................. 515
  VO20 Package Specification ............................................................. 516

Appendix C: Choosing the Battery for $V_{BATT}$

Battery Types and Chemistry Choices .................................................. 517
  Primary or Secondary ........................................................................ 517
  Battery Cost and Care ....................................................................... 518
  Battery Summary .............................................................................. 518
Design Examples .................................................................................. 519
  Case #1: ......................................................................................... 519
  Case #2: ......................................................................................... 519
  Case #3: ......................................................................................... 519
  Case #4: ......................................................................................... 520

Glossary ............................................................................................... 521

Index .................................................................................................. 551
Preface

About This Guide

This document, the Virtex-II Pro and Virtex-II Pro X FPGA User Guide, describes the function and operation of Virtex™-II Pro and Virtex-II Pro X devices, and also includes information on FPGA configuration techniques and PCB design considerations. It comprises the following main sections:

Guide Contents

- Chapter 1, “The Virtex-II Pro / Virtex-II Pro X FPGA Family”
- Chapter 2, “Timing Models”
- Chapter 3, “Design Considerations”
- Chapter 4, “Configuration”
- Chapter 5, “PCB Design Considerations”
- Appendix A, “BitGen and PROMGen Switches and Options”
- Appendix B, “Platform Flash Family PROMs”
- Appendix C, “Choosing the Battery for V_BATT”

For Virtex-II Pro device specifications, refer to the Virtex-II Pro Data Sheet:

- Virtex-II Pro and Virtex-II Pro X Platform FPGAs: Complete Data Sheet

The following documents offer in-depth technical design information about the RocketIO and RocketIO X multi-gigabit transceivers and PowerPC 405 processor, which are not covered in this User Guide:

- RocketIO Transceiver User Guide
- RocketIO X Transceiver User Guide
- PowerPC Processor Reference Guide
- PowerPC 405 Processor Block Reference Guide
Additional Resources

For additional information, go to http://support.xilinx.com. The following table lists some of the resources you can access from this website. You can also directly access these resources using the provided URLs.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Description/URL</th>
</tr>
</thead>
</table>
| Data Sheets         | Xilinx data sheets describe device-specific operating characteristics, architecture, and pinouts/packaging.  
                        | [http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp](http://www.xilinx.com/xlnx/xweb/xil_publications_index.jsp) |
| User Guides         | Xilinx user guides contain detailed, device-specific operating theory and generic design examples for various device functions.  
| Application Notes   | Xilinx application notes describe design techniques and approaches to general design as well as specific applications. Many application notes feature complete reference designs including source code.  
| Xcell Journals      | This site contains quarterly journals for Xilinx programmable logic users.  
| Tech Tips           | See this site for the latest news, design tips, and patch information on the Xilinx design environment.  
| Answers Database    | This database provides a current listing of solution records for Xilinx software tools. Search this database using the search function at:  

Conventions

This document uses the following conventions. An example illustrates each convention.

Typographical

The following typographical conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning or Use</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Courier font</td>
<td>Messages, prompts, and program files that the system displays</td>
<td>speed grade: - 100</td>
</tr>
<tr>
<td>Courier bold</td>
<td>Literal commands that you enter in a syntactical statement</td>
<td>ngdbuild design_name</td>
</tr>
<tr>
<td>Helvetica bold</td>
<td>Commands that you select from a menu</td>
<td>File → Open</td>
</tr>
<tr>
<td></td>
<td>Keyboard shortcuts</td>
<td>Ctrl+C</td>
</tr>
</tbody>
</table>
## Conventions

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning or Use</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>Italic font</em></td>
<td>Variables in a syntax statement for which you must supply values. (In angle brackets.)</td>
<td><code>ngdbuild &lt;design_name&gt;</code></td>
</tr>
<tr>
<td></td>
<td>References to other manuals</td>
<td>See the Development System Reference Guide for more information.</td>
</tr>
<tr>
<td></td>
<td>Emphasis in text</td>
<td>If a wire is drawn so that it overlaps the pin of a symbol, the two nets are <em>not</em> connected.</td>
</tr>
<tr>
<td>Square brackets [ ]</td>
<td>An optional entry or parameter. However, in bus specifications, such as <code>bus[7:0]</code>, they are required.</td>
<td><code>ngdbuild [option_name] design_name</code></td>
</tr>
<tr>
<td>Braces { }</td>
<td>A list of items from which you must choose one or more</td>
<td>`lowpwr ={on</td>
</tr>
<tr>
<td>Vertical bar</td>
<td></td>
<td>Separates items in a list of choices</td>
</tr>
</tbody>
</table>
| Vertical ellipsis . | Repetitive material that has been omitted                                      | IOB #1: Name = QOUT'
| .                   |                                                                                | IOB #2: Name = CLKN'
| .                   |                                                                                | .                       |
| Horizontal ellipsis . . | Repetitive material that has been omitted                                      | `allow block block_name loc1 loc2 ... locn;` |

### Online Document

The following conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning or Use</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Blue text</td>
<td>Cross-reference link to a location in the current document</td>
<td>See the section “Additional Resources” for details.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Refer to “Title Formats” in Chapter 1 for details.</td>
</tr>
<tr>
<td>Red text</td>
<td>Cross-reference link to a location in another document</td>
<td>See Figure 2-5 in the Virtex-II Handbook.</td>
</tr>
<tr>
<td>Blue, underlined text</td>
<td>Hyperlink to a website (URL)</td>
<td>Go to <a href="http://www.xilinx.com">http://www.xilinx.com</a> for the latest speed files.</td>
</tr>
</tbody>
</table>
Chapter 1

The Virtex-II Pro / Virtex-II Pro X FPGA Family

The Next Logical Revolution

The Virtex-II Pro™/Virtex-II Pro X Platform FPGA solution is the most technically sophisticated silicon and software product development in the history of the programmable logic industry. The goal was to revolutionize system architecture “from the ground up.” To achieve that objective, the best circuit engineers and system architects from IBM, Mindspeed, and Xilinx co-developed the world’s most advanced Platform FPGA silicon product. Leading teams from top embedded systems companies worked together with Xilinx software teams to develop the systems software and IP solutions that enabled this new system architecture paradigm. The result is the first Platform FPGA solution capable of implementing high performance system-on-a-chip designs previously the exclusive domain of custom ASICs, yet with the flexibility and low development cost of programmable logic. The Virtex-II Pro/Virtex-II Pro X family marks the first paradigm change from programmable logic to programmable systems, with profound implications for leading-edge system architectures in networking applications, deeply embedded systems, and digital signal processing systems. It allows custom user-defined system architectures to be synthesized, next-generation connectivity standards to be seamlessly bridged, and complex hardware and software systems to be co-developed rapidly with in-system debug at system speeds. Together, these capabilities usher in the next programmable logic revolution.

Built for Bandwidth

The Virtex-II Pro/Virtex-II Pro X family consists of eleven members. The nine Virtex-II Pro devices contain four to twenty RocketIO™ multi-gigabit transceivers (MGTs), while the two Virtex-II Pro X devices contain eight or twenty RocketIO X MGTs. Each Xilinx RocketIO/RocketIO X transceiver block contains a complete set of user-configurable supporting circuitry that addresses real-life, system-level challenges. These include standard 8B/10B encode/decode (plus 64B/66B encode/decode in the RocketIO X), programmable signal integrity adjustments for varying PCB trace lengths and materials, support for synchronization of multiple channels, and programmable support for channel control commands. In addition, the RocketIO and RocketIO X blocks are the first FPGA-embedded transceivers to reach baud rates of 3.125 Gbps and 10.3125 Gbps, respectively. Four RocketIO transceivers, employing sixteen PCB traces, can be used to support a full-duplex 10 Gbps channel by way of the RocketIO channel-bonding feature—or, a single RocketIO X transceiver can implement the same speed with just four PCB traces. This is equivalent to 256 traces of typical LVTTL buses, or 68 traces of a high-speed, source-synchronous parallel LVDS bus. It allows a PCB trace reduction of up to 64X over conventional parallel buses, resulting in significant reductions in PCB complexity and EMI...
system noise. The RocketIO/RocketIO X technology fulfills higher bandwidth system requirements than currently possible, with cost savings coming from faster time-to-market, reduced printed circuit board (PCB) complexity, and lower component count.

Each of the larger devices incorporates one or two small yet powerful IBM® PowerPC™ 405 processor cores, each capable of more than 300 MHz clock frequency and 420 Dhrystone MIPS. While the processor cores occupy a small area of the die, they provide tremendous system flexibility where they are used. The PowerPC 405 cores are fully embedded within the FPGA fabric, where all processor nodes are controlled by the FPGA routing resources. This provides the utmost architectural capability, where complex applications may be efficiently divided between high-speed logic implementation and high-flexibility software implementations. For example, a packet processing application using only the FPGA logic today for high-speed packet routing may be augmented to include a slave high-performance processor for exception handling or in-system statistics monitoring. In contrast, using a separate processor externally requires hundreds of additional interface pins, which degrades system performance and significantly increases FPGA I/O requirements and overall board costs.

The Virtex-II Pro/Virtex-II Pro X products are based on the most advanced FPGA fabric available: the Virtex-II™ architecture with IP-Immersion™ technology, which was developed to offer significant improvements in engineering productivity, silicon efficiency, and system flexibility. Unique features common in the Virtex-II Series—consisting of the Virtex-II and Virtex-II Pro families—include powerful SystemIO™ system connectivity solutions, digitally controlled impedance (DCI) technology, comprehensive clocking solutions, high-speed Active Interconnect™ routing architecture, and bitstream encryption. These features together constitute the most complete Platform FPGA solution available, optimized for high performance system-level applications. The upward compatibility of the Virtex Series of products ensures benefits in engineering productivity, performance, design longevity, and continuing cost reduction.

Legacy of Leadership

Each of the Virtex families of FPGAs has been the most successful programmable product family in its class, starting with the introduction of the original Virtex family in 1998. The Virtex and Virtex-E families were recognized by the industry as the highest technology products available when they were first introduced. The Virtex-II family, which again achieved technology leadership in density, performance, and features, ushered in the era of Platform FPGAs—programmable devices with the system-level capability and performance to implement systems functionality. The Virtex-II Pro family continues the tradition of technology leadership as the most sophisticated Platform FPGA yet, again breaking the technology barrier for the benefit of leading-edge system architects.

The Virtex-II Pro/Virtex-II Pro X family is the first FPGA family to incorporate both serial transceiver technology and a hard processor core within a general-purpose FPGA device. This is significant for new high-bandwidth embedded processing applications such as packet processing, where both high device I/O bandwidth and high performance processor cores are needed together.

The Virtex-II Pro/Virtex-II Pro X devices are the industry’s first FPGAs in a 0.13-micron process. The nine-layer metal, all-copper, low-k process technology is among the most advanced in the semiconductor industry. The combination of advanced Active Interconnect™ architecture and advanced process technology makes the Virtex-II Pro/Virtex-II Pro X family the highest performance FPGA in the world.

The RocketIO/RocketIO X multi-gigabit MGTs are the highest performance, most complete embedded serial transceivers available. They are user-configurable for up to
3.125 Gbps or 10.3125 Gbps baud rate per channel, which is many times the performance of other embedded transceivers at 1.25 Gbps. Each RocketIO/RocketIO X transceiver provides a complete set of common functionality available in standard SerDes transceivers. In contrast, “programmable ASSP” products with clock/data recovery (CDR) provide only the most basic transceiver capability.

The IBM PowerPC 405 processor core used in the Virtex-II Pro/Virtex-II Pro X family is the highest performance embedded core available in FPGAs. The PowerPC architecture is used in many markets including communications, industrial control, test and measurement systems, and other performance-oriented markets. It is currently the most popular processor architecture in embedded applications.

Packets Everywhere

The Virtex-II Pro/Virtex-II Pro X family provides a powerful new paradigm for network processing where low latency is required, such as storage area networks, wireless infrastructure, and voice-over-IP networks. The digital convergence phenomenon drives the need for packet routing based on type and priority. For example, live voice and video data packets require significantly lower latency than data file packets. New data networking applications must now handle higher bandwidth traffic as well as more complex types of prioritized packets. In many cases, Virtex-II Pro/Virtex-II Pro X devices can offer higher overall performance than other solutions, including specialized network processors (NPs). Using the Virtex-II Pro architecture, the most common packets may be quickly read and routed using FPGA logic, without incurring the lengthy software runtime needed by NPs. The FPGA logic interrupts the PowerPC processor core only when processor instructions are needed for special packet types. For example, packets may be stored into a 16 KB dual-port memory area accessible by both the FPGA logic and the PowerPC 405 on-chip memory (OCM) port, allowing rapid change of control and packet disposition. By using the FPGA logic to process the most common packet types while the processor core handles the more specialized ones as a slave to the logic, the Virtex-II Pro architecture can provide higher overall performance than NPs, as well as more sophisticated processing capabilities than FPGA logic alone.

Bridge, Anyone?

Powerful protocol bridges for tying together disparate data stream formats are well-suited for the Virtex-II Pro/Virtex-II Pro X solution. New interface standards and protocols include PCI Express™, Infiniband®, Gigabit Ethernet, XAUI/10 Gigabit Ethernet, RapidIO™, and HyperTransport™. These must interface seamlessly to one another, as well as to other standards such as PCI™, Fibre Channel, POS Phy Level 4, Flexbus 4, and others. This presents a significant challenge to system developers because of changing standards, scarcity of off-the-shelf interface components, and the inflexibility of available solutions. System designers have had to assemble their own blend of FPGAs, discrete physical transceivers, and discrete communications processors to solve their complex system challenges. Even newer “programmable ASSPs” (application-specific standard products) with built-in serial transceivers fall short, because they frequently require companion FPGAs to supplement their logic capacity. The Virtex-II Pro solution, using the powerful Xilinx SystemIO™ capability to fully integrate silicon, software, and IP capabilities, provides the most flexible pre-engineered protocol bridge solutions available for fast time-to-market and low development cost.
Simplifying Complexity

The Virtex-II Pro/Virtex-II Pro X solution offers a powerful paradigm for complex embedded systems found in signal processing, industrial control, image processing, networking, communications, and aeronautic applications. For the first time, complex embedded systems traditionally involving sophisticated hardware and software may be developed concurrently, emulated in actual hardware at speed, debugged in-system, and re-architected for performance within weeks, rather than months or years. In addition, full systems can be remotely upgraded as easily as software-only upgrades are performed today, using Compact Flash, CDROM, Internet, wireless transmission, or other flexible means. Hardware design is simplified using powerful development software and a large soft IP library to assemble logic- and processor-based platforms. Software development may be started earlier using the actual device in preconfigured sample platforms, without waiting for the new system board to be developed. In many cases, higher density Virtex-II Pro components may be used for early system development, whereby extra resources (including additional PowerPC processor cores) may be used to easily emulate board-level components yet to be developed. This flexibility, obviously unavailable in custom ASICs or ASSPs, allows systems to be emulated at speed, rather than simulated using software simulators at 100 or 1000 times slower. In-system debugging is further enhanced by the Xilinx ChipScope Pro tool, which provides comprehensive logic analysis—from probing internal nodes to full bus analysis with bus protocol adherence checks using an external logic analyzer via the IEEE 1149.1 (JTAG) test access port. Using ChipScope Pro can result in orders of magnitude of improvement in engineering productivity.

Complex systems can be optimally repartitioned between FPGA logic and processor cores, allowing a continuum of possible trade-offs between the speed of logic and the flexibility of software code. For example, a first implementation of an echo cancellation algorithm might be all-software in compiled C code running on a PowerPC core, in order to allow the system software development to start. As the system is further optimized, part of the DSP algorithm could be retargeted using Matlab Simulink into FPGA logic to achieve a significantly faster but functionally identical system for production release. In another example, an encryption application might implement the Diffie-Hellman key exchange algorithm, whereby exponentiation and message management could be optimally partitioned into FPGA logic and an embedded processor, respectively. In this way, the programmable systems paradigm offers tremendous flexibility to allow system designers and architects to optimize the trade-offs in development time, system performance, and system costs.

It is significant that the embedded systems enabled by Virtex-II Pro solutions are "all-soft," in that both logic and software code are controlled by a soft data file. Because of this, the low cost of design maintenance and degree of design reuse is greatly enhanced. Whole system upgrades, including both hardware and software, can now be accomplished with one unified soft file using System ACE™ configuration solutions, offering the same low cost and ease of use as software-only upgrades.

Time Is Money

The Virtex-II Series, comprising both the Virtex-II and Virtex-II Pro/Virtex-II Pro X families, offers significantly faster time-to-market and lower development costs than ASICs. Compared to a full-custom ASIC, the Virtex-II Pro solution eliminates the need for exhaustive verification during development, and allows hardware-software debug at system speeds rather than at slow software simulation speeds. In addition, the
Flexibility Is Money

The Virtex-II Pro/Virtex-II Pro X features of signal integrity, pre-engineered clocking capabilities, and an abundance of soft IP cores, significantly reduce development time.

The Virtex-II Series offers significantly lower development costs than ASICs, due to lower tool costs, lower third-party IP costs, and absence of NRE costs. The Virtex-II Series also increases engineering productivity by accelerating hardware availability for software development and increasing software debug speed. In addition, the availability of powerful development tools enables straightforward retargeting of other embedded processors into the PowerPC platform. Compared to other processor architectures, the PowerPC 405 core in most cases allows higher performance and more powerful capabilities, and thus can be used to accelerate preproduction of performance-sensitive applications.

Flexibility Is Money

The flexibility inherent in the Virtex-II Series allows system architects to fine-tune their architectural partitioning after the initial prototype is developed. That is, each subsystem function can be freely implemented as hardware only, software only, or any combination within the hardware-software continuum, depending on the trade-off between performance and complexity. For example, a wireless infrastructure system might initially implement a rake filter function in hardware, and then change to a firmware implementation as more software control is necessary during later development. This repartitioning would be impossible in custom ASICs without significant time and cost penalties.

The Virtex-II Series offers significantly more flexibility than fixed chip sets and ASSPs, allowing end user product differentiation and future-proofing. For a design requirement that can generally be met either by ASSPs or by Virtex-II Platform FPGAs, the initial design investment for an FPGA implementation may be higher. However, the advantages for Platform FPGA implementations include customizing of functionality, ease of design reuse, ability to fix design bugs, differentiation of user end products, and ownership and control of the entire system. These are important advantages in highly competitive markets where ASSPs have standing errata lists and unpredictable future availability. In contrast, properly developed Platform FPGA designs are soft designs that may be readily maintained and reused as needed. Therefore, FPGA methodologies can provide system manufacturers with greater competitive advantage in the short term, and greater ownership and control over their products in the long term.

Not Being Discrete

Many high-bandwidth systems today use large FPGAs together with discrete SerDes transceivers, discrete communications processors, or other discrete components. The Virtex-II Pro/Virtex-II Pro X family can eliminate the need for many of these external components, enhancing time-to-market and performance, even providing system cost benefits in many cases. Multi-chip solutions using FPGAs typically require over a hundred I/O pins to interface to each discrete quad 3.125 Gbps SerDes transceiver or discrete microprocessor. The result is increased PCB complexity to accommodate the hundreds of traces, reduced system performance due to on-chip/off-chip connections, and higher overall system costs. In some cases, the increased FPGA pin-count requirement may force a higher-density FPGA to be used, again increasing the overall cost. In these cases, the Virtex-II Pro/Virtex-II Pro X devices can integrate the discrete components to achieve faster system development, higher system performance, and lower costs.
Chapter 2

Timing Models

Summary

The following topics are covered in this chapter:

- CLB / Slice Timing Models
- Block SelectRAM Timing Model
- Embedded Multiplier Timing Model
- IOB Timing Models
- Pin-to-Pin Timing Models
- Digital Clock Manager Timing Model
- Additional Timing Models in Other Publications: PPC405 Processor Block and RocketIO/RocketIO X Transceivers

Introduction

Due to the large size and complexity of Virtex-II Pro/Virtex-II Pro X FPGAs, understanding the timing associated with the various paths and functional elements has become a difficult and important problem. Although it is not necessary to understand the various timing parameters in order to implement most designs using Xilinx software, a thorough timing model can assist advanced users in analyzing critical paths or planning speed-sensitive designs.

The Timing Model chapter is broken up into sections consisting of three basic components:

- Functional Element Diagram - basic architectural schematic illustrating pins and connections.
- Timing Parameters - Virtex-II Pro and Virtex-II Pro X Platform FPGA Data Sheet timing parameter definitions.
- Timing Diagram - illustrates functional element timing parameters relative to each other.

This chapter was written with the Xilinx Timing Analyzer software (TRCE) in mind. All pin names, parameter names, and paths are consistent with Post Route Timing and Pre-Route Static Timing reports. Use the models in this chapter in conjunction with both the Timing Analyzer software and the section on switching characteristics in the Virtex-II Pro Platform FPGA Data Sheet. Most of the timing parameters found in the section on switching characteristics are described in this chapter.
CLB / Slice Timing Models

The three sections below describe all timing parameters reported in the Data Sheet that are associated with slices and Configurable Logic Blocks (CLBs). The sections correspond to their respective (switching characteristics) sections in the data sheet:

- “General Slice Timing Model and Parameters” (CLB Switching Characteristics)
- “Slice Distributed RAM Timing Model and Parameters” (CLB Distributed RAM Switching Characteristics)
- “Slice SRL Timing Model and Parameters” (CLB SRL Switching Characteristics)

General Slice Timing Model and Parameters

Figure 2-1 illustrates the details of a Virtex-II Pro/Virtex-II Pro X slice.

Note: Some elements of the Virtex-II Pro/Virtex-II Pro X slice have been omitted for clarity. Only the elements relevant to the timing paths described in this section are shown.
## Timing Parameters

### Table 2-1: General Slice Timing Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Control Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Combinatorial Delays</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{ILO}$</td>
<td>F/G inputs to X/Y outputs</td>
<td></td>
<td>Propagation delay from the F/G inputs of the slice, through the look-up tables (LUTs), to the X/Y outputs of the slice.</td>
</tr>
<tr>
<td>$T_{IF5}$</td>
<td>F/G inputs to F5 output</td>
<td></td>
<td>Propagation delay from the F/G inputs of the slice, through the LUTs and MUXF5 to the F5 output of the slice.</td>
</tr>
<tr>
<td>$T_{IF5X}$</td>
<td>F/G inputs to X output</td>
<td></td>
<td>Propagation delay from the F/G inputs of the slice, through the LUTs and MUXF5 to the X output of the slice.</td>
</tr>
<tr>
<td>$T_{IFXY}$</td>
<td>FXINA/FXINB inputs to Y output</td>
<td></td>
<td>Propagation delay from the FXINA/FXINB inputs, through MUXFX to the Y output of the slice.</td>
</tr>
<tr>
<td>$T_{IFNCTL}$</td>
<td>Transparent Latch input to XQ/YQ outputs</td>
<td></td>
<td>Incremental delay through a transparent latch to XQ/YQ outputs.</td>
</tr>
<tr>
<td><strong>Sequential Delays</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{CKO}$</td>
<td>FF Clock (CLK) to XQ/YQ outputs</td>
<td></td>
<td>Time after the clock that data is stable at the XQ/YQ outputs of the slice sequential elements (configured as a flip-flop).</td>
</tr>
<tr>
<td>$T_{CKLO}$</td>
<td>Latch Clock (CLK) to XQ/YQ outputs</td>
<td></td>
<td>Time after the clock that data is stable at the XQ/YQ outputs of the slice sequential elements (configured as a latch).</td>
</tr>
<tr>
<td><strong>Setup and Hold for Slice Sequential Elements</strong></td>
<td></td>
<td></td>
<td>The following descriptions are for setup times only.</td>
</tr>
<tr>
<td>$T_{DICK}/T_{CKDI}$</td>
<td>BX/BY inputs</td>
<td></td>
<td>Time before Clock (CLK) that data from the BX or BY inputs of the slice must be stable at the D-input of the slice sequential elements (configured as a flip-flop).</td>
</tr>
<tr>
<td>$T_{DYCK}/T_{CKDY}$</td>
<td>DY input</td>
<td></td>
<td>Time before Clock (CLK) that data from the DY input of the slice must be stable at the D-input of the slice sequential elements (configured as a flip-flop).</td>
</tr>
<tr>
<td>$T_{DXCK}/T_{CKDX}$</td>
<td>DX input</td>
<td></td>
<td>Time before Clock (CLK) that data from the DX input of the slice must be stable at the D-input of the slice sequential elements (configured as a flip-flop).</td>
</tr>
<tr>
<td>$T_{CECK}/T_{CKCE}$</td>
<td>CE input</td>
<td></td>
<td>Time before Clock (CLK) that the CE (Clock Enable) input of the slice must be stable at the CE-input of the slice sequential elements (configured as a flip-flop).</td>
</tr>
</tbody>
</table>
Figure 2-2 illustrates general timing characteristics of a Virtex-II Pro/Virtex-II Pro X slice.

**Figure 2-2: General Slice Timing Characteristics**

- At time $T_{\text{CECK}}$ before Clock Event 1, the Clock-Enable signal becomes valid-high at the CE input of the slice register.
- At time $T_{\text{DYCK}}$ before Clock Event 1, data from the DY input becomes valid-high at the D input of the slice register and is reflected on the YQ pin at time $T_{\text{CKO}}$ after Clock Event 1*.
- At time $T_{\text{RCK}}$ before Clock Event 3, the SR signal (configured as synchronous reset in this case) becomes valid-high, resetting the slice register, and this is reflected on the YQ pin at time $T_{\text{CKO}}$ after Clock Event 3.

**Note:** In most cases software uses the DX/DY inputs to route data to the slice registers when at all possible. This is the fastest path to the slice registers and saves other slice routing resources.

---

### Table 2-1: General Slice Timing Parameters (Continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Control Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{\text{RCK}}/T_{\text{CKR}}$</td>
<td>SR/BY inputs</td>
<td></td>
<td>Time before CLK that the SR (Set/Reset) and the BY (Rev) inputs of the slice must be stable at the SR/Rev-inputs of the slice sequential elements (configured as a flip-flop). Synchronous set/reset only.</td>
</tr>
<tr>
<td>Clock CLK</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{\text{CH}}$</td>
<td></td>
<td></td>
<td>Minimum Pulse Width, High.</td>
</tr>
<tr>
<td>$T_{\text{CL}}$</td>
<td></td>
<td></td>
<td>Minimum Pulse Width, Low.</td>
</tr>
<tr>
<td>Set/Reset</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{\text{RPW}}$</td>
<td></td>
<td></td>
<td>Minimum Pulse Width for the SR (Set/Reset) and BY (Rev) pins.</td>
</tr>
<tr>
<td>$T_{\text{RQ}}$</td>
<td></td>
<td></td>
<td>Propagation delay for an asynchronous Set/Reset of the slice sequential elements. From SR/BY inputs to XQ/YQ outputs.</td>
</tr>
<tr>
<td>$F_{\text{TOG}}$</td>
<td></td>
<td></td>
<td>Toggle Frequency - Maximum Frequency that a CLB flip-flop can be clocked: $1/(T_{\text{CH}}+T_{\text{CL}})$</td>
</tr>
</tbody>
</table>
Slice Distributed RAM Timing Model and Parameters

Figure 2-3 illustrates the details of distributed RAM implemented in a Virtex-II Pro / Virtex-II Pro X slice.

**Note:** Some elements of the Virtex-II Pro/Virtex-II Pro X slice have been omitted for clarity. Only the elements relevant to the timing paths described in this section are shown.

![Virtex-II Pro/Virtex-II Pro X Slice Distributed RAM](image-url)
Timing Parameters

**Table 2-2: Slice Distributed RAM Timing Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Control Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{SHCKO16} )</td>
<td>CLK to X/Y outputs (WE active) in 16x1 mode</td>
<td></td>
<td>Time after the Clock (CLK) of a WRITE operation that the data written to the distributed RAM (in 16x1 mode) is stable on the X/Y outputs of the slice.</td>
</tr>
<tr>
<td>( T_{SHCKO32} )</td>
<td>CLK to X/Y outputs (WE active) in 32x1 mode</td>
<td></td>
<td>Time after the Clock (CLK) of a WRITE operation that the data written to the distributed RAM (in 32x1 mode) is stable on the X/Y outputs of the slice.</td>
</tr>
<tr>
<td>( T_{SHCKOF5} )</td>
<td>CLK to F5 output (WE active)</td>
<td></td>
<td>Time after the Clock (CLK) of a WRITE operation that the data written to the distributed RAM is stable on the F5 output of the slice.</td>
</tr>
</tbody>
</table>

**Setup and Hold for Slice LUT Configured as RAM (Distributed RAM)**

\[ T_s = \text{Setup time (before clock edge)} \]
\[ T_{sh} = \text{Hold time (after clock edge)} \]

| \( T_{DS}/T_{DH} \) | BX/BY Data inputs (DI)                      |                | Time before the clock that data must be stable at the DI input of the slice LUT (configured as RAM), via the slice BX/BY inputs. |
| \( T_{AS}/T_{AH} \) | F/G Address inputs                          |                | Time before the clock that address signals must be stable at the F/G inputs of the slice LUT (configured as RAM). |
| \( T_{WES}/T_{WEH} \) | WE input (SR)                               |                | Time before the clock that the Write Enable signal must be stable at the WE input of the slice LUT (configured as RAM). |

**Clock CLK**

| \( T_{WPH} \)   |                                              | Minimum Pulse Width, High (for a Distributed RAM clock). |
| \( T_{WPL} \)   |                                              | Minimum Pulse Width, Low (for a Distributed RAM clock). |
| \( T_{WC} \)    |                                              | Minimum clock period to meet address write cycle time. |
Timing Characteristics

Figure 2-4 illustrates the timing characteristics of a 16-bit distributed RAM implemented in a Virtex-II Pro/Virtex-II Pro X slice (LUT configured as RAM).

**Clock Event 1: WRITE Operation**
During a WRITE operation, the contents of the memory at the address on the ADDR inputs is changed. The data written to this memory location is reflected on the X/Y outputs synchronously.

- At time $T_{WES}$ before Clock Event 1, the Write Enable signal (WE) becomes valid-high, enabling the RAM for the following WRITE operation.
- At time $T_{AS}$ before Clock Event 1, the address (2) becomes valid at the F/G inputs of the RAM.
- At time $T_{DS}$ before Clock Event 1, the DATA becomes valid (1) at the DI input of the RAM and is reflected on the X/Y output at time $T_{SHCK016}$ after Clock Event 1.

**Clock Event 2: READ Operation**
All READ operations are asynchronous in distributed RAM. As long as write-enable (WE) is Low, the address bus can be asserted at any time, and the contents of the RAM at that address are reflected on the X/Y outputs after a delay of length $T_{ILO}$ (propagation delay through a LUT). Note that the Address (F) is asserted after Clock Event 2, and that the contents of the RAM at that location are reflected on the output after a delay of length $T_{ILO}$.
Slice SRL Timing Model and Parameters

Figure 2-5 illustrates shift register implementation in a Virtex-II Pro/Virtex-II Pro X slice.

**Note:** Note: Some elements of the Virtex-II Pro/Virtex-II Pro X slice have been omitted for clarity. Only the elements relevant to the timing paths described in this section are shown.
Timing Parameters

Table 2-3: Slice SRL Timing Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Control Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_{\text{REG}} )</td>
<td>CLK to X/Y outputs</td>
<td></td>
<td>Time after the Clock (CLK) of a WRITE operation that the data written to the SRL is stable on the X/Y outputs of the slice.</td>
</tr>
<tr>
<td>( T_{\text{CKSH}} )</td>
<td>CLK to Shiftout</td>
<td></td>
<td>Time after the Clock (CLK) of a WRITE operation that the data written to the SRL is stable on the Shiftout or XB/YB outputs of the slice.</td>
</tr>
<tr>
<td>( T_{\text{REGF5}} )</td>
<td>CLK to F5 output</td>
<td></td>
<td>Time after the Clock (CLK) of a WRITE operation that the data written to the SRL is stable on the F5 output of the slice.</td>
</tr>
</tbody>
</table>

Setup/Hold for Slice LUT Configured as SRL (Select Shift Register)

- \( T_{\text{WXS}} \): Setup time (before clock edge)
- \( T_{\text{WXS}} \): Hold time (after clock edge)

- \( T_{\text{SRDS}}/T_{\text{SRDH}} \): BX/BY Data inputs (DI)

The following descriptions are for setup times only.

- \( T_{\text{WSS}}/T_{\text{WSH}} \): CE input (WE)

Time before the clock that data must be stable at the DI input of the slice LUT (configured as SRL), via the slice BX/BY inputs.

Clock CLK

- \( T_{\text{SRPH}} \): Minimum Pulse Width, High (for SRL clock).
- \( T_{\text{SRPL}} \): Minimum Pulse Width, Low (for SRL clock).

Timing Characteristics

Figure 2-6 illustrates the timing characteristics of a 16-bit shift register implemented in a Virtex-II Pro/Virtex-II Pro X slice (LUT configured as SRL).

Figure 2-6: Slice SRL Timing Characteristics
Clock Event 1: Shift_In

During a WRITE (Shift_In) operation, the single-bit content of the register at the address on the ADDR inputs is changed, as data is shifted through the SRL. The data written to this register is reflected on the X/Y outputs synchronously, if the address is unchanged during the clock event. If the ADDR inputs are changed during a clock event, the value of the data at the addressable output (D) is invalid.

- At time T_WSS before Clock Event 1, the Write Enable signal (SR) becomes valid-high, enabling the SRL for the WRITE operation that follows.
- At time T_SRLOCKS before Clock Event 1 the data becomes valid (0) at the DI input of the SRL and is reflected on the X/Y output after a delay of length T_REG after Clock Event 1.*
  
  Note: Since the address 0 is specified at Clock Event 1, the data on the DI input is reflected at the D output, because it is written to Register 0.

Clock Event 2: Shift_In

- At time T_SRLDS before Clock Event 2, the data becomes valid (1) at the DI input of the SRL and is reflected on the X/Y output after a delay of length T_REG after Clock Event 2.*
  
  Note: Since the address 0 is still specified at Clock Event 2, the data on the DI input is reflected at the D output, because it is written to Register 0.

Clock Event 3: Shift_In / Addressable (Asynchronous) READ

All READ operations are asynchronous. If the address is changed (between clock events), the contents of the register at that address are reflected at the addressable output (X/Y outputs) after a delay of length T_ILO (propagation delay through a LUT).

- At time T_SRLDS before Clock Event 3 the Data becomes valid (1) at the DI input of the SRL, and is reflected on the X/Y output T_REG time after Clock Event 3.
- Notice that the address is changed (from 0 to 2) some time after Clock Event 3. The value stored in Register 2 at this time is a 0 (in this example, this was the first data shifted in), and it is reflected on the X/Y output after a delay of length T_ILO.

Clock Event 16: MSB (Most Significant Bit) Changes

- At time T_REGXB after Clock Event 16, the first bit shifted into the SRL becomes valid (logical 0 in this case) on the XB output of the slice via the MC15 output of the LUT (SRL).

Block SelectRAM Timing Model

Introduction

This section describes the timing parameters associated with the block SelectRAM (illustrated in Figure 2-7) in Virtex-II Pro/Virtex-II Pro X FPGA devices. This section is intended to be used with the section on switching characteristics in the Virtex-II Pro Platform FPGA Data Sheet and the Timing Analyzer (TRCE) report from Xilinx software. For specific timing parameter values, refer to the switching characteristics section in the Virtex-II Pro Platform FPGA Data Sheet.
Block SelectRAM Timing Model

Timing Parameters

Table 2-4: Block SelectRAM Timing Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Control Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Setup and Hold Relative to Clock (CLK)</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{BCLK}$ = Setup time (before clock edge) $T_{BCKS}$ = Hold time (after clock edge)</td>
<td>Address inputs ADDR</td>
<td></td>
<td>The following descriptions are for setup times only.</td>
</tr>
<tr>
<td>$T_{BACK}$/$T_{BCKA}$</td>
<td>Address inputs ADDR</td>
<td></td>
<td>Time before the clock that address signals must be stable at the ADDR inputs of the block RAM.</td>
</tr>
<tr>
<td>$T_{BDCK}$/$T_{BCKD}$</td>
<td>Data inputs DI</td>
<td></td>
<td>Time before the clock that data must be stable at the DI inputs of the block RAM.</td>
</tr>
<tr>
<td>$T_{BECK}$/$T_{BCKE}$</td>
<td>Enable EN</td>
<td></td>
<td>Time before the clock that the enable signal must be stable at the EN input of the block RAM.</td>
</tr>
<tr>
<td>$T_{BRCK}$/$T_{BCKR}$</td>
<td>Synchronous Set/Reset SSR</td>
<td></td>
<td>Time before the clock that the synchronous set/reset signal must be stable at the SSR input of the block RAM.</td>
</tr>
<tr>
<td>$T_{BWCK}$/$T_{BCKW}$</td>
<td>Write Enable WE</td>
<td></td>
<td>Time before the clock that the write enable signal must be stable at the WE input of the block RAM.</td>
</tr>
<tr>
<td><strong>Clock to Out</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{BCKO}$</td>
<td>Clock to Output CLK to DO</td>
<td></td>
<td>Time after the clock that the output data is stable at the DO outputs of the block RAM.</td>
</tr>
<tr>
<td><strong>Clock</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{BPWH}$</td>
<td>Clock CLK</td>
<td></td>
<td>Minimum pulse width, high.</td>
</tr>
<tr>
<td>$T_{BPWL}$</td>
<td>Clock CLK</td>
<td></td>
<td>Minimum pulse width, low.</td>
</tr>
</tbody>
</table>
Timing Characteristics

The timing diagram in Figure 2-8 describes a single-port block RAM in Write-First mode. The timing for Read-First and No-Change modes are similar (see Chapter 3, “Design Considerations”).

At time 0, the block RAM is disabled; EN (enable) is low.

Clock Event 1: READ Operation

During a read operation, the contents of the memory at the address on the ADDR inputs are unchanged.

- T_BACK before Clock Event 1, address 00 becomes valid at the ADDR inputs of the block RAM.
- At time T_BECK before Clock Event 1, Enable goes High at the EN input of the block RAM, enabling the memory for the READ operation that follows.
- At time T_BCKO after Clock Event 1, the contents of the memory at address 00 become stable at the DO pins of the block RAM.

Clock Event 2: WRITE Operation

During a write operation, the content of the memory at the location specified by the address on the ADDR inputs is replaced by the value on the DI pins and is immediately reflected on the output latches (in WRITE-FIRST mode); EN (enable) is high.

- At time T_BACK before Clock Event 2, address 0F becomes valid at the ADDR inputs of the block RAM.
- At time T_BDCK before Clock Event 2, data CCCC becomes valid at the DI inputs of the block RAM.
- At time T_BWCK before Clock Event 2, Write Enable becomes valid at the WE following the block RAM.
- At time T_BCKO after Clock Event 2, data CCCC becomes valid at the DO outputs of the block RAM.
Clock Event 4: SSR (Synchronous Set/Reset) Operation

During an SSR operation, initialization parameter value SRVAL is loaded into the output latches of the block SelectRAM. The SSR operation does NOT change the contents of the memory and is independent of the ADDR and DI inputs.

- At time $T_{BRCK}$ before Clock Event 4, the synchronous set/reset signal becomes valid (High) at the SSR input of the block RAM.
- At time $T_{BCKO}$ after Clock Event 4, the SRVAL 0101 becomes valid at the DO outputs of the block RAM.

Clock Event 5: Disable Operation

De-asserting the enable signal EN disables any write, read or SSR operation. The disable operation does NOT change the contents of the memory or the values of the output latches.

- At time $T_{BECK}$ before Clock Event 5, the enable signal becomes valid (Low) at the EN input of the block RAM.
- After Clock Event 5, the data on the DO outputs of the block RAM is unchanged.

Timing Model

Figure 2-9 illustrates the delay paths associated with the implementation of block SelectRAM. This example takes the simplest paths on and off chip (these paths can vary greatly depending on the design). This timing model demonstrates how and where the block SelectRAM timing parameters are used.

NET = Varying interconnect delays
$T_{IOPI} = $ Pad to I-output of IOB delay
$T_{IOOP} = $ O-input of IOB to pad delay
$T_{GIOO} = $ BUFGMUX delay
Embedded Multiplier Timing Model

Introduction

This section explains all timing parameters associated with the use of embedded 18-bit x 18-bit multipliers in Virtex-II Pro/Virtex-II Pro X FPGAs (see Figure 2-10). The propagation delays through the embedded multiplier differ based on the size of the multiplier function implemented. The longest delay through the multiplier is to the highest order bit output (P35). Therefore, if an 18-bit x 18-bit signed multiplier is implemented, the worst-case delay for this function is the longest delay associated with the embedded multiplier block. If smaller (LSB) multipliers are used, shorter delays can be realized.

This section is intended to be used in conjunction with the section on switching characteristics in the Virtex-II Pro Platform FPGA Data Sheet and the Timing Analyzer (TRCE) report from Xilinx software. For specific timing parameter values, refer to the Virtex-II Pro Platform FPGA Data Sheet.

Timing Parameters

Worst-Case Propagation Delays

The TMULT timing parameter is reported by the Timing Analyzer software. These values correspond to the propagation delay through the multiplier to a specific output pin of the multiplier block. The shortest delay is to pin 0, the longest to pin 35. These parameters can be found in the table entitled “Multiplier Switching Characteristics” in Module 3 of the Virtex-II Pro Platform FPGA Data Sheet. The delay-to-pin ratio is essentially linear (see Figure 2-11). This implies that smaller multiply functions are faster than larger ones. This is true as long as the LSB inputs are used.
Figure 2-11: Pin-to-Delay Ratio Curve
Figure 2-12 illustrates the result (outputs) of a 4-bit x 4-bit unsigned multiply implemented in an embedded multiplier block.

Figure 2-12: Embedded Multiplier Block Timing Diagram

At time 0 the two 4-bit numbers to be multiplied become valid at the A[3..0], B[3..0] inputs to the embedded multiplier. The result appears on the output pins P[0..7] in a staggered fashion. First, P0 becomes valid at time $T_{MULT}(P0)$, followed by each subsequent output pin, until P7 becomes valid at time $T_{MULT}(P7)$. In this case, the delay for this multiply function should correspond to that of Pin 7. In other words, the result is not valid until all output pins become valid.
IOB Timing Models

The following three sections describe all timing parameters associated with the Virtex-II Pro/Virtex-II Pro X IOB. These three sections are:

- “IOB Input Timing Model and Parameters”
- “IOB Output Timing Model and Parameters”
- “IOB 3-State Timing Model and Parameters”

These sections are intended to be used in conjunction with the section on switching characteristics in the Data Sheet and the Timing Analyzer (TRCE) report from Xilinx software. For specific timing parameter values, refer to the Data Sheet.

A Note on I/O Standard Adjustments:

The "IOB Input and Output Switching Characteristics Standard Adjustments" tables in Module 3 of the Data Sheet are delay adders (+/-) to be added to all timing parameter values associated with the IOB and the Global Clock (see “Pin-to-Pin Timing Models,” page 55), if an I/O standard other than LVCMOS_25 12 mA is used.

All values specified in the Data Sheet for the parameters covered in this section are specified for LVCMOS_25 12 mA. If another I/O standard is used, these delays change. However, there are several exceptions. The following parameters associated with the pad going to high-impedance (3-State buffer OFF) should NOT be adjusted:

- TiOThz
- TiOTLPHZ
- TGTS
- TiOCKHZ
- TiOSRHZ
IOB Input Timing Model and Parameters

Figure 2-13 illustrates IOB inputs.

![IOB Input Timing Diagram](image)

**Figure 2-13: Virtex-II Pro/Virtex-II Pro X IOB Inputs**

**Timing Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Control Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_{IOPI}</td>
<td></td>
<td></td>
<td>Propagation delay from the pad to I output of the IOB with no delay adder.</td>
</tr>
<tr>
<td>T_{IOPID}</td>
<td></td>
<td></td>
<td>Propagation delay from the pad to I output of the IOB with the delay adder.</td>
</tr>
<tr>
<td>T_{IOPLI}</td>
<td></td>
<td></td>
<td>Propagation delay from the pad to IQ output of the IOB via transparent latch with no delay adder.</td>
</tr>
<tr>
<td>T_{IOPLID}</td>
<td></td>
<td></td>
<td>Propagation delay from the pad to IQ output of the IOB via transparent latch with the delay adder.</td>
</tr>
</tbody>
</table>
### IOB Timing Models

#### Table 2-5: IOB Input Timing Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Control Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Setup and Hold With Respect to Clock at IOB Input Register</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{xxCK}$</td>
<td>Setup time (before clock edge)</td>
<td></td>
<td>The following descriptions are for setup times only.</td>
</tr>
<tr>
<td>$T_{xxCKxx}$</td>
<td>Hold time (after clock edge)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{IOIck}/T_{IOIckp}$</td>
<td>ID input with NO delay</td>
<td></td>
<td>Time before the clock that the input signal from the pad must be stable at the ID input of the IOB Input Register, with no delay.</td>
</tr>
<tr>
<td>$T_{IoIckd}/T_{IoIckpd}$</td>
<td>ID input with delay</td>
<td></td>
<td>Time before the clock that the input signal from the pad must be stable at the ID input of the IOB Input Register, with delay.</td>
</tr>
<tr>
<td>$T_{IOIceck}/T_{IOIcece}$</td>
<td>ICE input</td>
<td></td>
<td>Time before the clock that the Clock Enable signal must be stable at the ICE input of the IOB Input Register.</td>
</tr>
<tr>
<td>$T_{IOSrcki}$</td>
<td>SR input (IFF, synchronous)</td>
<td></td>
<td>Time before the clock that the Set/Reset signal must be stable at the SR input of the IOB Input Register.</td>
</tr>
<tr>
<td><strong>Clock to Out</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{Iockiq}$</td>
<td>Clock (CLK) to (IQ) output</td>
<td></td>
<td>Time after the clock that the output data is stable at the IQ output of the IOB Input Register.</td>
</tr>
<tr>
<td><strong>Set/Reset Delays</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{Iosriq}$</td>
<td>SR Input to IQ (asynchronous)</td>
<td></td>
<td>Time after the Set/Reset signal of the IOB is toggled that the output of the IOB input register (IQ) reflects the signal.</td>
</tr>
<tr>
<td>$T_{Gsrq}$</td>
<td>GSR to output IQ</td>
<td></td>
<td>Time after the Global Set/Reset is toggled that the output of the IOB input register (IQ) reflects the set or reset.</td>
</tr>
</tbody>
</table>
Timing Characteristics

Figure 2-14 illustrates IOB input register timing.

Clock Event 1
- At time $T_{YOICECK}$ before Clock Event 1, the input clock enable signal becomes valid-high at the ICE input of the input register, enabling the input register for incoming data.
- At time $T_{YOIPICK}$ before Clock Event 1, the input signal becomes valid-high at the I input of the input register and is reflected on the IQ output of the input register at time $T_{YOICKIQ}$ after Clock Event 1.

Clock Event 4
- At time $T_{YOSRCKI}$ before Clock Event 4, the SR signal (configured as synchronous reset in this case) becomes valid-high resetting the input register and reflected at the IQ output of the IOB at time $T_{YOICKIQ}$ after Clock Event 4.
Timing Characteristics, DDR

**Figure 2-15** illustrates IOB DDR input register timing.

**Clock Event 1**
- At time $T_{\text{IOICECK}}$ before Clock Event 1 the input clock enable signal becomes valid-high at the ICE input of both DDR input registers, enabling them for incoming data. Since the ICE and I signals are common to both DDR registers, care must be taken to toggle these signals between the rising edges of ICLK1 and ICLK2 as well as meeting the register setup-time relative to both clocks.
- At time $T_{\text{IOIPICK}}$ before Clock Event 1 (rising edge of ICLK1) the input signal becomes valid-high at the I input of both registers and is reflected on the IQ1 output of input-register 1 at time $T_{\text{IOCKIQ}}$ after Clock Event 1.

**Clock Event 2**
At time $T_{\text{IOIPICK}}$ before Clock Event 2 (rising edge of ICLK2) the input signal becomes valid-low at the I input of both registers and is reflected on the IQ2 output of input-register 2 at time $T_{\text{IOCKIQ}}$ after Clock Event 2 (no change in this case).

**Clock Event 9**
At time $T_{\text{IOSRCK1}}$ before Clock Event 9 the SR signal (configured as synchronous reset in this case) becomes valid-high resetting input-register 1 (IQ1) at time $T_{\text{IOCKIQ}}$ after Clock Event 9, and input-register 2 (IQ2) at time $T_{\text{IOCKIQ}}$ after Clock Event 10.
IOB Output Timing Model and Parameters

Figure 2-16 illustrates IOB outputs.

![Virtex-II Pro/Virtex-II Pro X IOB Output Diagram](image)

Figure 2-16: Virtex-II Pro/Virtex-II Pro X IOB Output

Timing Parameters

Table 2-6: IOB Output Timing Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Control Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Propagation Delays</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{IOOP}$</td>
<td></td>
<td></td>
<td>Propagation delay from the O input of the IOB to the pad.</td>
</tr>
<tr>
<td>$T_{IOOLP}$</td>
<td></td>
<td></td>
<td>Propagation delay from the O input of the IOB to the pad via transparent latch.</td>
</tr>
<tr>
<td><strong>Setup and Hold With Respect to Clock at IOB Output Register</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$T_{OCCK}$ = Setup time (before clock edge)</td>
<td>O input</td>
<td></td>
<td>Time before the clock that data must be stable at the O input of the IOB Output Register.</td>
</tr>
<tr>
<td>$T_{OCCKX}$ = Hold time (after clock edge)</td>
<td></td>
<td></td>
<td>The following descriptions are for setup times only.</td>
</tr>
<tr>
<td>$T_{IOOCECK}/T_{IOCKO}$</td>
<td>OCE input</td>
<td></td>
<td>Time before the clock that the Clock Enable signal must be stable at the OCE input of the IOB Output Register.</td>
</tr>
</tbody>
</table>
Table 2-6: IOB Output Timing Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Control Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{IOSRCKO}$/$T_{IOCKOSR}$</td>
<td>SR input (OFF)</td>
<td></td>
<td>Time before the clock that the Set/Reset signal must be stable at the SR input of the IOB Output Register.</td>
</tr>
</tbody>
</table>

Clock to Out

| $T_{IOCKP}$ | Clock (CLK) to pad | Time after the clock that the output data is stable at the pad. |

Set/Reset Delays

| $T_{IOSRP}$ | SR Input to pad (asynchronous) | Time after the Set/Reset input of the IOB is toggled that the pad reflects the set or reset. |
| $T_{IOGSRQ}$ | GSR to pad | Time after the Global Set/Reset is toggled that the pad reflects the set or reset. |

Timing Characteristics

Figure 2-17 illustrates IOB output register timing.

Clock Event 1

- At time $T_{IOOCECK}$ before Clock Event 1, the output clock enable signal becomes valid-high at the OCE input of the output register, enabling the output register for incoming data.
- At time $T_{IOOCK}$ before Clock Event 1, the output signal becomes valid-high at the O input of the output register and is reflected on the pad at time $T_{IOCKP}$ after Clock Event 1.
Clock Event 4

At time $T_{IOSRCKO}$ before Clock Event 4, the SR signal (configured as synchronous reset in this case) becomes valid-high, resetting the output register and reflected on the pad at time $T_{IOCKP}$ after Clock Event 4.

Timing Characteristics, DDR

Figure 2-18 illustrates IOB DDR output register timing.

![Figure 2-18: IOB DDR Output Register Timing Characteristics](image)

Clock Event 1

- At time $T_{IOOCECK}$ before Clock Event 1, the output clock enable signal becomes valid-high at the OCE input of both of the DDR output registers, enabling them for incoming data. Since the OCE signal is common to both DDR registers, care must be taken to toggle this signal between the rising edges of OTCLK1 and OTCLK2 as well as meeting the register setup-time relative to both clocks.
- At time $T_{IOOCK}$ before Clock Event 1 (rising edge of OTCLK1), the output signal O1 becomes valid-high at the O1 input of output register 1 and is reflected on the pad at time $T_{IOCKP}$ after Clock Event 1.

Clock Event 2*

At time $T_{IOOCECK}$ before Clock Event 2 (rising edge of OTCLK2), the output signal O2 becomes valid-high at the O2 input of output register 2 and is reflected on the pad at time $T_{IOCKP}$ after Clock Event 2 (no change on the pad in this case).

Clock Event 9

At time $T_{IOSRCKO}$ before Clock Event 9, the SR signal (configured as synchronous reset in this case) becomes valid-high, resetting output-register 1 (reflected on the pad at time $T_{IOOCK}$ before Clock Event 9).
T_{IOCKP} after Clock Event 9) (no change in this case) and output-register 2 (reflected on the pad at time T_{IOCKP} after Clock Event 10) (no change in this case).

**IOB 3-State Timing Model and Parameters**

Figure 2-19 illustrates IOB 3-state timing

![Figure 2-19: Virtex-II Pro/Virtex-II Pro X IOB 3-State](image)

**Timing Parameters**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Control Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_{IOTHZ}</td>
<td>Time after T input of the IOB is toggled that the pad goes to high-impedance.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_{IOTON}</td>
<td>Time after the T input of the IOB is toggled that the pad goes from high-impedance to valid data.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>T_{IOTPLHZ}</td>
<td>Time after the T input of the IOB via transparent latch is toggled that the pad goes to high-impedance.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 2-7: IOB 3-State Timing Parameters (Continued)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Control Signal</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{IOTLPON}$</td>
<td>Time after the T</td>
<td></td>
<td>Time after the T input of the IOB via transparent latch is toggled that the pad goes from high-impedance to valid data.</td>
</tr>
<tr>
<td>$T_{GTS}$</td>
<td></td>
<td></td>
<td>Time after the Global 3-state signal is asserted that the pad goes to high-impedance.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Setup and Hold With Respect to Clock at IOB 3-State Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{xxCK}$ = Setup time (before clock edge)</td>
</tr>
<tr>
<td>$T_{xxCKxx}$ = Hold time (after clock edge)</td>
</tr>
<tr>
<td>The following descriptions are for setup times only.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{IOTCK}/T_{IOCKT}$</td>
<td>T input</td>
<td>Time before the clock that the signal must be stable at the T input of the IOB 3-state Register.</td>
</tr>
<tr>
<td>$T_{IOTCECK}/T_{IOCKTCE}$</td>
<td>TCE input</td>
<td>Time before the clock that the clock enable signal must be stable at the TCE input of the IOB 3-state Register.</td>
</tr>
<tr>
<td>$T_{IOSRCKT}/T_{IOCKTSR}$</td>
<td>SR input (TFF)</td>
<td>Time before the clock that the set/reset signal.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Clock to Out</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{IOCKHZ}$</td>
</tr>
<tr>
<td>$T_{IOCKON}$</td>
</tr>
<tr>
<td>Time after clock that the pad goes to high-impedance.</td>
</tr>
<tr>
<td>Time after clock that the pad goes from high-impedance to valid data.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Set/Reset Delays</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T_{IOSRHZ}$</td>
</tr>
<tr>
<td>Time after the SR signal is toggled that the pad goes to high-impedance.</td>
</tr>
<tr>
<td>$T_{IOSRON}$</td>
</tr>
<tr>
<td>Time after the SR signal is toggled that the pad goes from high-impedance to valid data.</td>
</tr>
</tbody>
</table>
Timing Characteristics

Figure 2-20 illustrates IOB 3-state register timing.

Clock Event 1
- At time $T_{\text{IOTCECK}}$ before Clock Event 1, the 3-state clock enable signal becomes valid-high at the TCE input of the 3-state register, enabling the 3-state register for incoming data.
- At time $T_{\text{IOTCK}}$ before Clock Event 1 the 3-state signal becomes valid-high at the T input of the 3-state register, returning the pad to high-impedance at time $T_{\text{IOCKHZ}}$ after Clock Event 1.

Clock Event 2
At time $T_{\text{IOSRCKT}}$ before Clock Event 2, the SR signal (configured as synchronous reset in this case) becomes valid-high, resetting the 3-state register and returning the pad to valid data at time $T_{\text{IOSRON}}$ after Clock Event 2.
Timing Characteristics, DDR

Figure 2-21 illustrates IOB DDR 3-state register timing.

Clock Event 1
At time $T_{IOTCECK}$ before Clock Event 1, the 3-state clock enable signal becomes valid-high at the TCE input of both of the DDR 3-state registers, enabling them for incoming data. Since the TCE signal is common to both DDR registers, care must be taken to toggle this signal between the rising edges of OTCLK1 and OTCLK2 as well as meeting the register setup-time relative to both clocks.

Clock Event 2
At time $T_{IOTCK}$ before Clock Event 2 (rising edge of OTCLK2), the 3-state signal T2 becomes valid-high at the T2 input of 3-state register 2, switching the pad to high-impedance at time $T_{IOCKHZ}$ after Clock Event 2.

Clock Event 3
At time $T_{IOTCK}$ before Clock Event 3 (rising edge of OTCLK1), the 3-state signal T1 becomes valid-high at the T1 input of 3-state register 1, keeping the pad at high-impedance for another half clock cycle (half the period of OTCLK1 or 2).

Clock Event 4
At time $T_{IOTCK}$ before Clock Event 4 (rising edge of OTCLK2), the 3-state signal T2 becomes valid-low at the T2 input of 3-state register 2, switching the pad to valid data at time $T_{IOCKON}$ after Clock Event 4. This is repeated for 3-state signal T1 at the following clock event (5) maintaining valid data on the pad until Clock Event 8.
Pin-to-Pin Timing Models

Clock Event 8
At time $T_{IOTCK}$ before Clock Event 8 (rising edge of OTCLK2), the 3-state signal $T_2$ becomes valid-high at the $T_2$ input of 3-state register 2, switching the pad to high-impedance at time $T_{IOCKHz}$ after Clock Event 8.

Clock Event 9
At time $T_{IOSRCKT}$ before Clock Event 9 (rising edge of OTCLK1), the SR signal (configured as synchronous reset in this case) becomes valid-high at the SR input of 3-state Register 1, returning the pad to valid data at time $T_{IOSRON}$ after Clock Event 9.

Pin-to-Pin Timing Models

This section explains the delays and timing parameters associated with the use of the Global Clock network and the DCM. These delays are true pin-to-pin delays relative to the Global Clock pin and an output or input pin with or without the DCM.

This section consists of two parts:

- “Global Clock Input to Output”
- “Global Clock Setup and Hold”

The former describes the delay from the Global Clock pin (with and without the DCM) to an output pin via an Output flip-flop. The latter describes the set-up time for an Input flip-flop from an input pin relative to the Global Clock pin (with and without the DCM).

The values reported in the switching characteristics section of the Virtex-II Pro Platform FPGA Data Sheet are for LVCMOS_25 12 mA I/O standards. For different I/O standards, adjust these values with those shown in the “IOB Switching Characteristics Standard Adjustments” tables.

This section is intended to be used in conjunction with the section on switching characteristics in the Virtex-II Pro Platform FPGA Data Sheet and the Timing Analyzer (TRCE) report from Xilinx software. For specific timing parameter values, refer to the Virtex-II Pro Platform FPGA Data Sheet.
Global Clock Input to Output

Figure 2-22 illustrates the paths associated with the timing parameters defined in this section. Note that they differ only in their use of the DCM.

Timing Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>T_{ICKOFDLL}</td>
<td>Time after the Global Clock (pin), using the DCM, that the output data from an IOB Output flip-flop is stable at the output pin.</td>
</tr>
<tr>
<td>T_{ICKOF}</td>
<td>Time after the Global Clock (pin), without the DCM, that the output data from an IOB Output flip-flop is stable at the output pin.</td>
</tr>
</tbody>
</table>
Timing Characteristics

The waveforms depicted in Figure 2-23 demonstrate the relation of the Global Clock pin, the output data, and the use of the timing parameters.

![Global Clock Input to Output Timing Characteristics](UG002_C3_015_101300)

Figure 2-23: Global Clock Input to Output Timing Characteristics
Global Clock Setup and Hold

Figure 2-24 illustrates the paths associated with the timing parameters defined in this section. Note, they differ only in their use of the DCM.

Timing Parameters

Setup and Hold for Input Registers Relative to the Global Clock (pin):

- $T_{PSDLL} / T_{PHDLL}$ - Time before the Global Clock (pin), with DCM, that the input signal must be stable at the D-input of the IOB input register.

- $T_{PSFD} / T_{PHFD}$ - Time before the Global Clock (pin), without DCM, that the input signal must be stable at the D-input of the IOB input register.

**Note:** $T_{PSFD} = \text{Setup time (before clock edge)}$ and $T_{PHFD} = \text{Hold time (after clock edge)}$. The previous descriptions are for setup times only.
Timing Characteristics

The waveforms depicted in Figure 2-25 demonstrate the relation of the Global Clock pin, the input data, and the use of the timing parameters.

**Figure 2-25:** Global Clock Setup and Hold Timing Characteristics
Digital Clock Manager Timing Model

This section describes the timing parameters associated with the Digital Clock Manager (DCM), which are reported in the Virtex-II Pro Platform FPGA Data Sheet. Note that these parameters are not used by the Timing Analyzer software in the production of timing reports; they are all measured values and are fully characterized in silicon. For specific timing parameter values, refer to the Virtex-II Pro Platform FPGA Data Sheet. This section discusses the following:

- **Operating Frequency Ranges:** The minimum and maximum frequencies supported by the DCM for all clock inputs and outputs.
- **Input Clock Tolerances:** Input clock period (pulse widths), jitter, and drift requirements for proper function of the DCM for all clock inputs.
- **Output Clock Precision:** Output clock period jitter, phase offsets, and duty cycle for all clock outputs of the DCM (worst case).
- **Miscellaneous Timing Parameters:** DCM lock times, Tap delay and shifting range.

For a detailed description of input clock tolerance, jitter, and phase offset see the waveforms at the end of this section.

Operating Frequency Ranges

Figure 2-26 illustrates the DCM functional block and corresponding timing parameters for all clock inputs and outputs.

![DCM Functional Block: Operating Frequency Ranges](image_url)
## Timing Parameters

### Table 2-9: Operating Frequency Range Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Low Frequency Mode</strong></td>
<td></td>
</tr>
<tr>
<td>CLKOUT_FREQ_1X_LF</td>
<td>The minimum and maximum frequency for the CLK0, CLK90, CLK180, CLK270 outputs of the DCM in low-frequency mode.</td>
</tr>
<tr>
<td>CLKOUT_FREQ_2X_LF</td>
<td>The minimum and maximum frequency for the CLK2X and CLK2X180 outputs of the DCM in low-frequency mode.</td>
</tr>
<tr>
<td>CLKOUT_FREQ_DV_LF</td>
<td>The minimum and maximum frequency for the CLKDV output of the DCM in low-frequency mode.</td>
</tr>
<tr>
<td>CLKOUT_FREQ_FX_LF</td>
<td>The minimum and maximum frequency for the CLKFX and CLKFX180 outputs of the DCM in low-frequency mode.</td>
</tr>
<tr>
<td>CLKIN_FREQ_DLL_LF(1)</td>
<td>The minimum and maximum frequency for the CLKIN input to the DCM in low-frequency mode when using the DLL outputs.</td>
</tr>
<tr>
<td>CLKIN_FREQ_FX_LF(2)</td>
<td>The minimum and maximum frequency for the CLKIN input to the DCM in low-frequency mode when using the FX outputs.</td>
</tr>
<tr>
<td>PSCLK_FREQ_LF</td>
<td>The minimum and maximum frequency for the PSCLK input to the DCM in low-frequency mode.</td>
</tr>
<tr>
<td><strong>High Frequency Mode</strong></td>
<td></td>
</tr>
<tr>
<td>CLKOUT_FREQ_1X_HF</td>
<td>The minimum and maximum frequency for the CLK0, CLK180 outputs of the DCM in high-frequency mode.</td>
</tr>
<tr>
<td>CLKOUT_FREQ_DV_HF</td>
<td>The minimum and maximum frequency for the CLKDV output of the DCM in high-frequency mode.</td>
</tr>
<tr>
<td>CLKOUT_FREQ_FX_HF</td>
<td>The minimum and maximum frequency for the CLKFX and CLKFX180 outputs of the DCM in high-frequency mode.</td>
</tr>
<tr>
<td>CLKIN_FREQ_DLL_HF</td>
<td>The minimum and maximum frequency for the CLKIN input to the DCM in high-frequency mode when using the DLL outputs.</td>
</tr>
<tr>
<td>CLKIN_FREQ_FX_HF</td>
<td>The minimum and maximum frequency for the CLKIN input to the DCM in high-frequency mode when using the FX outputs.</td>
</tr>
<tr>
<td>PSCLK_FREQ_HF</td>
<td>The minimum and maximum frequency for the PSCLK input to the DCM in high-frequency mode.</td>
</tr>
</tbody>
</table>

### Notes:
1. Delay-locked loop (DLL) outputs include: CLK0, CLK90, CLK180, CLK270, CLK2X, CLK2X180, and CLKDV.
2. FX outputs include: CLKFX and CLKFX180
# Input Clock Tolerances

## Timing Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSCLK_PULSE(1)</td>
<td>The minimum pulse width (HIGH and LOW) that the PSCLK input to the DCM can have over a range of frequencies.</td>
</tr>
<tr>
<td>CLKIN_PULSE</td>
<td>The minimum pulse width (HIGH and LOW) that the CLKIN input to the DCM can have over a range of frequencies. Also applies to PSCLK.</td>
</tr>
<tr>
<td>CLKFB_DELAY_VAR_EXT</td>
<td>The maximum allowed variation in delay (across environmental changes) of the feedback clock path when routed externally for board-level de-skew.</td>
</tr>
</tbody>
</table>

### Low Frequency Mode

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKIN_CYC_JITT_DLL_LF</td>
<td>The maximum cycle-to-cycle jitter the CLKIN input to the DCM can have when using the DLL outputs in low-frequency mode.</td>
</tr>
<tr>
<td>CLKIN_CYC_JITT_FX_LF</td>
<td>The maximum cycle-to-cycle jitter the CLKIN input to the DCM can have when using the FX outputs in low-frequency mode.</td>
</tr>
<tr>
<td>CLKIN_PER_JITT_DLL_LF</td>
<td>The maximum period jitter the CLKIN input to the DCM can have when using the DLL outputs in low-frequency mode.</td>
</tr>
<tr>
<td>CLKIN_PER_JITT_FX_LF</td>
<td>The maximum period jitter the CLKIN input to the DCM can have when using the FX outputs in low-frequency mode.</td>
</tr>
</tbody>
</table>

### High Frequency Mode

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKIN_CYC_JITT_DLL_HF</td>
<td>The maximum cycle-to-cycle jitter the CLKIN input to the DCM can have when using the DLL outputs in high-frequency mode.</td>
</tr>
<tr>
<td>CLKIN_CYC_JITT_FX_HF</td>
<td>The maximum cycle-to-cycle jitter the CLKIN input to the DCM can have when using the FX outputs in high-frequency mode.</td>
</tr>
<tr>
<td>CLKIN_PER_JITT_DLL_HF</td>
<td>The maximum period jitter the CLKIN input to the DCM can have when using the DLL outputs in high-frequency mode.</td>
</tr>
<tr>
<td>CLKIN_PER_JITT_FX_HF</td>
<td>The maximum period jitter the CLKIN input to the DCM can have when using the FX outputs in high-frequency mode.</td>
</tr>
</tbody>
</table>

### Notes:

1. The frequencies applicable to CLKIN_PULSE range from 1 to >400 MHz. These frequencies also apply to PSCLK_PULSE. Since PSCLK can be less than 1 MHz, the pulse width under this condition is specified for PSCLK only.
Output Clock Precision

Timing Parameters

Table 2-11: Output Clock Precision Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKOUT_PER_JITT_0</td>
<td>The maximum period jitter of the CLK0 output clock from the DCM (worst case).</td>
</tr>
<tr>
<td>CLKOUT_PER_JITT_90</td>
<td>The maximum period jitter of the CLK90 output clock from the DCM (worst case).</td>
</tr>
<tr>
<td>CLKOUT_PER_JITT_180</td>
<td>The maximum period jitter of the CLK180 output clock from the DCM (worst case).</td>
</tr>
<tr>
<td>CLKOUT_PER_JITT_270</td>
<td>The maximum period jitter of the CLK270 output clock from the DCM (worst case).</td>
</tr>
<tr>
<td>CLKOUT_PER_JITT_2X</td>
<td>The maximum period jitter of the CLK2X and CLK2X180 output clocks from the DCM (worst case).</td>
</tr>
<tr>
<td>CLKOUT_PER_JITT_DV1</td>
<td>The maximum period jitter of the CLKDV (integer division) output clock from the DCM (worst case).</td>
</tr>
<tr>
<td>CLKOUT_PER_JITT_DV2</td>
<td>The maximum period jitter of the CLKDV (non-integer division) output clock from the DCM (worst case).</td>
</tr>
<tr>
<td>CLKOUT_PER_JITT_FX</td>
<td>The maximum period jitter of the FX output clocks from the DCM (worst case).</td>
</tr>
<tr>
<td>CLKIN_CLKFB_PHASE</td>
<td>Maximum phase offset between the CLKIN and CLKFB inputs to the DCM.</td>
</tr>
<tr>
<td>CLKOUT_PHASE</td>
<td>Maximum phase offset between any DCM clock outputs.</td>
</tr>
<tr>
<td>CLKOUT_DUTY_CYCLE_DLL</td>
<td>The duty-cycle precision for all DLL outputs.</td>
</tr>
<tr>
<td>CLKOUT_DUTY_CYCLE_FX</td>
<td>The duty-cycle precision for the FX outputs.</td>
</tr>
</tbody>
</table>

Miscellaneous DCM Timing Parameters

Table 2-12: Miscellaneous DCM Timing Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LOCK_DLL</td>
<td>Time required for DCM to lock over a range of clock frequencies when using the DLL outputs.</td>
</tr>
<tr>
<td>LOCK_FX</td>
<td>Time required for DCM to lock when using the FX outputs.</td>
</tr>
<tr>
<td>LOCK_DLL_FINE_SHIFT</td>
<td>Additional lock time when performing fine phase shifting.</td>
</tr>
<tr>
<td>FINE_SHIFT_RANGE</td>
<td>Absolute range for fine phase shifting.</td>
</tr>
<tr>
<td>DCM_TAP</td>
<td>Resolution of delay line.</td>
</tr>
</tbody>
</table>
The waveforms in Figure 2-27 demonstrate the relationship between clock tolerance, jitter, and phase.

**Period Tolerance**: the allowed input clock period change in nanoseconds.

**Output Jitter**: the difference between an ideal reference clock edge and the actual design.

**Phase Offset and Maximum Phase Difference**:

Output jitter is period jitter measured on the DLL output clocks, excluding input clock jitter.

Phase offset between CLKIN and CLKFB is the worst-case fixed time difference between rising edges of CLKIN and CLKFB, excluding output jitter and input clock jitter.

Phase offset between clock outputs on the DLL is the worst-case fixed time difference between rising edges of any two DLL outputs, excluding output jitter and input clock jitter.

Maximum phase difference between CLKIN and CLKFB is the sum of output jitter and phase offset between CLKIN and CLKFB, or the greatest difference between CLKIN and CLKFB rising edges due to DLL alone (excluding input clock jitter).

Maximum phase difference between clock outputs on the DLL is the sum of output jitter and phase offset between any DLL clock outputs, or the greatest difference between any two DLL output rising edges due to DLL alone (excluding input clock jitter).
Additional Timing Models in Other Publications: PPC405 Processor Block and RocketIO/RocketIO X Transceivers

Due to their length and complexity, timing models for the IBM PPC405 Processor Block and the RocketIO/RocketIO X Multi-Gigabit Transceivers are not included in this User Guide. They can be found in the publications described below, which cover in detail the operation and implementation of these Virtex-II Pro/Virtex-II Pro X components:

IBM PPC405 Processor Block Timing Models

Timing models for the IBM PPC405 Processor Block can be found in PowerPC 405 Processor Block Reference Guide. The most current version of this guide is available on the website at www.xilinx.com/xlnx/xweb/xil_publications_index.jsp?category=User+Guides. It is intended to be used in conjunction with Module 3 of the Virtex-II Pro/Virtex-II Pro X Data Sheet and the Timing Analyzer (TRCE) report from Xilinx software. For specific timing parameter values, refer to the data sheet.

RocketIO Transceiver Timing Models

Timing models for the RocketIO Transceiver can be found in the RocketIO Transceiver User Guide, UG024. The most current version of this volume is available on the Xilinx website at www.xilinx.com/xlnx/xweb/xil_publications_index.jsp?category=User+Guides.

RocketIO X Transceiver Timing Models

Timing models for the RocketIO X Transceiver can be found in the RocketIO X Transceiver User Guide, UG035. The most current version of this volume is available on the Xilinx website at www.xilinx.com/xlnx/xweb/xil_publications_index.jsp?category=User+Guides.
Chapter 3

Design Considerations

Summary

This chapter covers the following topics:

- RocketIO and RocketIO X Transceivers
- Processor Block
- Global Clock Networks
- Digital Clock Managers (DCMs)
- Block SelectRAM™ Memory
- Distributed SelectRAM Memory
- Look-Up Tables as Shift Registers (SRLs)
- Large Multiplexers
- Sum of Products (SOP) Logic
- Embedded Multipliers
- Single-Ended SelectIO-Ultra Resources
- Digitally Controlled Impedance (DCI)
- Double-Data-Rate (DDR) I/O
- LVDS I/O
- LVPECL I/O
- Bitstream Encryption
- CORE Generator System

Introduction

This chapter describes how to take advantage of the many special features of Virtex-II Pro / Virtex-II Pro X architecture to achieve maximum density and performance. In many cases, the functions described can be automatically generated using the Xilinx CORE Generator™ tool. This is noted throughout the chapter, in the following sections specifically:

- Block SelectRAM™ Memory
- Distributed SelectRAM Memory
- Look-Up Tables as Shift Registers (SRLs)
- Large Multiplexers
- Embedded Multipliers
RocketIO and RocketIO X Transceivers

Virtex-II Pro / Virtex-II Pro X devices provide up to twenty multi-gigabit transceivers capable of various high-speed serial standards such as Gigabit Ethernet, FiberChannel, Infiniband, and XAUI. In addition, the channel-bonding feature aggregates multiple channels allowing for even higher data transfer rate.

For the most up-to-date information on RocketIO and RocketIO X transceiver features, design examples, power considerations, and board layout suggestions, refer to the most current revision of the RocketIO Transceiver User Guide or the RocketIO X Transceiver User Guide the on the Xilinx website at www.xilinx.com/xlnx/xweb/xil_publications_index.jsp?category=User+Guides.

Processor Block

Two additional user manuals detail the hardware and software design aspects of the processor block. The PowerPC 405 Processor Block Reference Guide provides information on input/output signals, timing relationships between signals, and the mechanisms software can use to control the interface operation. The PowerPC Processor User Manual serves as a stand-alone reference for application and system programmers of the PPC405 processor core.

For the most up-to-date information on the PPC405 processor core, refer to the current revisions of these manuals on the Xilinx website at www.xilinx.com/xlnx/xweb/xil_publications_index.jsp?category=User+Guides.

Global Clock Networks

Introduction

Virtex-II Pro / Virtex-II Pro X devices support very high frequency designs and thus require low-skew advanced clock distribution. With device density up to 10 million system gates, numerous global clocks are necessary in most designs. Therefore, to provide a uniform and portable solution (soft-IP), all Virtex-II Pro / Virtex-II Pro X devices from XC2VP2 to XC2VP100 have 16 global clock buffers and support 16 global clock domains. Up to eight of these clocks can be used in any quadrant of the device by the synchronous logic elements (that is, registers, 18Kb block RAM, pipeline multipliers) and the IOBs. The software tools place and route these global clocks automatically.

If the design uses between 8 and 16 clocks, it must be partitioned into quadrants, with up to 8 clocks per quadrant. If more than 16 clocks are required, the backbone (24 horizontal and vertical long lines routing resources) can be used as additional clock network.

In addition to clock distribution, the 16 clock buffers are also “glitch-free” synchronous 2:1 multiplexers. These multiplexers are capable of switching between two asynchronous (or synchronous) clocks at any time. No particular phase relations between the two clocks are needed. The clock multiplexers can also be configured as a global clock buffer with a clock enable. The clock can be stopped High or Low at the clock buffer output.
Global Clock Networks

Clock Distribution Resources

The various resources available to manage and distribute the clocks include:

- Sixteen clock pads that can be used as regular user I/Os if not used as clock inputs. The sixteen clock pads can be configured for any I/O standard, including differential standards (for example, LVDS).
- Sixteen “IBUF” elements that represent the clock inputs in VHDL or Verilog designs.
- Eight “IBUFGDS” elements (that is, attributes LVPECL_25, LVDS_25, LDT_25, or ULVDS_25) that represent the differential clock input pairs in a VHDL or Verilog design. Each IBUFGDS replaces two IBUF elements.
- Four to twelve Digital Clock Managers (DCMs), depending on the device size, to de-skew and generate the clocks. For more information on DCMs, see “Digital Clock Managers (DCMs),” page 89.
- Sixteen “BUFGMUX” elements that can consist of up to sixteen global clock buffers (BUFG), global clock buffers with a clock enable (BUFGCE), or global clock multiplexers (BUFGMUX).

Figure 3-1 illustrates placement of these clock resources in Virtex-II Pro / Virtex-II Pro X devices (the XC2VP20/XC2VPX20 through the XC2VP70/XC2VPX70) that have eight DCMs.

*Figure 3-1: Clock Resources in Virtex-II Pro/Virtex-II Pro X Devices*
The simple scheme to distribute an external clock in the device is to implement a clock pad with an IBUFG input buffer connected to a BUFG global buffer, as shown in Figure 3-2 and Figure 3-3. The primary (GCLKP) and secondary (GCLKS) clock pads have no relationship with the P-side and N-side of differential clock inputs. In banks 0 and 1, the GCLKP corresponds to the N-side, and the GCLKS corresponds to the P-side of a differential clock input. In banks 4 and 5, this correspondence is reversed.

Major synthesis tools automatically infer the IBUFG and BUFG when the corresponding input signal is used as a clock in the VHDL or Verilog code.

A high frequency or adapted (frequency, phase, and so forth) clock distribution with low skew is implemented by using a DCM between the output of the IBUFG and the input of the BUFG, as shown in Figure 3-4. “Digital Clock Managers (DCMs),” page 89 provides details about DCMs and their use.
Clock distribution from internal sources is also possible with a BUFG only or with a DCM, as shown in Figure 3-5.

Global Clock Inputs

The clock buffer inputs are fed either by one of the 16 clock pads (refer to the Virtex-II Pro and Virtex-II Pro X Platform FPGA Data Sheet), by the outputs of the DCM, or by local interconnect. Each clock buffer can be a synchronous “glitch-free” 2:1 multiplexer with two clock inputs and one select input. Internal logic (or alternatively a regular IOB) can feed the clock inputs. Any internal or external signal can drive the select input or clock enable input.

The possible inputs driving a global clock buffer or multiplexer are summarized in Table 3-1.
Table 3-1: Inputs Driving Global Clock Buffers or DCMs

<table>
<thead>
<tr>
<th>Source</th>
<th>BUFG(I) or BUFGCE(I)</th>
<th>BUFGCE (CE)</th>
<th>BUFGMUX (I0 or I1)</th>
<th>BUFGMUX (S)</th>
<th>DCM (CLKIN)</th>
</tr>
</thead>
<tbody>
<tr>
<td>External Clock via IBUFG(O)</td>
<td>Dedicated in same quadrant(1)</td>
<td>NA</td>
<td>Dedicated in same quadrant(1)</td>
<td>NA</td>
<td>Same edge</td>
</tr>
<tr>
<td>DCM Clock Outputs</td>
<td>Same edge (top or bottom)(2)</td>
<td>NA</td>
<td>Same edge (top or bottom)(2)</td>
<td>NA</td>
<td>General interconnect(3)</td>
</tr>
<tr>
<td>Internal Logic</td>
<td>General interconnect</td>
<td>General interconnect</td>
<td>General interconnect</td>
<td>General interconnect</td>
<td>General interconnect(3)</td>
</tr>
<tr>
<td>User I/O Pad via IBUF(O) (not IBUFG)</td>
<td>General interconnect</td>
<td>General interconnect</td>
<td>General interconnect</td>
<td>General interconnect</td>
<td>General interconnect(3)</td>
</tr>
<tr>
<td>BUFG(O)</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>Global clock net</td>
</tr>
<tr>
<td>BUFGMUX(O)</td>
<td>NA</td>
<td>NA</td>
<td>General interconnect</td>
<td>NA</td>
<td>Global clock net</td>
</tr>
</tbody>
</table>

Notes:
1. Not all IBUFGs in the quadrant have a dedicated connection to a specific BUFG. Others would require general interconnect to be hooked up.
2. Same edge (top or bottom) enables use of dedicated routing resources.
3. Pad to DCM input skew is not compensated.

All BUFG (BUFGCE, BUFGMUX) outputs are available at the quadrant boundaries.
The output of the global clock buffer can be routed to non-clock pins.

Primary and Secondary Global Multiplexers

Each global clock buffer is a self-synchronizing circuit called a clock multiplexer.
The 16 global clock buffers or multiplexers are divided as follows:
- Eight primary clock multiplexers
- Eight secondary clock multiplexers

No hardware difference exists between a primary and a secondary clock multiplexer.
However, some restrictions apply to primary/secondary multiplexers, because they share input connections, as well as access to a quadrant.

Each Virtex-II Pro / Virtex-II Pro X device is divided into four quadrants: North-West, South-West, North-East, and South-East. Each quadrant has two primary and two secondary clock multiplexers. The clock multiplexers are indexed 0 to 7, with one primary and one secondary for each index, alternating on the top and on the bottom (i.e., clock multiplexer “0P” at the bottom is facing clock multiplexer “0S” at the top).

In each device, the eight top/bottom clock multiplexers are divided into four primary and four secondary, indexed 0 to 7, as shown in Figure 3-6.
Global Clock Networks

Primary/Secondary: Rule 1

Considering two “facing” clock multiplexers (BUFG#P and BUFG#S), one or the other of these clock outputs can enter any quadrant of the chip to drive a clock within that quadrant, as shown in Figure 3-7. Note that the clock multiplexers “xP” and “xS” compete for quadrant access. For example, BUFG0P output cannot be used in the same quadrant as BUFG0S.

Figure 3-6: Primary and Secondary Clock Multiplexer Locations

Figure 3-7: Facing BUFG#P and BUFG#S Connections
Primary/Secondary: Rule 2

In a BUFGCE or BUFGMUX configuration, shared inputs have to be considered. Any two adjacent clock multiplexers share two inputs, as shown in Figure 3-8. The clock multiplexer “1P” and “0S” have common I0/I1 and I1/I0 inputs.

**Figure 3-8: Clock Multiplexer Pair Sharing Clock Multiplexer Inputs**

Table 3-2 lists the clock multiplexer pairs in any Virtex-II Pro / Virtex-II Pro X device. The primary multiplexer inputs I1/I0 are common with the corresponding secondary multiplexer inputs I0/I1 (i.e., Primary I1 input is common with secondary I0 input, and primary I0 input is common with secondary I1 input).

**Table 3-2: Top Clock Multiplexer Pairs**

<table>
<thead>
<tr>
<th>Primary I1/I0</th>
<th>1P</th>
<th>3P</th>
<th>5P</th>
<th>7P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secondary I0/I1</td>
<td>0S</td>
<td>2S</td>
<td>4S</td>
<td>6S</td>
</tr>
</tbody>
</table>

**Table 3-3: Bottom Clock Multiplexer Pairs**

<table>
<thead>
<tr>
<th>Primary I1/I0</th>
<th>0P</th>
<th>2P</th>
<th>4P</th>
<th>6P</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secondary I0/I1</td>
<td>1S</td>
<td>3S</td>
<td>5S</td>
<td>7S</td>
</tr>
</tbody>
</table>

Primary/Secondary Usage

For up to eight global clocks, it is safe to use the eight primary global multiplexers (1P, 3P, 5P, 7P on the top and 0P, 2P, 4P, 6P on the bottom). Because of the shared inputs, a maximum of eight independent global clock multiplexers can be used in a design, as shown in Figure 3-9.
Global Clock Networks

DCM Clocks

The four clock pins (IBUFG) in a quadrant can feed all DCMs on the same edge of the device. The clock-to-out and setup times are identical for all DCMs. Up to four clock outputs per DCM can be used to drive any clock multiplexer on the same edge (top or bottom), as shown in Figure 3-10.

BUFG Exclusivity

Each DCM has a restriction on the number of BUFGs it can drive on its (top or bottom) edge. Pairs of buffers with shared dedicated routing resources exist such that only one buffer from each dedicated pair can be driven by a single DCM. The exclusive pairs for each edge are: 0:4, 1:5, 2:6, and 3:7.
Chapter 3: Design Considerations

Clock Output

The clock distribution is based on eight clock trees per quadrant. Each clock multiplexer output is driving one global clock net. The Virtex-II Pro / Virtex-II Pro X device has eight dedicated low-skew clock nets. The device is divided into four quadrants (NW, NE, SW and SE) with eight global clocks available per quadrant.

Eight clock buffers are in the middle of the top edge and eight are in the middle of the bottom edge. Any of these 16 clock buffer outputs can be used in any quadrant, up to a maximum of eight clocks per quadrant, as illustrated in Figure 3-11, provided there is not a primary vs. secondary conflict.

Figure 3-10: DCM Clocks

Figure 3-11: Clock Buffer Outputs per Quadrant
Designs with more than eight clocks must be floorplanned manually or automatically, distributing the clocks in each quadrant. As an example, a design with 16 clocks can be floorplanned as shown in Figure 3-12.

Figure 3-12: 16-Clock Floorplan
The clock nets and clock buffers in this example are associated as shown in Table 3-4.

### Table 3-4: Clock Net Association With Clock Buffers

<table>
<thead>
<tr>
<th>Clock Net (top edge)</th>
<th>CLK_A</th>
<th>CLK_B</th>
<th>CLK_C</th>
<th>CLK_D</th>
<th>CLK_E</th>
<th>CLK_F</th>
<th>CLK_G</th>
<th>CLK_H</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFG</td>
<td>7P</td>
<td>6S</td>
<td>5P</td>
<td>4S</td>
<td>3P</td>
<td>2S</td>
<td>1P</td>
<td>0S</td>
</tr>
<tr>
<td>Clock Net (bottom edge)</td>
<td>CLK_I</td>
<td>CLK_J</td>
<td>CLK_K</td>
<td>CLK_L</td>
<td>CLK_M</td>
<td>CLK_N</td>
<td>CLK_O</td>
<td>CLK_P</td>
</tr>
<tr>
<td>BUFG</td>
<td>7S</td>
<td>6P</td>
<td>5S</td>
<td>4P</td>
<td>3S</td>
<td>2P</td>
<td>1S</td>
<td>0P</td>
</tr>
<tr>
<td>Quadrant NW</td>
<td>CLK_A</td>
<td>CLK_B</td>
<td>CLK_C</td>
<td>–</td>
<td>–</td>
<td>CLK_F</td>
<td>–</td>
<td>CLK_P</td>
</tr>
<tr>
<td>Quadrant SW</td>
<td>CLK_A</td>
<td>–</td>
<td>CLK_C</td>
<td>CLK_L</td>
<td>CLK_M</td>
<td>CLK_N</td>
<td>CLK_G</td>
<td>CLK_H</td>
</tr>
<tr>
<td>Quadrant NE</td>
<td>CLK_I</td>
<td>CLK_B</td>
<td>CLK_K</td>
<td>CLK_D</td>
<td>CLK_E</td>
<td>CLK_N</td>
<td>CKL_O</td>
<td>CLK_H</td>
</tr>
<tr>
<td>Quadrant SE</td>
<td>CLK_A</td>
<td>CLK_J</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>CLK_P</td>
</tr>
</tbody>
</table>

 CLK_A is used in three quadrants, and the other clocks are used in one or two quadrants, regardless of the position of the clock buffers (multiplexers), as long as they are not competing to access the same quadrant. That is, CLK_A (BUFG7P) cannot be used in the same quadrant with CLK_I (BUFG7S). Refer to “Primary/Secondary: Rule 1,” page 73.) In other words, two buffers with the same index (0 to 7) cannot be used in the same quadrant.

Each register, block RAM, registered multiplier, or DDR register (IOB) can be connected to any of the eight clock nets available in a particular quadrant.

Note that if a global clock (primary buffer) is used in four quadrants, the corresponding secondary buffer is not available.

### Power Consumption

Clock trees have been designed for low skew and low-power operation. Any unused branch is disconnected, as shown in Figure 3-13.
Also available to reduce overall power consumption are the BUFGCE feature, for dynamically driving a clock tree only when the corresponding module is used, and the BUFGMUX feature, for switching from a high-frequency clock to a low-frequency clock.

The frequency synthesizer capability of the DCM can generate the low (or high) frequency clock from a single source clock, as illustrated in Figure 3-14. (See “Digital Clock Managers (DCMs),” page 89.)

Library Primitives and Submodules

The primitives in Table 3-5 are available with the input, output, and control pins listed.

**Table 3-5: ClockPrimitives**

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Input</th>
<th>Output</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBUFG</td>
<td>I</td>
<td>O</td>
<td>–</td>
</tr>
<tr>
<td>IBUFGDS</td>
<td>I, IB</td>
<td>O</td>
<td>–</td>
</tr>
<tr>
<td>BUFG</td>
<td>I</td>
<td>O</td>
<td>–</td>
</tr>
<tr>
<td>BUFGMUX</td>
<td>I0, I1</td>
<td>O</td>
<td>S</td>
</tr>
<tr>
<td>BUFGMUX_1</td>
<td>I0, I1</td>
<td>O</td>
<td>S</td>
</tr>
</tbody>
</table>

Refer to “Single-Ended SelectIO-Ultra Resources,” page 185 for a list of the attributes available for IBUFG and Refer to “LVDS I/O,” page 256 for a list of the attributes available for IBUFGDS.

The submodules in Table 3-6 are available with the input, output, and control pins listed.

**Table 3-6: Clock Submodules**

<table>
<thead>
<tr>
<th>Submodule</th>
<th>Input</th>
<th>Output</th>
<th>Control</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFGCE</td>
<td>I</td>
<td>O</td>
<td>CE</td>
</tr>
<tr>
<td>BUFGCE_1</td>
<td>I</td>
<td>O</td>
<td>CE</td>
</tr>
</tbody>
</table>
Chapter 3: Design Considerations

Primitive Functions

IBUFG
IBUFG is an input clock buffer with one clock input and one clock output.

IBUFGDS
IBUFGDS is a differential input clock buffer with two clock inputs (positive and negative polarity) and one clock output.

BUFG
All Virtex-II Pro/Virtex-II Pro X devices have 16 global clock buffers (each of which can be used as BUFG, BUFGMUX, or BUFGCE).

BUFG is a global clock buffer with one clock input and one clock output, driving a low-skew clock distribution network. The output follows the input, as shown in Figure 3-15.

BUFGMUX and BUFGMUX_1
BUFGMUX (see Figure 3-16) can switch between two unrelated, even asynchronous clocks. Basically, a Low on S selects the I₀ input, a High on S selects the I₁ input. Switching from one clock to the other is done in such a way that the output High and Low time is never shorter than the shortest High or Low time of either input clock. As long as the presently selected clock is High, any level change of S has no effect. (For BUFGMUX_1, as long as the presently selected clock is Low, any level change of S has no effect.)

BUFGMUX is the preferred circuit for rising edge clocks, while BUFGMUX_1 is preferred for falling edge clocks.

Operation of the BUFGMUX Circuit
If the presently selected clock is Low while S changes, or if it goes Low after S has changed, the output is kept Low until the other ("to-be-selected") clock has made a transition from High to Low. At that instant, the new clock starts driving the output.

The two clock inputs can be asynchronous with regard to each other, and the S input can change at any time, except for a short setup time prior to the rising edge of the presently
selected clock; that is, prior to the rising edge of the BUFGMUX output O. Violating this setup time requirement can result in an undefined runt pulse output.

Figure 3-17 shows a switch-over from CLK0 to CLK1.

Operation of the BUFGMUX_1 Circuit

If the presently selected clock is High while S changes, or if it goes High after S has changed, the output is kept High until the other ("to-be-selected") clock has made a transition from Low to High. At that instant, the new clock starts driving the output.

The two clock inputs can be asynchronous with regard to each other, and the S input can change at any time, except for a short setup time prior to the falling edge of the presently selected clock; that is, prior to the falling edge of the BUFGMUX output O. Violating this setup time requirement can result in an undefined runt pulse output.

Figure 3-18 shows a switch-over from CLK0 to CLK1.
The current clock is CLK0.

S is activated High.

If CLK0 is currently Low, the multiplexer waits for CLK0 to go High.

Once CLK0 is High, the multiplexer output stays High until CLK1 transitions Low to High.

When CLK1 transitions from Low to High, the output switches to CLK1.

No glitches or short pulses can appear on the output.

Submodules

BUFGCE and BUFGCE_1

BUFGCE and BUFGCE_1 are submodules based on BUFGMUX and BUFGMUX_1, respectively. BUFGCE and BUFGCE_1 are global clock buffers incorporating a smart enable function that avoids output glitches or runt pulses. The select signal must meet the setup time for the clock.

BUFGCE is the preferred circuit for clocking on the rising edge, while BUFGCE_1 is preferred when clocking on the falling edge.

Operation of the BUFGCE Circuit

If the CE input (see Figure 3-19) is active (High) prior to the incoming rising clock edge, this Low-to-High-to-Low clock pulse passes through the clock buffer. Any level change of CE during the incoming clock High time has no effect.

If the CE input is inactive (Low) prior to the incoming rising clock edge, the following clock pulse does not pass through the clock buffer, and the output stays Low. Any level change of CE during the incoming clock High time has no effect. CE must not change during a short setup window just prior to the rising clock edge on the BUFGCE_1 input I. Violating this setup time requirement can result in an undefined runt pulse output.

This means the output stays Low when the clock is disabled, but it completes the clock-High pulse when the clock is being disabled, as shown in Figure 3-20.

![Figure 3-19: Virtex-II Pro/Virtex-II Pro X BUFGCE or BUFGCE_1 Function](DS031_62_101200)

![Figure 3-20: BUFGCE Waveforms](UG002_C2_100_120100)
Global Clock Networks

Operation of the BUFGCE_1 Circuit

If the CE input is active (High) prior to the incoming falling clock edge, this High-to-Low-to-High clock pulse passes through the clock buffer. Any level change of CE during the incoming clock Low time has no effect.

If the CE input is inactive (Low) prior to the incoming falling clock edge, the following clock pulse does not pass through the clock buffer, and the output stays High. Any level change of CE during the incoming clock Low time has no effect. CE must not change during a short setup window just prior to the falling clock edge on the BUFGCE input I. Violating this setup time requirement can result in an undefined runt pulse output.

This means the output stays High when the clock is disabled, but it completes the clock-Low pulse when the clock is being disabled, as shown in Figure 3-21.

![Figure 3-21: BUFGCE_1 Waveforms](image)

When BUFGCE (or BUFGCE_1) is used with DCM outputs, a second BUFG can be used for clock feedback. Buffer sharing the inputs with BUFGCE is the preferred solution.

Summary

Table 3-7 shows the maximum resources available per Virtex-II Pro / Virtex-II Pro X device.

<table>
<thead>
<tr>
<th>Resource</th>
<th>Maximum Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single-ended IBUFG (pads)</td>
<td>16</td>
</tr>
<tr>
<td>Differential IBUFGDS (pairs)</td>
<td>8</td>
</tr>
<tr>
<td>BUFG (Global Clock Buffer)</td>
<td>8</td>
</tr>
<tr>
<td>BUFGCE (or BUFGCE_1)</td>
<td>16</td>
</tr>
<tr>
<td>BUFGMUX (or BUFGMUX_1)</td>
<td>8</td>
</tr>
</tbody>
</table>

Characteristics

The following are characteristics of global clocks in Virtex-II Pro / Virtex-II Pro X devices:

- Low-skew clock distribution.
- Synchronous “glitch-free” multiplexer that avoids runt pulses. Switching between two asynchronous clock sources is usually considered unsafe, but it is safe with the Virtex-II Pro / Virtex-II Pro X global clock multiplexer.
- Any level change on S must meet a setup time requirement with respect to the signal on the output O (rising edge for BUFGMUX, falling edge for BUFGMUX_1). Any level change on CE must meet a setup time requirement with respect to the signal on the Input I (rising edge for BUFGCE, falling edge for BUFGCE_1).
- Two BUFGMUX (or BUFGMUX_1) resources can be cascaded to create a 3 to 1 clock multiplexer.
Location Constraints

BUFGMUX and BUFGMUX_1 (primitives) and IBUFG (IBUFGDS) instances can have LOC properties attached to them to constrain placement. The LOC properties use the following form to constrain a clock net:

```plaintext
INST "clock_buffer_instance_name" LOC="BUFGMUX#P/S";
```

Each clock pad (or IBUFG) has a direct connection with a specific global clock multiplexer (input I0). A placement that does not conform to this rule causes the software to send a warning.

If the clock pad (or IBUFG) has LOC properties attached, the DCM allows place and route software maximum flexibility, as compared to a direct connection to the global clock buffer (BUFG).

Secondary Clock Network

If more clocks are required, the 24 horizontal and vertical long lines in Virtex-II Pro / Virtex-II Pro X devices can be used to route additional clock nets. Skew is minimized by the place and route software, if the USELOWSKEWLINES constraint is attached to the net.

VHDL and Verilog Instantiation

VHDL and Verilog instantiation templates are available in the Libraries Guide for all submodules.

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

The following are templates for primitives:

- BUFGMUX_INST
- BUFGMUX_1_INST

The following are templates for submodules:

- BUFGCE_SUBM
- BUFGCE_1_SUBM

As examples, the BUFGMUX_INST.vhd, BUFGMUX_1_INST.vhd, BUFGCE_SUBM.vhd, and BUFGCE_1_SUBM.vhd VHDL templates are shown. In addition, the BUFGMUX_INST.v, BUFGMUX_1_INST.v, BUFGCE_1_SUBM.v, and BUFGCE_SUBM.v Verilog templates are shown.

VHDL Template

```vhdl
-- Module: BUFGMUX_INST
-- Description: VHDL instantiation template
-- Global Clock Multiplexer (Switch Low)
-- Device: Virtex-II Pro Family
---------------------------------------------------------------------
-- Component Declarations:
--
component BUFGMUX
port (I0 : in std_logic;
      I1 : in std_logic;
```
Global Clock Networks

S : in std_logic;
O : out std_logic
);
end component;
--
-- Architecture section:
--
-- Global Clock Buffer Instantiation
U_BUFGMUX: BUFGMUX
  port map (
    I0 => , -- insert clock input used when select (S) is Low
    I1 => , -- insert clock input used when select (S) is High
    S => , -- insert Mux-Select input
    O => -- insert clock output
  );
--
---------------------------------------------------------------------
-- Module: BUFGMUX_1_INST
-- Description: VHDL instantiation template
-- Global Clock Multiplexer (Switch High)
--
-- Device: Virtex-II Pro Family
---------------------------------------------------------------------
-- Component Declarations:
component BUFGMUX_1
  port (   
    I0 : in std_logic;
    I1 : in std_logic;
    S : in std_logic;
    O : out std_logic
  );
end component;
--
-- Architecture section:
--
-- Global Clock Buffer Instantiation
U_BUFGMUX_1: BUFGMUX_1
  port map (
    I0 => , -- insert clock input used when select (S) is Low
    I1 => , -- insert clock input used when select (S) is High
    S => , -- insert Mux-Select input
    O => -- insert clock output
  );
--
---------------------------------------------------------------------
-- Module: BUFGCE_SUBM
-- Description: VHDL instantiation template
-- Global Clock Buffer with Clock Enable:
-- Input Clock Buffer to BUFGMUX - Clock disabled = Low
-- Device: Virtex-II Pro Family
---------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
--
-- pragma translate_off
library UNISIM;
use UNISIM.VCOMPONENTS.ALL;
-- pragma translate_on
--
entity BUFGCE_SUBM is  
  port (  
    I: in std_logic;  
    CE: in std_logic;  
    O: out std_logic  
  );  
end BUFGCE_SUBM;  
--  
architecture BUFGCE_SUBM_arch of BUFGCE_SUBM is  
--  
-- Component Declarations:  
component BUFGMUX  
  port (  
    I0   : in std_logic;  
    I1   : in std_logic;  
    S    : in std_logic;  
    O    : out std_logic  
  );  
end component;  
--  
-- signal declarations  
signal GND : std_logic;  
signal CE_B : std_logic;  
--  
begin  
  GND <= '0';  
--  
  CE_B <= not CE;  
--  
-- Global Clock Buffer Instantiation  
U_BUFGMUX: BUFGMUX  
  port map (  
    I0     => I,  
    I1     => GND,  
    S      => CE_B,  
    O      => O  
  );  
--  
end BUFGCE_SUBM_arch;  

---------------------------------------------------------------------  
-- Module: BUFGCE_1_SUBM  
-- Description: VHDL instantiation template  
-- Global Clock Buffer with Clock Enable:  
-- Input Clock Buffer to BUFGMUX_1 - Clock disabled = High  
-- Device: Virtex-II Pro Family  
---------------------------------------------------------------------  
library IEEE;  
use IEEE.std_logic_1164.all;  
--  
-- pragma translate_off  
library UNISIM;  
use UNISIM.VCOMPONENTS.ALL;  
-- pragma translate_on  
--  
entity BUFGCE_1_SUBM is  
  port (  
    I: in std_logic;  
    CE: in std_logic;  
    O: out std_logic  
  );
architecture BUFGCE_1_SUBM_arch of BUFGCE_1_SUBM is

-- Component Declarations:
component BUFGMUX_1
port ( 
  I0   : in std_logic;
  I1   : in std_logic;
  S    : in std_logic;
  O    : out std_logic
); 
end component;
--
-- signal declarations
signal VCC : std_logic;
--
signal CE_B : std_logic;
--
begin
  VCC <= '1';
--
  CE_B <= not CE;
--
-- Global Clock Buffer Instantiation
U_BUFGMUX_1: BUFGMUX_1
  port map ( 
    I0     => I,
    I1     => VCC,
    S      => CE_B,
    O      => O
  );
-- 
end BUFGCE_1_SUBM_arch;

Verilog Template

// Module:      BUFGMUX_INST
// Description: Verilog Instantiation Template
// Global Clock Multiplexer (Switch Low)
//
// Device: Virtex-II Pro Family
//
/BUFGMUX Instantiation
BUFGMUX  U_BUFGMUX
  (.I0(),  // insert clock input used when select(S) is Low
   .I1(),  // insert clock input used when select(S) is High
   .S(),   // insert Mux-Select input
   .O()    // insert clock output
); 

// Module:      BUFGMUX_1_INST
// Description: Verilog Instantiation Template
// Global Clock Multiplexer (Switch High)
//
// Device: Virtex-II Pro Family
//-------------------------------------------------------------------
//BUFGMUX_1 Instantiation
BUFGMUX_1  U_BUFGMUX_1
  (.I0(),  // insert clock input used when select(S) is Low
   .I1(),  // insert clock input used when select(S) is High
   .S(),   // insert Mux-Select input
   .O()    // insert clock output
);
//-------------------------------------------------------------------
// Module: BUFGCE_SUBM
// Description: Verilog Submodule
// Global Clock Buffer with Clock Enable:
// Input Clock Buffer to BUFGMUX - Clock disabled = Low
//-------------------------------------------------------------------
module BUFGCE_SUBM (I,
  CE,
  O);
input   I,
  CE;
output  O;
wire GND;
assign GND = 1'b0;
BUFGMUX U_BUFGMUX
  (.I0(I),
   .I1(GND),
   .S(~CE),
   .O(O)
);
endmodule
//-------------------------------------------------------------------
// Module: BUFGCE_1_SUBM
// Description: Verilog Submodule
// Global Clock Buffer with Clock Enable:
// Input Clock Buffer to BUFGMUX_1 - Clock disabled = High
//-------------------------------------------------------------------
module BUFGCE_1_SUBM (I,
  CE,
  O);
input   I,
  CE;
output  O;
wire VCC;
assign VCC = 1'b1;
Digital Clock Managers (DCMs)

Overview

Virtex-II Pro / Virtex-II Pro X devices have four to twelve DCMs, and each DCM provides a wide range of powerful clock management features:

- **Clock De-Skew**: The DCM contains a delay-locked loop (DLL) that can completely eliminate clock distribution delays, hence deskewing the DCM’s output clocks with respect to the input clock. The DLL contains delay elements (individual small buffers) and control logic. The incoming clock drives a chain of delay elements, thus the output of every delay element represents a version of the incoming clock delayed at a different point.

  The control logic contains a phase detector and a delay line selector. The phase detector compares the incoming clock signal (CLKIN) against a feedback input (CLKFB) and steers the delay line selector, essentially adding delay to the output of DCM until the CLKIN and CLKFB coincide.

- **Frequency Synthesis**: Separate outputs provide a doubled frequency (CLK2X and CLK2X180). Another output (CLKDV) provides a frequency that is a specified fraction of the input frequency (÷1.5, ÷2, ÷2.5, and so forth, up to ÷15 and ÷16.)

  Two other outputs (CLKFX and CLKFX180) provide an output frequency that is derived from the input clock by simultaneous frequency division and multiplication. The user can specify any integer multiplier (M) and divisor (D) within the range specified in the DCM Timing Parameters section of the Data Sheet. An internal calculator figures out the appropriate tap selection, so that the output edge coincides with the input clock whenever that is mathematically possible. For example, M=9 and D=5, multiply the frequency by 1.8, and the output rising edge is coincident with the input rising edge every 5 input periods = every 9 output periods.

- **Phase Shifting**: Three outputs drive the same frequency as CLK0 but are delayed by 1/4, 1/2, and 3/4 of a clock period. An additional control optionally shifts all nine clock outputs by a fixed fraction of the clock period (defined during configuration, and described in multiples of the clock period divided by 256).

  The user can also dynamically and repetitively move the phase forwards or backwards by one unit of the clock period divided by 256. Note that any such phase shift is always invoked as a specific fraction of the clock period, but is always implemented by moving delay taps with a resolution of DCM_TAP (see the DCM Timing Parameters section in the Data Sheet).

- **General Control Signals**: The input is an asynchronous reset; when High, it resets the entire DCM (all clock outputs, LOCKED, and STATUS signals are brought LOW). The LOCKED output is High when all enabled DCM circuits have locked. The active High STATUS outputs indicate the following:
When using DCMs it is important to observe the $V_{CCAUX}$ recommended operating noise specification in the Data Sheet. Power distribution design details are available in Xilinx Application Note XAPP623.

Clock De-Skew

The Virtex-II Pro / Virtex-II Pro X Digital Clock Manager (DCM) offers a fully digital, dedicated on-chip de-skew circuit providing zero propagation delay, low clock skew between output clock signals distributed throughout the device, and advanced clock domain control. These features can be used to implement several circuits that improve and simplify system level design.

Any four of the nine outputs of the DCM can be used to drive a global clock network. All DCM outputs can drive general interconnect at the same time; for example, DCM output can be used to generate board-level clocks. The well-buffered global clock distribution network minimizes clock skew caused by loading differences. By monitoring a sample of the output clock (CLK0), the de-skew circuit compensates for the delay on the routing network, effectively eliminating the delay from the external input port to the individual clock loads within the device.

Figure 3-22 shows all of the inputs and outputs relevant to the DCM de-skew feature.

The de-skew feature can also act as a clock mirror. By driving the CLK0 output off-chip and then back in again, the de-skew feature can be used to de-skew a board-level clock serving multiple devices. Figure 3-23 shows an example.
Notes:
1. For a discussion of when to reset the DCM, see section “External Feedback,” page 93.

By taking advantage of the de-skew circuit to remove on-chip clock delay, the designer can greatly simplify and improve system level design involving high-fanout, high-performance clocks.

Operation

A de-skew circuit in its simplest form consists of variable delay line and control logic. The delay line produces a delayed version of the input clock (CLKIN). The clock distribution network routes the clock to all internal registers and to the clock feedback CLKFB pin. The control logic samples the input clock, as well as the feedback clock, and adjusts the delay line.

For optimum performance, the Virtex-II Pro/Virtex-II Pro X DCM uses a discrete digital delay line, which is a series of buffer elements each with an intrinsic delay of less than DCM_TAP (see AC Characteristics in the Data Sheet).

A de-skew circuit works by inserting delay between the input clock and the feedback clock until the two rising edges align, putting the two clocks 360 degrees out of phase, which means they are in phase. When the edges from the input clock line up with the edges from the feedback clock, the DCM achieves “lock.” The two clocks have no discernible
difference. Thus, the DCM output clock compensates for the delay in the clock distribution network, effectively removing the delay between the source clock and its loads.

**Input Clock Requirements**

The clock input of the DCM can be driven either by an IBUFG, an IBUF, or a BUFGMUX. An LVDS clock can also be used as input.

The output clock signal of a DCM, essentially a delayed version of the input clock signal, reflects any instability on the input clock in the output waveform. A DCM cannot improve the input jitter. The DCM input clock requirements are specified in the Data Sheet.

Once locked, the DCM can tolerate input clock period variations of up to the value specified by CLkin_PER_JITT_DLL_HF (at high frequencies) or CLKin_PER_JITT_DLL_LF (at low frequencies). Larger frequency changes can cause the DCM to lose lock, which is indicated by the LOCKED output going low. The user must then reset the DCM. The cycle-to-cycle input jitter must be kept to less than CLKin_CYC_JITT_DLL_LF in the low frequencies and CLKin_CYC_JITT_DLL_HF for the high frequencies.

**Input Clock Changes**

Changing the period of the input clock beyond the maximum input period jitter specification requires a manual reset of the DCM. Failure to reset the DCM produces an unreliable lock signal and output clock.

While the DCM is in the locking process, no input clock edge can be missing. Once locked, it is possible to temporarily stop the input clock with little impact to the de-skew circuit, as long as CLKFX or CLKFX180 is not used.

If the input clock is stopped and CLKFX or CLKFX180 is used, the CLKFX or CLKFX180 outputs might stop toggling, and STATUS[2] ("CLKFX Stopped") is asserted. The DCM must be reset to recover from this event.

CLKFX or CLKFX180 stops within D cycles of CLKin or when CLKFX is concurrent to CLKin. STATUS[2] is asserted within 1 to D CLKin + 1 CLKFX cycles of CLKFX or when CLKFX180 output stops. (D is the CLKFX_DIVIDE value.)

In any other cases, the clock should not be stopped for more than 100 ms to minimize the effect of device cooling; otherwise, the tap delays might change. The clock should be stopped during a Low or a High phase, and must be restored with the same input clock period/frequency. During this time, LOCKED stays High and remains High when the clock is restored. Thus, a High on LOCKED does not necessarily mean that a valid clock is available.

When the input clock is being stopped (CLKin remains High or Low for more than 1 clock cycle), one to eight more output clock cycles are still generated as the delay line is flushed. When the output clock stops, the CLKin stopped (STATUS(1)) signal is asserted. When the clock is restarted, the output clock cycles are not generated for one to eight clocks while the delay line is filled. Similarly, the STATUS(1) signal is deasserted once the output clock is generated. The most common case is two or three clocks. In a similar manner, a phase shift of the input clock is possible. For example, the input clock can be stopped for 12 ns to achieve a 12 ns phase shift at the output. The phase shift propagates one to eight clocks to the output after the original shift, with no disruption to the DCM control. The STATUS(1) is forced Low whenever LOCKED is Low.
Output Clocks

Some restrictions apply regarding the connectivity of the output pins. The DCM clock outputs can each drive an OBUF, a global clock buffer BUFGMUX, or they can route directly to the clock input of a synchronous element. To use the dedicated routing, the DCM clock outputs should drive BUFGMUXs that are on the same edge (top or bottom) of the device. If the DCM and BUFGMUX are not on the same edge, local routing is used and the DCM might not deskew properly.

Do not use the DCM output clock signals until after activation of the LOCKED signal. Prior to the activation of the LOCKED signal, the DCM output clocks are not valid and can exhibit glitches, spikes, or other spurious movement.

External Feedback

To ensure consistent locking, if a DCM is configured with external feedback, applying a reset after configuration is strongly recommended.

For an optimum locking process, a DCM configured with external feedback requires both CLkin and CLk0 to be present and stable when the DCM begins to lock. During configuration with external feedback, it is not possible to provide CLKFB in the beginning of the locking process. At the end of configuration, the DCM begins to lock once the device enters the startup sequence. Because a global 3-state signal (GTS) is still asserted during this time, the output I/O pins remain in a 3-state condition, effectively putting the CLKFB signal into a 3-state condition.

When CLKFB eventually appears (after the GTS is deasserted), the DCM proceeds with the lock. However, it might not lock at the optimal point and can introduce slightly more jitter (as well as greater clock cycle latency) through the DCM.

In addition, if CLKFB is coupling with another signal when it is put into a 3-state condition (a PCB signal integrity issue), the DCM might sense this invalid clock as CLKFB and use it to proceed with a lock. This second possibility can cause the DCM to not lock properly once the GTS deasserts and the true CLKFB signal is present.

Use of reset after configuration guarantees that the locking process starts with a valid CLkin and CLKFB signal and ensures consistent locking.

Notes:
1. The default value is -g LCK_cycle:NoWait and -g GTS_cycle:5. When these settings are used, the startup sequence does not wait for the DCM to lock.
2. If the startup sequence is altered (for example, using the BitGen option), the LCK_cycle (wait for DCM to lock) must not be placed before the GTS_cycle (de-assert GTS); otherwise, the DCM never locks and configuration does not complete.
3. HDL example code to reset the DCM after configuration are shown below. This example requires that GTS_cycle be set before GWE_cycle in the BitGen options (the default). This ensures that the DCM is reset after the I/O pins are activated.

Verilog Example:
// DCM instantiation to show the reset connection

```verilog
DCM u_dcml (.CLKIN(clkin),
.CLKFB(clkfb),
.CLK0(clk0),
.RST(rstin),
.LOCKED(locked));

// rstin connects to RST pin of DCM
assign rstin = (user_reset || config_rst);
```
// This is the actual reset circuit that outputs config_rst. It is a four-cycle shift register.

FDS flop1 (.D(1'b0), .C(clkin), .Q(out1), .S(1'b0));
FD flop2 (.D(out1), .C(clkin), .Q(out2));
FD flop3 (.D(out2), .C(clkin), .Q(out3));
FD flop4 (.D(out3), .C(clkin), .Q(out4));

// config_rst will be asserted for 3 clock cycles.
assign config_rst = (out2 | out3 | out4);

VHDL Example:

-- DCM instantiation to show the reset connection

u_dcm1: DCM port map (
  CLKIN => clkin,
  CLKFB => clkfb,
  CLK0 => clk0,
  RST => rstin,
  LOCKED => locked);

-- rstin connects to RST pin of DCM
rstin <= user_reset or config_rst;

-- This is the actual reset circuit that outputs config_rst. It is a four-cycle shift register.

flop1: FDS port map (D => '0', C => clkin, Q =>
  out1, S => '0');
flop2: FD port map (D => out1, C => clkin, Q => out2);
flop3: FD port map (D => out2, C => clkin, Q => out3);
flop4: FD port map (D => out3, C => clkin, Q => out4);

-- config_rst will be asserted for 3 clock cycles.
config_rst <= out2 or out3 or out4;

Characteristics of the De-Skew Circuit

- Can eliminate clock distribution delay by effectively adding one clock period delay. Clocks are de-skewed to within CLKOUT_PHASE, specified in the Data Sheet.
- Can be used to eliminate on-chip as well as off-chip clock delay.
- Has no restrictions on the delay in the feedback clock path.
- Requires a continuously running input clock.
- Adapts to a wide range of frequencies. However, once locked to a frequency, cannot tolerate large variations of the input frequency.
- De-skew circuit is part of the DCM, which also includes phase adjustment, frequency synthesis, and spread spectrum techniques that are described in this document.
- Does not eliminate jitter. The de-skew circuit output jitter is the sum of input jitter and some jitter value that the de-skew circuit might add.
- The completion of configuration can be delayed until after DCM locks to guarantee the system clock is established prior to initiating the device.
Digital Clock Managers (DCMs)

Port Signals

Source Clock Input — CLKin

The CLKin pin provides the user source clock (the clock signal on which the de-skew circuit operates) to the DCM. The CLKin frequency must fall in the ranges specified in the Data Sheet. The clock input signal can be provided by one of the following:

- IBUF — Input buffer
- IBUFG — Global clock input buffer on the same edge of the device (top or bottom)
- BUFGMUX — Internal global clock buffer

**Note:** When IBUF is used as the clock input, the PAD to DCM input skew is not compensated. Refer to Table 3-1, page 72 for more information.

Feedback Clock Input — CLKFB

A reference or feedback signal is required to delay-compensate the output. Connect only the CLK0 DCM outputs to the feedback clock input (CLKFB) pin to provide the necessary feedback to the DCM. The feedback clock input signal can be driven by an internal global clock buffer (BUFGMUX), one of the global clock input buffers (IBUFG) on the same edge of the device (top or bottom), or IBUF (the input buffer.) Note that when IBUF is used, the PAD to DCM input skew is not compensated.

If an IBUFG sources the CLKFB pin of a DCM pin, the following special rules apply:

1. The CLK0 of the DCM must directly drive an OBUF or a BUFG-to-DDR configuration to forward the clock.
2. External to the FPGA, the same forwarded clock signal must be connected to the IBUFG (GCLK pin) that drives the CLKFB of the DCM.

Reset Input — RST

When the reset pin is activated, the LOCKED signal deactivates within four source clock cycles. The RST pin, active High, must be connected to a dynamic signal or tied to ground. For all designs that use the feedback feature of the DCM, applying a reset signal after configuration is strongly recommended for both production and ES device in order to ensure consistent locking. As the DCM delay taps reset to zero, glitches can occur on the DCM clock output pins. Activation of the RST pin can also severely affect the duty cycle of the clock output pins. Furthermore, the DCM output clocks no longer de-skew with respect to one another and are eventually stopped Low if Status signals are deactivated (brought to Low). For these reasons, use the reset pin only when reconfiguring the device or changing the input frequency. The reset input signal is asynchronous and should be held High for at least three clock cycles. The time it takes for the DCM to achieve lock is specified as LOCK_DLL (for DLL output) and LOCK FX (for DFS output). The DCM locks faster at higher frequencies. See the LOCK_DLL timing parameter in the Data Sheet.

Locked Output — LOCKED

In order to achieve lock, the DCM may need to sample several thousand clock cycles. After the DCM achieves lock, the LOCKED signal goes High. The DCM timing parameters section of the Data Sheet provides estimates for locking times.

To guarantee that the system clock is established prior to the device “waking up,” the DCM can delay the completion of the device configuration process until after the DCM locks. The STARTUP_WAIT attribute activates this feature.
Until the LOCKED signal activates, the DCM output clocks are not valid and can exhibit glitches, spikes, or other spurious movement. In particular, the CLK2X output appears as a 1x clock with a 25/75 duty cycle.

Status - STATUS
The STATUS output is an 8-bit output, of which STATUS[1] reveals the loss of the input clock, CLkin to the DCM.

Attributes
The following attributes provide access to some of the Virtex-II Pro / Virtex-II Pro X series de-skew features, (for example, clock division and duty cycle correction).

Frequency Mode
The de-skew feature of the DCM is achieved with a delay-locked loop (DLL). This attribute specifies either the high or low-frequency mode of the DLL. The default is low-frequency mode. In high-frequency mode, the only outputs available from the DLL are the CLK0, CLK180, CLKDV, and LOCKED. (CLK90, CLK270, CLK2X, and CLK2X180 are not available in high-frequency mode.) The frequency ranges for both frequency modes are specified in the Data Sheet. To set the DLL to high-frequency mode, attach the DLL_FREQUENCY_MODE=HIGH attribute in the source code or schematic.

Duty Cycle Correction
The 1x clock outputs, CLK0, CLK90, CLK180, and CLK270, use the duty cycle corrected default such that they exhibit a 50/50 duty cycle. The DUTY_CYCLE_CORRECTION attribute (by default TRUE) controls this feature.

Setting DUTY_CYCLE_CORRECTION=FALSE deactivates duty cycle correction for the DCM outputs listed above. It is strongly recommended to always set the DUTY_CYCLE_CORRECTION attribute to TRUE (default value). Note that setting this attribute to FALSE does not necessarily produce output clocks with the same duty cycle as the source clock.

Startup Delay
The default value of the STARTUP_WAIT attribute is FALSE. When STARTUP_WAIT is set to TRUE, and the LCK_cycle BitGen option is used, then the configuration startup sequence waits in the specified cycle until the DCM locks. For details, see Chapter 4, “Configuration” and Appendix A, “BitGen and PROMGen Switches and Options.”
Legacy Support

The Virtex/Virtex-E library primitives/sub modules are supported in Virtex-II Pro /
Virtex-II Pro X for legacy purposes. The following are supported primitives/submodules:

- CLKDLL
- CLKDLLE
- CLKDLLHF
- BUFGDLL

Library Primitive

Only a single library primitive is available for the DLL, a part of the DCM. It is labeled the
'DCM' primitive.

Submodules

Figure 3-24: **BUFG_CLK0_SUBM**

Figure 3-25: **BUFG_CLK0_FB_SUBM**
Frequency Synthesis

The DCM provides several flexible methods for generating new clock frequencies. Each method has a different operating frequency range and different AC characteristics. The CLK2X and CLK2X180 outputs double the clock frequency. The CLKDV output provides divided output clocks with division options of 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, and 16.

The DCM also offers a fully digital, dedicated Frequency Synthesizer output (CLKFX) and its opposite phase (CLKFX180). The output frequency can be any function of the input clock frequency described by \( M \div D \), where \( M \) is the multiplier (numerator) and \( D \) is the divisor (denominator).

The two counter-phase frequency synthesized outputs can drive global clock routing networks within the device. The well-buffered global clock distribution network minimizes clock skew due to differences in distance or loading. See Figure 3-27.

Figure 3-26: **BUFG_CLKDV_SUBM**

Figure 3-27: **Frequency Synthesis Outputs**
Operation

The DCM clock output CLKFX is any M/D product of the clock input to the DCM. Specifications for M and D, as well as input and output frequency ranges for the frequency synthesizer, are provided in the Data Sheet. The frequency synthesizer output is phase aligned to the clock output, CLK0, only if feedback is provided to the CLKFB input of the DCM.

The internal operation of the frequency synthesizer is complex and beyond the scope of this document. The frequency synthesizer multiplies the incoming frequencies by the pre-calculated quotient M/D and generates the correct output frequencies as long as it is within the range specified in the Data Sheet.

For example, assume input frequency = 50 MHz, M = 25, and D = 8 (note that M and D values have no common factors and hence cannot be reduced). The output frequency is correctly 156.25 MHz, although 25 x 50 MHz = 1.25 GHz and 50 MHz / 8 = 6.25 MHz, and both of these values are far outside the range of the input frequency.

Frequency Synthesizer Characteristics

- The frequency synthesizer provides an output frequency equal to the input frequency multiplied by M and divided by D.
- The outputs CLKFX and CLKFX180 always have a 50/50 duty-cycle.
- Smaller M and D values achieve faster lock times. The user should divide M and D by the largest common factor.
- The outputs are phase aligned with CLK0 when CLKFB is connected.

Port Signals

Source Clock Input — CLKin

The CLKin pin provides the user source clock to the DCM. The CLKin frequency must fall in the ranges specified in the Data Sheet. The clock input signal can be provided by one of the following:

- IBUF — Input buffer
- IBUFG — Global clock input buffer
- BUFGMUX — Internal global clock buffer

Note: When IBUF is used as the clock input, the PAD to DCM input skew is not compensated. Refer to Table 3-1, page 72 for more information.

2x Clock Output — CLK2X

The CLK2X output provides a frequency-doubled clock with an automatic 50/50 duty-cycle correction. This output is not available in high-frequency mode.

Until the DCM has achieved lock, the CLK2X output appears as a 1x version of the input clock with a 25/75 duty cycle. This behavior allows the DCM to lock on the correct edge with respect to source clock.

Clock Divide Output — CLKDv

The clock divide output pin CLKDv provides a lower frequency version of the source clock. The CLKDv_DIVIDE property controls CLKDv such that the source clock is divided by N where N is either 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, or 16.
This feature provides automatic duty cycle correction such that the CLKDV output pin has a 50/50 duty cycle always in low-frequency mode, as well as for all integer values of the division factor N in high-frequency mode.

**Frequency Synthesized Clock Output - CLKFX**

The CLKFX output provides a frequency-synthesized clock (M/D * CLKIN) with a 50/50 duty cycle. For the CLKFX output to be phase-aligned with CLKIN, the clock feedback (CLK0) must be provided at the CLKFB input. With M and D adjusted such that they have no common factor, the alignment occurs only once every D input clock cycles. There is no CLKFX phase alignment when the CLKIN frequency is below 24 MHz.

**Frequency Synthesized Clock Output 180° Phase Shifted - CLKFX180**

The CLKFX180 output is a 180° phase shifted version of the CLKFX clock output, also with a 50/50 duty cycle.

**Locked Output — LOCKED**

The LOCKED signal is activated after the DCM has achieved the parameter values set by the user parameters. To guarantee that the system clock is established prior to the device “waking up,” the DCM can delay the completion of the device configuration process until after the DCM locks. The STARTUP_WAIT attribute activates this feature. Until the LOCKED signal activates, the DCM output clocks are not valid and can exhibit glitches, spikes, or other spurious signals.

**Reset Input — RST**

When the reset pin activates, the LOCKED signal deactivates within four source clock cycles. The M and D values at configuration are maintained after the reset. The RST pin, active High, must either connect to a dynamic signal or be tied to ground. Activation of the RST pin can also severely affect the duty cycle of the clock output pins. For this reason, activate the reset pin only when reconfiguring the device or changing the input frequency. The reset input signal is asynchronous and should be held High for at least 2 ns.

**Status - STATUS**

The STATUS output is an 8-bit output:

- STATUS[1] indicates the loss of the input clock, CLKIN, only when CLKFB is connected.
- STATUS[2] indicates loss of CLKFX and CLKFX180 even though LOCKED might still be High. Note that this “CLKFX stopped” status functions only when CLKIN is present.

**Attributes**

The following attributes provide access to some of the Virtex-II Pro/Virtex-II Pro X series frequency synthesis features, (for example, clock multiplication, clock division).

**Clock Divide**

The CLKDV_DIVIDE attribute specifies how the signal on the CLKDV pin is frequency divided with respect to the CLK0 pin. The values allowed for this attribute are 1.5, 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5, 6, 6.5, 7, 7.5, 8, 9, 10, 11, 12, 13, 14, 15, or 16; the default value is 2.

**Frequency Mode for Frequency Synthesis**

This attribute specifies either the high or low-frequency mode of the frequency synthesizer. The default is low-frequency mode. The frequency ranges for both frequency modes are specified in the Data Sheet.
To set the frequency synthesizer to high-frequency mode, attach the DFS_FREQUENCY_MODE=HIGH attribute in the source code or schematic.

**Multiply/Divide Attribute**

The M and D values can be set using the CLKFX_MULTIPLY and the CLKFX_DIVIDE attributes. The default settings are M = 4 and D = 1.

**Startup Delay**

The default value of the STARTUP_WAIT attribute is FALSE. When STARTUP_WAIT is set to TRUE, and the LCK_cycle BitGen option is used, then the configuration startup sequence waits in the specified cycle until the DCM locks. For details, see Chapter 4, “Configuration” and Appendix A, “BitGen and PROMGen Switches and Options.”

**Submodules**

Figure 3-28: **BUFG_DFS_SUBM**

Figure 3-29: **BUFG_DFS_FB_SUBM**
Phase Shifting

The DCM can also provide coarse and fine-grained phase shifting. The CLK0, CLK90, CLK180, and CLK270 outputs are each phase shifted by ¼ of the input clock period relative to each other, providing coarse phase control. Note that CLK90 and CLK270 are not available in high-frequency mode.

Operation

Figure 3-30 shows a block diagram of the DCM and all of the outputs affected by the circuitry of the phase shift feature.

![Phase Shift Outputs](image_url)

**Figure 3-30: Phase Shift Outputs**

Fine-phase adjustment affects all nine DCM output clocks. When activated, the phase shift between the rising edges of CLKIN and CLKFB is a specified fraction of the input clock period.

In variable mode, the PHASE_SHIFT value can also be dynamically incremented or decremented as determined by PSINCDEC synchronously to PSCLK, when the PSEN input is active. Figure 3-31 illustrates the effects of fine-phase shifting.
Digital Clock Managers (DCMs)

Two separate components of the phase shift range must be understood:

- **PHASE_SHIFT** attribute range
- **FINE_SHIFT_RANGE** DCM timing parameter range

The **PHASE_SHIFT** attribute is the numerator in the following equation:

\[
\text{Phase Shift (ns)} = \frac{\text{PHASE SHIFT}}{256} \times \text{PERIODCLKIN}
\]

The full range of this attribute is always -255 to +255, but its practical range varies with CLKin frequency, as constrained by the **FINE_SHIFT_RANGE** component, which represents the total delay achievable by the phase shift delay line. Total delay is a function of the number of delay taps used in the circuit. Across process, voltage, and temperature, this absolute range is guaranteed to be as specified in the DCM Timing Parameters section of the Data Sheet.

Absolute range (fixed mode) = ± **FINE_SHIFT_RANGE**

Absolute range (variable mode) = ± **FINE_SHIFT_RANGE** / 2

The reason for the difference between fixed and variable modes is as follows. For variable mode to allow symmetric, dynamic sweeps from -255/256 to +255/256, the DCM sets the 'zero phase skew' point as the middle of the delay line, thus dividing the total delay line range in half. In fixed mode, since the **PHASE_SHIFT** value never changes after configuration, the entire delay line is available for insertion into either the CLKin or CLKFB path (to create either positive or negative skew).

Taking both of these components into consideration, the following are some usage examples:

- If **PERIOD_CLKIN** = two times **FINE_SHIFT_RANGE**, then **PHASE_SHIFT** in fixed mode is limited to ± 128, and in variable mode it is limited to ± 64.
- If **PERIOD_CLKIN** = **FINE_SHIFT_RANGE**, then **PHASE_SHIFT** in fixed mode is limited to ± 255, and in variable mode it is limited to ± 128.
- If **PERIOD_CLKIN** ≤ half of the **FINE_SHIFT_RANGE**, then **PHASE_SHIFT** is limited to ± 255 in either mode.

In variable mode, the phase factor can be changed by activating PSEN for one period of PSCLK. Increments or decrements to the phase factor can be made by setting the
PSINCDEC pin to a High or Low, respectively. When the de-skew circuit has completed an increment or decrement operation, the signal PSDONE goes High for a single PSCLK cycle. This indicates to the user that the next change may be made.

The user interface and the physical implementation are different. The user interface describes the phase shift as a fraction of the clock period (N/256). The physical implementation adds the appropriate number of buffer stages (each DCM_TAP) to the clock delay. The DCM_TAP granularity limits the phase resolution at higher clock frequencies.

### Phase Shift Characteristics

- Offers fine-phase adjustment with a resolution of ±1/256 of the clock period (or ± one DCM_TAP, whichever is greater) by configuration and also dynamically under user control.
- The phase shift settings affect all nine DCM outputs.
- VCC and temperature do not affect the phase shift.

The minimum granularity is the greater of two limiting factors:

1. The minimum phase shift step size = 1/256 * CLKIN_Period.
2. The tap delay resolution (DCM_TAP). The specifications for DCM_TAP_MIN and DCM_TAP_MAX are available in the datasheet.

Therefore, assuming a typical DCM_TAP of 45 ps, at ~90 Mhz with an 11.11 ns period, the 1/256 * CLKIN_Period is ~45 ps.

- At a frequency < approximately 90 MHz, 1/256 * CLKIN_Period > 45 ps; hence the minimum step size is limited by #1 above.
- At a frequency > approximately 90 MHz, 1/256 * CLKIN_Period < 45 ps; hence the minimum step size is limited by #2 above.

At any condition, regardless of the CLKIN frequency, the PHASE_SHIFT (PS) value is determined by the following equation:

\[
\text{"desired phase shift"} = \left(\frac{PS}{256}\right) \times \text{CLKIN period}
\]

From this PS value, the DPS function of DCM will select the appropriate tap setting. This tap setting will vary across process, voltage, and temperature (PVT).

For example, if CLKIN = 150 Mhz (6.67 ns period), and the "desired phase shift" = 680 ps:

\[
0.680 = PS/256 \times 6.67 \\
PS = 26
\]

(This will be the PHASE_SHIFT value.)

The DCM_TAP of ~45 ps is only an approximation. The actual tap value can range from 30 ps to 60 ps at any given time, depending upon PVT. Hence, 10 tap steps does not always mean 10 * 45 ps = 450 ps. In addition, the DCM might not adjust to a new phase shift at every 1/256 phase shift increment if 1/256 * CLK_Period < DCM_TAP, because 1/256 is not equal to 1 tap.

For example, when you change the PS value from "3" to "4", the DCM may not increment the phase shift if the current phase shift is closer to the ideal value when compared to the next tap at the PVT.

In this case, the design consideration is that at any time, the DCM will find the closest phase shift given the PHASE_SHIFT attribute value. The error at this PHASE_SHIFT for a
simple case of using CLK0 will be CLKOUT_PER_JITT_0 + CLkin_CLOCK FB_PHASE. This is specified in the datasheet.

The DCM_TAP value is included in the output jitter specs along with random jitter introduced by the DCM and by the FPGA. Using a FIXED phase shift will allow the design to be within 5-10% of the desired phase shift point. If more accuracy is needed, use the VARIABLE mode to dial it in.

Fixed-Mode Phase Shifting in Certain Virtex-II Pro Devices

The use of negative, fixed-mode phase shifting requires a minor modification to the clock muxing in certain older Virtex-II Pro devices. Refer to Solution Record 13349 for information on how to identify these older devices.

In devices requiring this special clock muxing treatment, CLkin must be used to drive CLKFB while the DCM is in reset (when the RST signal is held High or during the startup sequence after configuration). Figure 3-32 and the example Verilog and VHDL code below illustrate the implementation of this simple modification. This modification is NOT required for any positive phase shifting or for any variable-mode phase shifting.

Note that this requirement applies to any DCM using fixed-mode phase shifting, regardless of whether the RST of the DCM is connected to a user signal. RST must be held High for at least three clock cycles.

Solution 1: Clock-Muxing Modification

Figure 3-32 illustrates the clock-muxing modification.

---

**Note:** PAR might report a warning that the IBUFG (CLkin)-to-BUFGMUX connection is not an optimal connection and so does not use the fast connection between the two components. This warning can be safely ignored; in this case, a dedicated connection is not necessary for the IBUFG-BUFGMUX connection.
Chapter 3: Design Considerations

Verilog Example:

// DCM instantiation
DCM U_DCM ( .CLKIN(clkin), .CLKFB(clk0_new), .RST(rst), .CLK0(clk0),
            .LOCKED(locked) );

// BUFGMUX instantiation
BUFGMUX U_BUFGMUX ( .O(clk0_new), .I0(clk0), .I1(clkin), .S(rst) );

VHDL Example:

-- DCM instantiation --
U_DCM: DCM port map ( CLKIN => clkin,
                   CLKFB => clk0_new,
                   DSSEN => '0',
                   PSCLK => '0',
                   PSINCDEC => '0',
                   RST => rst,
                   CLK0 => clk0,
                   LOCKED => locked);

-- BUFGMUX instantiation --
U_BUFGMUX: BUFGMUX( I0 => clk0
                     I1 => clkin,
                     O => clk0_new);

Solution 2: Alternative Workaround Using Positive Phase Shift

This problem can also be resolved by using an equivalent positive PHASE_SHIFT value. Given the current data sheet specification of FINE_SHIFT_RANGE = 10 ns, this restriction begins to have an effect for input frequencies below 100 MHz (input periods larger than 10 ns).

Given the equation:

\[ \text{Phase shift} = \left( \frac{\text{PS}}{256} \right) \times \text{PeriodCLKIN} \]

For a 10 ns input clock period:

- Variable mode: $-128 \leq \text{PS} \leq 128$
- Fixed mode: $0 \leq \text{PS} \leq 255$ (full range)

For a 20 ns input clock period:

- Variable mode: $-64 \leq \text{PS} \leq 64$
- Fixed mode: $0 \leq \text{PS} \leq 128$

For a 40 ns input clock period:

- Variable mode: $-32 \leq \text{PS} \leq 32$
- Fixed mode: $0 \leq \text{PS} \leq 64$

In either fixed or variable mode, the range can be extended by choosing CLK90, CLK180, or CLK270, rather than CLK0, choosing CLK2X180 rather than CLK2X, or choosing CLKFX180 rather than CLKFX. Even at 25 MHz (40 ns period), the fixed mode coupled with these CLK* phases allows shifting throughout the entire input clock period range. See Figure 3-33.

Furthermore, the phase-shifting (DPS) function in the DCM requires the CLKFB for delay adjustment.

Because CLKFB must be from CLK0, the DLL output is used. Hence the minimum CLKin frequency for the DPS function is 24 MHz.
Digital Clock Managers (DCMs)

Solution 3: Alternative Workaround Using Variable Phase Shift

Configure the DCM with variable phase shifting, as follows:

1. Set the CLK_OUT_PHASE_SHIFT attribute to VARIABLE.
2. Set the PHASE_SHIFT attribute to the required value (a negative value is acceptable).
3. Connect PSCLK to a clock signal.
4. Connect PSEN to Ground if incrementing or decrementing the phase shift is not intended.
5. Connect PSINCDEC to a signal, GND, or VCC.

Port Signals

1x Clock Outputs — CLK[0|90|180|270]

The 1x clock output pin CLK0 represents a delay-compensated version of the source clock (CLKIN) signal. In low-frequency mode, the DCM provides three phase-shifted versions of the CLK0 signal (CLK90, CLK180, and CLK270), whereas in high-frequency mode, only the 180 phase-shifted version is provided. All four (including CLK0) of the phase shifted outputs can be used simultaneously in low-frequency mode. The relationship between phase shift and the corresponding period shift appears in Table 3-8.

Figure 3-33: Fixed Phase Shift Example

Table 3-8: Relationship of Phase-Shifted Output Clock to Period Shift

<table>
<thead>
<tr>
<th>Phase (degrees)</th>
<th>% Period Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0%</td>
</tr>
<tr>
<td>90</td>
<td>25%</td>
</tr>
<tr>
<td>180</td>
<td>50%</td>
</tr>
<tr>
<td>270</td>
<td>75%</td>
</tr>
</tbody>
</table>
By default, the DCM provides a 50/50 duty cycle correction on all 1x clock outputs. The DUTY_CYCLE_CORRECTION attribute (TRUE by default), controls this feature. Attach the DUTY_CYCLE_CORRECTION=FALSE property to the DCM symbol in order to deactivate the DCM duty cycle correction. With duty cycle correction deactivated, the output clocks have the same duty cycle as the source clock.

The DCM clock outputs can drive an OBUF, a BUFGMUX, or they can route directly to the clock input of a synchronous element.

![Diagram of DLL Output Characteristics](x132_07_002599)

*Figure 3-34: DLL Output Characteristics*
Digital Clock Managers (DCMs)

Source Clock Input — CLKIN

The CLKIN pin provides the user source clock to the DCM. The CLKIN frequency must fall in the ranges specified in the Data Sheet. The clock input signal can be provided by one of the following:

- IBUF — Input buffer
- IBUFG — Global clock input buffer
- BUFGMUX — Internal global clock buffer

*Note:* When IBUF is used as the clock input, the PAD to DCM input skew is not compensated. Refer to Table 3-1, page 72 for more information.

Feedback Clock Input — CLKFB

A DCM requires a reference or feedback signal to provide delay-compensated output. Connect only the CLK0 DCM outputs to the feedback clock input (CLKFB) pin to provide the necessary feedback to the DCM. The feedback clock input signal can be driven by an internal global clock buffer (BUFGMUX), one of the global clock input buffers (IBUFG) on the same edge of the device (top or bottom), or IBUF (the input buffer.)

If an IBUFG sources the CLKFB pin, the following special rules apply:

1. The CLK0 of the DCM must directly drive an OBUF or a BUFG-to-DDR configuration to forward the clock.
2. External to the FPGA, the same forwarded clock signal must be connected to the IBUFG (GCLK pin) that drives the CLKFB of the DCM.

Phase Shift Clock - PSCLK

The PSCLK input can be sourced by the CLKIN signal to the DCM, or it can be a lower or higher frequency signal provided from any clock source (external or internal). The frequency range of PSCLK is defined by PSCLK_FREQ_LF/HF (see the Data Sheet). This input has to be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED.

Phase Shift Increment/Decrement - PSINCDEC

The PSINCDEC signal is synchronous to PSCLK and is used to increment or decrement the phase shift factor. In order to increment or decrement the phase shift by $\frac{1}{256}$ of the clock period, the PSINCDEC signal must be High for increment or Low for decrement. This input has to be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED.

Phase Shift Enable - PSEN

To initiate a variable phase-shift operation, the PSEN input must be activated for one period of PSCLK. The phase change becomes effective after up to 100 CLKIN pulse cycles plus three PSCLK cycles, and is indicated by a High pulse on PSDONE. During the phase transition there are no sporadic changes or glitches on any output. From the time when PSEN is enabled until PSDONE is flagged, the DCM output clock will slide bit-by-bit from its original phase shift to the incremented phase shift. The increment/decrement will be ready by PSDONE. PSEN must be tied to ground when the CLKOUT_PHASE_SHIFT attribute is set to NONE or FIXED.
Reset Input — RST

When the reset pin is activated, the LOCKED signal deactivates within four source clock cycles. After reset, the phase shift value is set to its value at configuration in both the fixed and variable modes. The RST pin, active High, must either connect to a dynamic signal or be tied to ground. Activation of the RST pin can also severely affect the duty cycle of the clock output pins. For this reason, activate the reset pin only when reconfiguring the device or changing the input frequency. The reset input signal is asynchronous and should be held High for at least 2 ns.

Locked Output — LOCKED

The LOCKED signal activates after the DCM has achieved lock. To guarantee that the system clock is established prior to the device “waking up,” the DCM can delay the completion of the device configuration process until after the DCM locks. The STARTUP_WAIT attribute activates this feature. Until the LOCKED signal activates, the DCM output clocks are not valid and can exhibit glitches, spikes, or other spurious movement. For details, refer to Chapter 4, “Configuration.”

Phase Shift DONE - PSDONE

The PSDONE signal is synchronous to PSCLK and it indicates, by pulsing High for one period of PSCLK, that the requested phase shift was achieved. This signal also indicates to the user that a new change to the phase shift numerator can be made. This output signal is not valid if the phase shift feature is not being used or is in fixed mode.

Status - STATUS

STATUS[0] indicates the overflow of the phase shift numerator (when DCM is phase shifted beyond +255 or -255) and that the absolute delay range of the phase shift delay line is exceeded.

Attributes

The following attributes provide access to the Virtex-II Pro / Virtex-II Pro X fine-phase adjustment capability.

Clock Out Phase Shift

The CLKOUT_PHASE_SHIFT attribute controls the use of the PHASE_SHIFT value. It can be set to NONE, FIXED, or VARIABLE. By default, this attribute is set to NONE, indicating that the phase shift feature is not being used. When this attribute is set to NONE, the PHASE_SHIFT value has no effect on the DCM outputs. If the CLKOUT_PHASE_SHIFT attribute is set to FIXED or NONE, then the PSEN, PSINCDEC, and the PSCLK inputs must be tied to ground. The effects of the CLKOUT_PHASE_SHIFT attribute are shown in Figure 3-31.

PHASE_SHIFT

This attribute specifies the phase shift numerator as any value from -255 to 255. This attribute can be used with both fixed and variable phase shift mode. If used with variable mode, the attribute sets the starting phase shift.
Figure 3-35: **BUFG_PHASE_CLKFX_FB_SUBM**

Figure 3-36: **BUFG_PHASE_CLK0_SUBM**
Figure 3-37: **BUFG_PHASE_CLKDV_SUBM**
Summary of All DCM Attributes

A handful of DCM attributes govern the functionality of DCM features. Below is a collection of all DCM attributes. Table 3-9 summarizes all attributes applicable to DCMs.

Table 3-9: DCM Attributes

<table>
<thead>
<tr>
<th>DCM Attribute Name</th>
<th>Description</th>
<th>Value</th>
<th>Default Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLKDV_DIVIDE</td>
<td>This attribute controls CLKDV such that the source clock is divided by N.</td>
<td>Real: 1.5 &lt;= N+0.5 &lt;= 8, 8 &lt; N+1 &lt; 16</td>
<td>2.0</td>
</tr>
<tr>
<td></td>
<td>This feature provides automatic duty cycle correction such that the</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>CLKDV output pin has a 50/50 duty cycle always in low-frequency mode, as</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>well as for all integer values of the division factor N in high-frequency</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mode.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKFX_DIVIDE</td>
<td>Integer: 1 to 32</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>CLKFX_MULTIPLY</td>
<td>Integer: 2 to 32</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>CLKIN_PERIOD</td>
<td>This specifies the source clock period to help DCM adjust for optimum</td>
<td>Real in ns</td>
<td>0.0</td>
</tr>
<tr>
<td></td>
<td>CLKFX/CLKFX180 outputs.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKIN_DIVIDE_BY_2</td>
<td>This attribute allows for the input clock frequency to be divided in half</td>
<td>Boolean: FALSE or TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td></td>
<td>when such a reduction is necessary to meet the DCM input clock frequency</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>requirements.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CLKOUT_PHASE_SHIFT</td>
<td>This controls the use of the PHASE_SHIFT value.</td>
<td>String: “NONE” or “FIXED” or “VARIABLE”</td>
<td>NONE</td>
</tr>
<tr>
<td>DESKEW_ADJUST</td>
<td>This affects the amount of delay in the feedback path, and should be used</td>
<td>String: “SYSTEM_SYNCHRONOUS” or</td>
<td>SYSTEM_SYNCHRONOUS</td>
</tr>
<tr>
<td></td>
<td>for source-synchronous interfaces. [See the “Clock De-Skew” section.]</td>
<td>“SOURCE_SYNCHRONOUS” or “0 to 15”</td>
<td></td>
</tr>
<tr>
<td>DFS_FREQUENCY_MODE</td>
<td>This specifies the frequency mode of the frequency synthesizer.</td>
<td>String: “LOW” or “HIGH”</td>
<td>LOW</td>
</tr>
<tr>
<td>DLL_FREQUENCY_MODE</td>
<td>This specifies the DLL’s frequency mode. CLK90,</td>
<td>String: “LOW” or “HIGH”</td>
<td>LOW</td>
</tr>
<tr>
<td></td>
<td>CLK270, CLK2X, and CLK2X180 are not available in high-frequency mode.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Chapter 3: Design Considerations

For more information on applying these attributes in UCF, VHDL, or Verilog code, refer to the Constraints Guide at: http://toolbox.xilinx.com/docsan/xilinx4/manuals.htm.

VHDL and Verilog Instantiation

VHDL and Verilog instantiation templates are available in the Libraries Guide for all submodules.

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

VHDL and Verilog Templates

The following submodules described in this section are available:

- BUFG_CLK0_SUBM
- BUFG_CLK0_FB_SUBM
- BUFG_CLKDV_SUBM
- BUFG_DFS_SUBM
- BUFG_DFS_FB_SUBM
- BUFG_PHASE_CLKFX_FB_SUBM
- BUFG_PHASE_CLK0_SUBM
- BUFG_PHASE_CLKDV_SUBM

The corresponding submodules must be synthesized with the design. The BUFG_CLK0_SUBM submodule is provided in VHDL and Verilog as an example.

VHDL Template

```vhdl
-- Module: BUFG_CLK0_SUBM
-- Description: VHDL submodule
```
-- DCM with CLK0 deskew
-- Device: Virtex-II Pro Family
---------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
--
-- pragma translate_off
library UNISIM;
use UNISIM.VCOMPONENTS.ALL;
-- pragma translate_on
--
entity BUFG_CLK0_SUBM is
    port ( 
        CLK_IN : in std_logic;
        RST    : in std_logic;
        CLK1X  : out std_logic;
        LOCK   : out std_logic
    );
end BUFG_CLK0_SUBM;
--
architecture BUFG_CLK0_SUBM_arch of BUFG_CLK0_SUBM is
-- Components Declarations:
component BUFG
    port ( 
        I : in std_logic;
        O : out std_logic
    );
end component;
component DCM
-- pragma translate_off
generic ( 
    DLL_FREQUENCY_MODE : string := "LOW";
    DUTY_CYCLE_CORRECTION : boolean := TRUE;
    STARTUP_WAIT : boolean := FALSE
);
-- pragma translate_on
--
port ( 
    CLKIN     : in  std_logic;
    CLKFB     : in  std_logic;
    DSSEN     : in  std_logic;
    PSINCDEC  : in  std_logic;
    PSEN      : in  std_logic;
    PSCLK     : in  std_logic;
    RST       : in  std_logic;
    CLK0      : out std_logic;
    CLK90     : out std_logic;
    CLK180    : out std_logic;
    CLK270    : out std_logic;
    CLK2X     : out std_logic;
    CLK2X180  : out std_logic;
    CLKDV     : out std_logic;
    CLKFX     : out std_logic;
    CLKFX180  : out std_logic;
    LOCKED    : out std_logic;
    PSDONE    : out std_logic;
    STATUS    : out std_logic_vector(7 downto 0)
);
end component;
-- Attributes
attribute DLL_FREQUENCY_MODE : string;
attribute DUTY_CYCLE_CORRECTION : string;
attribute STARTUP_WAIT : string;
attribute DLL_FREQUENCY_MODE of U_DCM: label is "LOW";
attribute DUTY_CYCLE_CORRECTION of U_DCM: label is "TRUE";
attribute STARTUP_WAIT of U_DCM: label is "FALSE";
-- Signal Declarations:
signal GND : std_logic;
signal CLK0_W: std_logic;
signal CLK1X_W: std_logic;
beg
GND <= '0';
CLK1X <= CLK1X_W;
-- DCM Instantiation
U_DCM: DCM
  port map (       
    CLKIN =>    CLK_IN,
    CLKFB =>    CLK1X_W,
    DSSEN =>    GND,
    PSINCDEC => GND,
    PSEN =>     GND,
    PSCLK =>    GND,
    RST =>      RST,
    CLK0 =>     CLK0_W,
    LOCKED =>   LOCK
  );
-- BUFG Instantiation
U_BUFG: BUFG
  port map (       
    I => CLK0_W,
    O => CLK1X_W
  );
end BUFG_CLK0_SUBM_arch;

Verilog Template

// Module: BUFG_CLK0_SUBM
// Description: Verilog Submodule
// DCM with CLK0 deskew
//
// Device: Virtex-II Pro Family
(Locale: //--

module BUFG_CLK0_SUBM (       
  CLK_IN,
  RST,
  CLK1X,
  LOCK
);

  input CLK_IN;
  input RST;

  output CLK1X;
  output LOCK;

  wire CLK0_W;
  wire GND;

  assign GND = 1'b0;
//BUFG Instantiation
//
BUFG U_BUFG
(.I(CLK0_W),
 .O(CLK1X)
);

// Attributes for functional simulation//
// synopsys translate_off
//synopsys translate_off
  defparam U_DCM.DLL_FREQUENCY_MODE = "LOW";
  defparam U_DCM.DUTY_CYCLE_CORRECTION = "TRUE";
  defparam U_DCM.STARTUP_WAIT = "FALSE";
// synopsys translate_on
// Instantiate the DCM primitive//
DCM U_DCM (  .CLKFB(CLK1X),
 .CLKin(CLK_IN),
 .DSSEN(GND),
 .PSCLK(GND),
 .PSEN(GND),
 .PSINCDEC(GND),
 .RST(RST),
 .CLK0(CLK0_W),
 .LOCKED(LOCK)
);

// synthesis attribute declarations
/* synopsys attribute
 DLL_FREQUENCY_MODE "LOW"
 DUTY_CYCLE_CORRECTION "TRUE"
 STARTUP_WAIT "FALSE"
 */
endmodule
DCM Waveforms

The DCM waveforms shown below are the results of functional simulation using Model Technology's ModelSim EE/Plus 5.3a_p1 simulator. Note that the time scale for these simulations were set to 1ns/1ps. It is important to set the unused inputs of the DCM to logic 0 and to set the attribute values to the correct data types. For example, the PHASE_SHIFT, CLKFX_DIVIDE, and CLKFX_MULTIPLY attributes are integers and should be set to values as shown.

```vhdl
defparam U_DCM.DFS_FREQUENCY_MODE = "LOW";
defparam U_DCM.CLKFX_DIVIDE = 1;  \(\text{(this value's range is specified under Frequency Synthesis in the Virtex-II Pro Data Sheet)}\)
defparam U_DCM.CLKFX_MULTIPLY = 4;  \(\text{(this value's range is specified under Frequency Synthesis in the Virtex-II Pro Data Sheet)}\)
defparam U_DCM.CLKOUT_PHASE_SHIFT = "FIXED";
defparam U_DCM.PHASE_SHIFT = 150;       \(\text{(Any value from 1 to 255)}\)
defparam U_DCM.STARTUP_WAIT = "FALSE";
```

The input clock, 'clk_in' (CLKIN input of DCM) in all these waveforms is 50 MHz. The DCM_DLL waveforms in Figure 3-38 shows four DCM outputs, namely, clk1x (CLK0 output of DCM), clk2x (CLK2X output of DCM), clk90 (CLK90 output of DCM), and clk180 (CLK180 output of DCM).

![DCM_DLL Waveforms](image.png)

**Figure 3-38:** DCM_DLL Waveforms
The DCM_DFS Waveforms in Figure 3-39 shows four DCM outputs namely, clk1x (CLK0 output of DCM), clk2x (CLK2X output of DCM), clkfx (CLKFX output of DCM), and clkfx180 (CLKFX180 output of DCM). In this case the attributes, CLKFX_DIVIDE = 1, and the CLKFX_MULTIPLY = 3.

Figure 3-39: DCM_DFS Waveforms
The DCM_DPS waveforms in Figure 3-40 shows four DCM outputs, namely, clk1x (CLK0 output of DCM), clk2x (CLK2X output of DCM), clk90 (CLK90 output of DCM), and clk180 (CLK180 output of DCM). In this case, the attribute PHASE_SHIFT = 150 which translates to a phase shift of \((150 \times 20 \text{ ns})/256 = 11.719 \text{ ns}\), where 20 ns is the clock period.

![Figure 3-40: DCM_DPS Waveforms](image-url)
The DCM_DPS_DFS waveforms in Figure 3-41 shows four DCM outputs namely, clk1x (CLK0 output of DCM), clk90 (CLK90 output of DCM), clkfx (CLKFX output of DCM), and clkfx180 (CLKFX180 output of DCM). In this case, the attributes, CLKFX_DIVIDE = 1, and the CLKFX_MULTIPLY = 4. The attribute, PHASE_SHIFT = 150 which translates to a phase shift of \( (150 \times 20 \text{ ns})/256 = 11.719 \text{ ns} \), where 20 ns is the clock period.

![DCM_DPS_DFS Waveforms](ug002_c2_098_113000)

**Figure 3-41:** DCM_DPS_DFS Waveforms
Block SelectRAM™ Memory

Introduction

In addition to distributed SelectRAM memory, Virtex-II Pro / Virtex-II Pro X devices feature a large number of 18 Kb block SelectRAM memories. The block SelectRAM memory is a True Dual-Port™ RAM, offering fast, discrete, and large blocks of memory in the device. The memory is organized in columns, and the total amount of block SelectRAM memory depends on the size of the Virtex-II Pro / Virtex-II Pro X device. The 18 Kb blocks are cascadable to enable a deeper and wider memory implementation, with a minimal timing penalty incurred through specialized routing resources.

Embedded dual- or single-port RAM modules, ROM modules, synchronous FIFOs, and data width converters are easily implemented using the Xilinx CORE Generator “Block Memory” modules. Asynchronous FIFOs can be generated using the CORE Generator Asynchronous FIFO and "Block Memory" module. Starting with IP Update #3, the designer can also generate synchronous FIFOs using Block Memory.

Synchronous Dual-Port and Single-Port RAM

Data Flow

The 18Kb block SelectRAM dual-port memory consists of an 18 Kb storage area and two completely independent access ports, A and B. The structure is fully symmetrical, and both ports are interchangeable.

Data can be written to either port and can be read from the same or the other port. Each port is synchronous, with its own clock, clock enable, and write enable. Note that the read operation is also synchronous and requires a clock edge.

![Dual-Port Data Flows](image-url)
As described below, there are three options for the behavior of the data output during a write operation on its port. There is no dedicated monitor to arbitrate the result of identical addresses on both ports. It is up to the user to time the two clocks appropriately. However, conflicting simultaneous writes to the same location never cause any physical damage.

Operating Modes

To maximize utilization of the True Dual-Port memory at each clock edge, the block SelectRAM memory supports three different write modes for each port. The “read during write” mode offers the flexibility of using the data output bus during a write operation on the same port. Output behavior is determined by the configuration. This choice increases the efficiency of block SelectRAM memory at each clock cycle and allows designs that use maximum bandwidth.

Read Operation

The read operation uses one clock edge. The read address is registered on the read port, and the stored data is loaded into the output latches after the RAM access interval passes.

Write Operations

A write operation is a single clock-edge operation. The write address is registered on the write port, and the data input is stored in memory.

Three different modes are used to determine data available on the output latches after a write clock edge.

WRITE_FIRST or Transparent Mode (Default)

In WRITE_FIRST mode, the input data is simultaneously written into memory and stored in the data output (transparent write), as shown in Figure 3-43.

![WRITE_FIRST Mode Waveforms](DS031_19_040400)
READ_FIRST or Read-Before-Write Mode

In READ_FIRST mode, data previously stored at the write address appears on the output latches, while the input data is being stored in memory (read before write). See Figure 3-44.

```
Figure 3-44: READ_FIRST Mode Waveforms
```

NO_CHANGE Mode

In NO_CHANGE mode, the output latches remain unchanged during a write operation. As shown in Figure 3-45, data output is still the last read data and is unaffected by a write operation on the same port.

Mode selection is set by configuration. One of these three modes is set individually for each port by an attribute. The default mode is WRITE_FIRST.

```
Figure 3-45: NO_CHANGE Mode Waveforms
```

Conflict Resolution

Virtex-II Pro / Virtex-II Pro X block SelectRAM memory is a True Dual-Port RAM that allows both ports to simultaneously access the same memory cell. When one port writes to a given memory cell, the other port must not address that memory cell (for a write or a
read) within the clock-to-clock setup window. Figure 3-46 describes this asynchronous operation.

Asynchronous Clocks

The first CLK_A clock edge violates the clock-to-clock setup parameter, because it occurs too soon after the last CLK_B clock edge. The write operation on port B is valid, and the read operation on port A is invalid.

At the second rising edge of the CLK_B pin, the write operation is valid. The memory location (bb) contains 4444. The second rising edge of CLK_A reads the new data at the same location (bb), which now contains 4444.

The clock-to-clock setup timing parameter is specified together with other block SelectRAM switching characteristics in the Data Sheet.

Synchronous Clocks

When both clocks are synchronous or identical, the result of simultaneous accesses from both ports to the same memory cell is best described in words:

- If both ports read simultaneously from the same memory cell:
  Both Data_out ports will have the same data.

- If both ports write simultaneously into the same memory cell:
  The data stored in that cell becomes invalid (unless both ports write identical data).

- If one port writes and the other port reads from the same memory cell:
  The write operation succeeds, and the data to be read out from the read port will be determined by the read output mode (write_first, read_first, or no_change).

  If the write port is in read_first mode, the read port’s Data_out represents the previous content of the memory cell. If the write port is in write_first mode or in no_change
mode, the read port’s Data_out becomes invalid. Obviously, the read port’s mode setting does not affect this operation.

Characteristics

- A write operation requires only one clock edge.
- A read operation requires only one clock edge.
- All inputs are registered with the port clock and have a setup-to-clock timing specification.
- All outputs have a read-through function or one of three read-during-write functions, depending on the state of the WE pin. The outputs relative to the port clock are available after the clock-to-out timing interval.
- Block SelectRAM cells are true synchronous RAM memories and do not have a combinatorial path from the address to the output.
- The ports are completely independent of each other (that is, clocking, control, address, read/write functions, initialization, and data width) without arbitration.
- Output ports are latched with a self-timed circuit, guaranteeing glitch-free reads. The state of the output port does not change until the port executes another read or write operation.
- Data input and output signals are always buses; that is, in a 1-bit width configuration, the data input signal is DI[0] and the data output signal is DO[0].

Library Primitives

The input and output data buses are represented by two buses for 9-bit width (8+1), 18-bit width (16+2), and 36-bit width (32+4) configurations. The ninth bit associated with each byte can store parity or error correction bits. No specific function is performed on this bit. The separate bus for parity bits facilitates some designs. However, other designs safely use a 9-bit, 18-bit, or 36-bit bus by merging the regular data bus with the parity bus. Read/write and storage operations are identical for all bits, including the parity bits.
Figure 3-47 shows the generic dual-port block RAM primitive. DIA, DIPA, ADDRA, DOA, DOPA, and the corresponding signals on port B are buses.

Figure 3-47: Dual-Port Block RAM Primitive
Table 3-10 lists the available dual-port primitives for synthesis and simulation.

**Table 3-10: Dual-Port Block RAM Primitives**

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Port A Width</th>
<th>Port B Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAMB16_S1_S1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>RAMB16_S1_S2</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S1_S4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S1_S9</td>
<td>(8+1)</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S1_S18</td>
<td>(16+2)</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S1_S36</td>
<td>(32+4)</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S2_S2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>RAMB16_S2_S4</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S2_S9</td>
<td>(8+1)</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S2_S18</td>
<td>(16+2)</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S2_S36</td>
<td>(32+4)</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S4_S4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>RAMB16_S4_S9</td>
<td>(8+1)</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S4_S18</td>
<td>(16+2)</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S4_S36</td>
<td>(32+4)</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S9_S9</td>
<td>(8+1)</td>
<td>(8+1)</td>
</tr>
<tr>
<td>RAMB16_S9_S18</td>
<td>(16+2)</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S9_S36</td>
<td>(32+4)</td>
<td></td>
</tr>
<tr>
<td>RAMB16_S18_S18</td>
<td>(16+2)</td>
<td>(16+2)</td>
</tr>
<tr>
<td>RAMB16_S18_S36</td>
<td></td>
<td>(32+4)</td>
</tr>
<tr>
<td>RAMB16_S36_S36</td>
<td></td>
<td>(32+4)</td>
</tr>
</tbody>
</table>

Figure 3-48 shows the generic single-port block RAM primitive. DI, DIP, ADDR, DO, and DOP are buses.
Table 3-11 lists all of the available single-port primitives for synthesis and simulation.

### Table 3-11: Single-Port Block RAM Primitives

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Port Width</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAMB16_S1</td>
<td>1</td>
</tr>
<tr>
<td>RAMB16_S2</td>
<td>2</td>
</tr>
<tr>
<td>RAMB16_S4</td>
<td>4</td>
</tr>
<tr>
<td>RAMB16_S9</td>
<td>(8+1)</td>
</tr>
<tr>
<td>RAMB16_S18</td>
<td>(16+2)</td>
</tr>
<tr>
<td>RAMB16_S36</td>
<td>(32+4)</td>
</tr>
</tbody>
</table>

**VHDL and Verilog Instantiation**

VHDL and Verilog instantiation templates are available as examples (“VHDL and Verilog Templates,” page 134).

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

The SelectRAM_Ax templates (with \( x = 1, 2, 4, 9, 18, \) or 36) are single-port modules and instantiate the corresponding RAMB16_Sx module.

SelectRAM_Ax_By templates (with \( x = 1, 2, 4, 9, 18, \) or 36 and \( y = 1, 2, 4, 9, 18, \) or 36) are dual-port modules and instantiate the corresponding RAMB16_Sx_Sy module.

**Port Signals**

Each block SelectRAM port operates independently of the other while accessing the same set of 18K-bit memory cells.

**Clock - CLK[A|B]**

Each port is fully synchronous with independent clock pins. All port input pins have setup time referenced to the port CLK pin. The data bus has a clock-to-out time referenced to the CLK pin. Clock polarity is configurable (rising edge by default).

**Enable - EN[A|B]**

The enable pin affects the read, write, and set/reset functionality of the port. Ports with an inactive enable pin keep the output pins in the previous state and do not write data to the memory cells. Enable polarity is configurable (active High by default).

**Write Enable - WE[A|B]**

Both EN and WE are active when the contents of the data input bus is written to memory at the address pointed to by the address bus. The output latches are loaded or not loaded according to the write configuration (WRITE_FIRST, READ_FIRST, NO_CHANGE). When inactive, a read operation occurs, and the contents of the memory cells referenced by the address bus reflect on the data-out bus, regardless of the write mode attribute. Write enable polarity is configurable (active High by default).
Set/Reset - SSR[A|B]

The SSR pin forces the data output latches to contain the value “SRVAL” (see “Attributes,” page 131). The data output latches are synchronously asserted to 0 or 1, including the parity bit. In a 36-bit width configuration, each port has an independent SRVAL[A | B] attribute of 36 bits. This operation does not affect RAM memory cells and does not disturb write operations on the other port. Like the read and write operation, the set/reset function is active only when the enable pin of the port is active. Set/reset polarity is configurable (active High by default).

Address Bus - ADDR[A|B]<#:0>

The address bus selects the memory cells for read or write. The width of the port determines the required address bus width, as shown in Table 3-12.

Table 3-12: Port Aspect Ratio

<table>
<thead>
<tr>
<th>Port Data Width</th>
<th>Depth</th>
<th>ADDR Bus</th>
<th>DI Bus / DO Bus</th>
<th>DIP Bus / DOP Bus</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>16,384</td>
<td>&lt;13:0&gt;</td>
<td>&lt;0&gt;</td>
<td>NA</td>
</tr>
<tr>
<td>2</td>
<td>8,192</td>
<td>&lt;12:0&gt;</td>
<td>&lt;1:0&gt;</td>
<td>NA</td>
</tr>
<tr>
<td>4</td>
<td>4,096</td>
<td>&lt;11:0&gt;</td>
<td>&lt;3:0&gt;</td>
<td>NA</td>
</tr>
<tr>
<td>9</td>
<td>2,048</td>
<td>&lt;10:0&gt;</td>
<td>&lt;7:0&gt;</td>
<td>&lt;0&gt;</td>
</tr>
<tr>
<td>18</td>
<td>1,024</td>
<td>&lt;9:0&gt;</td>
<td>&lt;15:0&gt;</td>
<td>&lt;1:0&gt;</td>
</tr>
<tr>
<td>36</td>
<td>512</td>
<td>&lt;8:0&gt;</td>
<td>&lt;31:0&gt;</td>
<td>&lt;3:0&gt;</td>
</tr>
</tbody>
</table>

Data-In Buses - DI[A|B]<#:0> & DIP[A|B]<#:0>

Data-in buses provide the new data value to be written into RAM. The regular data-in bus (DI) and the parity data-in bus (when available) have a total width equal to the port width. For example the 36-bit port data width is represented by DI<31:0> and DIP<3:0>, as shown in Table 3-12.

Data-Out Buses - DO[A|B]<#:0> & DOP[A|B]<#:0>

Data-out buses reflect the contents of memory cells referenced by the address bus at the last active clock edge during a read operation. During a write operation (WRITE_FIRST or READ_FIRST configuration), the data-out buses reflect either the data-in buses or the stored value before write. During a write operation in NO_CHANGE mode, data-out buses are not affected. The regular data-out bus (DO) and the parity data-out bus (DOP) (when available) have a total width equal to the port width, as shown in Table 3-12.

Inverting Control Pins

For each port, the four control pins (CLK, EN, WE, and SSR) each have an individual inversion option. Any control signal can be configured as active High or Low, and the clock can be active on a rising or falling edge (active High on rising edge by default) without requiring other logic resources.

Unused Inputs

Non-connected Data and/or address inputs should be connected to logic “1”.


GSR

The global set/reset (GSR) signal of a Virtex-II Pro / Virtex-II Pro X device is an asynchronous global signal that is active at the end of device configuration. The GSR can also restore the initial state at any time. The GSR signal initializes the output latches to the INIT, or to the INIT_A and INIT_B value (see “Attributes”). A GSR signal has no impact on internal memory contents. Because it is a global signal, the GSR has no input pin at the functional level (block SelectRAM primitive).

Address Mapping

Each port accesses the same set of 18,432 memory cells using an addressing scheme dependent on the width of the port. The physical RAM locations addressed for a particular width are determined using the following formula (of interest only when the two ports use different aspect ratios):

\[
\text{END} = ((\text{ADDR} + 1) \times \text{Width}) - 1 \quad \text{START} = \text{ADDR} \times \text{Width}
\]

Table 3-13 shows low-order address mapping for each port width.

<table>
<thead>
<tr>
<th>Port Width</th>
<th>Parity Locations</th>
<th>Data Locations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>N.A.</td>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>2</td>
<td>15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>7 6 5 4 3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>8 + 1</td>
<td>3 2 1 0</td>
<td></td>
</tr>
<tr>
<td>16 + 2</td>
<td>1 0</td>
<td></td>
</tr>
<tr>
<td>32 + 4</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Attributes

Content Initialization - INIT_xx

INIT_xx attributes define the initial memory contents. By default block SelectRAM memory is initialized with all zeros during the device configuration sequence. The 64 initialization attributes from INIT_00 through INIT_3F represent the regular memory contents. Each INIT_xx is a 64-digit hex-encoded bit vector. The memory contents can be partially initialized and are automatically completed with zeros.

The following formula is used for determining the bit positions for each INIT_xx attribute. Given \( y = \) conversion hex-encoded to decimal (xx), INIT_xx corresponds to the memory cells as follows:

- from \([y + 1) \times 256]\) -1
- to \((y) \times 256\)

For example, for the attribute INIT_1F, the conversion is as follows:

- \(y = \) conversion hex-encoded to decimal X”1F” = 31
- from \([31+1) \times 256]\) -1 = 8191
- to 31 * 256 = 7936
More examples are given in Table 3-14.

Table 3-14: Block SelectRAM Initialization Attributes

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Memory Cell</th>
</tr>
</thead>
<tbody>
<tr>
<td>INIT_00</td>
<td>255 - 0</td>
</tr>
<tr>
<td>INIT_01</td>
<td>511 - 256</td>
</tr>
<tr>
<td>INIT_02</td>
<td>767 - 512</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>INIT_0E</td>
<td>3839 - 3584</td>
</tr>
<tr>
<td>INIT_0F</td>
<td>4095 - 3840</td>
</tr>
<tr>
<td>INIT_10</td>
<td>4351 - 4096</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>INIT_1F</td>
<td>8191 - 7936</td>
</tr>
<tr>
<td>INIT_20</td>
<td>8447 - 8192</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>INIT_2F</td>
<td>12287 - 12032</td>
</tr>
<tr>
<td>INIT_30</td>
<td>12543 - 12288</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>INIT_3F</td>
<td>16383 - 16128</td>
</tr>
</tbody>
</table>

Content Initialization - INITP_xx

INITP_xx attributes define the initial contents of the memory cells corresponding to DIP/DOP buses (parity bits). By default these memory cells are also initialized to all zeros. The eight initialization attributes from INITP_00 through INITP_07 represent the memory contents of parity bits. Each INITP_xx is a 64-digit hex-encoded bit vector and behaves like a regular INIT_xx attribute. The same formula can be used to calculate the bit positions initialized by a particular INITP_xx attribute.

Output Latches Initialization - INIT (INIT_A & INIT_B)

The INIT (single-port) or INIT_A and INIT_B (dual-port) attributes define the output latches values after configuration. The width of the INIT (INIT_A & INIT_B) attribute is the port width, as shown in Table 3-15. These attributes are hex-encoded bit vectors and the default value is 0.

Output Latches Synchronous Set/Reset - SRVAL (SRVAL_A & SRVAL_B)

The SRVAL (single-port) or SRVAL_A and SRVAL_B (dual-port) attributes define output latch values when the SSR input is asserted. The width of the SRVAL (SRVAL_A and SRVAL_B) attribute is the port width, as shown in Table 3-15. These attributes are hex-encoded bit vectors and the default value is 0.
Initialization in VHDL or Verilog Codes

Block SelectRAM memory structures can be initialized in VHDL or Verilog code for both synthesis and simulation. For synthesis, the attributes are attached to the block SelectRAM instantiation and are copied in the EDIF output file to be compiled by Xilinx Alliance Series™ tools. The VHDL code simulation uses a generic parameter to pass the attributes. The Verilog code simulation uses a defparam parameter to pass the attributes.

The XC2V_RAMB_1_PORT block SelectRAM instantiation code examples (in VHDL and Verilog) illustrate these techniques (see “VHDL and Verilog Templates,” page 134).

Location Constraints

Block SelectRAM instances can have LOC properties attached to them to constrain placement. Block SelectRAM placement locations differ from the convention used for naming CLB locations, allowing LOC properties to transfer easily from array to array.

The LOC properties use the following form:

LOC = RAMB16_X#Y#

The RAMB16_X0Y0 is the bottom-left block SelectRAM location on the device.

Applications

Creating Larger RAM Structures

Block SelectRAM columns have specialized routing to allow cascading blocks with minimal routing delays. Wider or deeper RAM structures are achieved with a smaller timing penalty than is encountered when using normal routing resources.

The CORE Generator program offers the designer a painless way to generate wider and deeper memory structures using multiple block SelectRAM instances. This program outputs VHDL or Verilog instantiation templates and simulation models, along with an EDIF file for inclusion in a design.

Multiple RAM Organizations

The flexibility of block SelectRAM memories allows designs with various types of RAM in addition to regular configurations. Application notes at www.xilinx.com describe some of these designs, with VHDL and Verilog reference designs included:

- **XAPP258, FIFOs Using Virtex-II Block RAM**
- **XAPP260, Using Virtex-II Block RAM for High Performance Read/Write CAMs**

### Table 3-15: Port Width Values

<table>
<thead>
<tr>
<th>Port Data Width</th>
<th>DOP Bus</th>
<th>DO Bus</th>
<th>INIT / SRVAL</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>NA</td>
<td>&lt;0&gt;</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>NA</td>
<td>&lt;1:0&gt;</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>NA</td>
<td>&lt;3:0&gt;</td>
<td>4</td>
</tr>
<tr>
<td>9</td>
<td>&lt;0&gt;</td>
<td>&lt;7:0&gt;</td>
<td>(1+8) = 9</td>
</tr>
<tr>
<td>18</td>
<td>&lt;1:0&gt;</td>
<td>&lt;15:0&gt;</td>
<td>(2+16) = 18</td>
</tr>
<tr>
<td>36</td>
<td>&lt;3:0&gt;</td>
<td>&lt;31:0&gt;</td>
<td>(4 + 32) = 36</td>
</tr>
</tbody>
</table>
VHDL and Verilog Templates

VHDL and Verilog templates are available in the Libraries Guide for all single-port and dual-port primitives. The A and B numbers indicate the width of the ports.

The following are single-port templates:

- SelectRAM_A1
- SelectRAM_A2
- SelectRAM_A4
- SelectRAM_A9
- SelectRAM_A18
- SelectRAM_A36

The following are dual-port templates:

- SelectRAM_A1_B1
- SelectRAM_A1_B2
- SelectRAM_A1_B4
- SelectRAM_A1_B9
- SelectRAM_A1_B18
- SelectRAM_A1_B36
- SelectRAM_A2_B2
- SelectRAM_A2_B4
- SelectRAM_A2_B9
- SelectRAM_A2_B18
- SelectRAM_A2_B36
- SelectRAM_A4_B4
- SelectRAM_A4_B9
- SelectRAM_A4_B18
- SelectRAM_A4_B36
- SelectRAM_A9_B9
- SelectRAM_A9_B18
- SelectRAM_A9_B36
- SelectRAM_A18_B18
- SelectRAM_A18_B36
- SelectRAM_A36_B36

VHDL Template

As an example, the XC2V_RAMB_1_PORT.vhd file uses the SelectRAM_A36 template:

```vhdl
-- Module: XC2V_RAMB_1_PORT
-- Description: 18Kb Block SelectRAM example
-- Single Port 512 x 36 bits
-- Use template "SelectRAM_A36.vhd"
--
-- Device: Virtex-II Pro Family
--------------------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
```
-- Syntax for Synopsys FPGA Express
-- pragma translate_off
library UNISIM;
use UNISIM.VCOMPONENTS.ALL;
-- pragma translate_on
--
entity XC2V_RAMB_1_PORT is
  port (
    DATA_IN : in std_logic_vector (35 downto 0);
    ADDRESS : in std_logic_vector (8 downto 0);
    ENABLE: in std_logic;
    WRITE_EN     : in std_logic;
    SET_RESET    : in std_logic;
    CLK    : in std_logic;
    DATA_OUT     : out std_logic_vector (35 downto 0)
  );
end XC2V_RAMB_1_PORT;
--
arquitectura XC2V_RAMB_1_PORT_arch of XC2V_RAMB_1_PORT is
--
-- Components Declaration:
--
component BUFG
  port ( 
    I: in std_logic;
    O: out std_logic
  );
end component;
--
-- Syntax for Synopsys FPGA Express
component RAMB16_S36
-- pragma translate_off
generic ( 
  -- "Read during Write" attribute for functional simulation
  WRITE_MODE : string := "READ_FIRST" ; -- WRITE_FIRST(default)/
  READ_FIRST/ NO_CHANGE
  -- Output value after configuration
  INIT : bit_vector(35 downto 0) := X"000000000";
  -- Output value if SSR active
  SRVAL : bit_vector(35 downto 0) := X"012345678";
  -- Plus bits initial content
  INITP_00 : bit_vector(255 downto 0) := X"0000000000000000000000000000000000000000000000000000000000000000";
  INITP_01 : bit_vector(255 downto 0) := X"0000000000000000000000000000000000000000000000000000000000000000";
  INITP_02 : bit_vector(255 downto 0) := X"0000000000000000000000000000000000000000000000000000000000000000";
  INITP_03 : bit_vector(255 downto 0) := X"0000000000000000000000000000000000000000000000000000000000000000";
  INITP_04 : bit_vector(255 downto 0) := X"0000000000000000000000000000000000000000000000000000000000000000";
  INITP_05 : bit_vector(255 downto 0) := X"0000000000000000000000000000000000000000000000000000000000000000";
  INITP_06 : bit_vector(255 downto 0) := X"0000000000000000000000000000000000000000000000000000000000000000";
  INITP_07 : bit_vector(255 downto 0) := X"0000000000000000000000000000000000000000000000000000000000000000";

-- Regular bits initial content
INIT_00 : bit_vector (255 downto 0) :=
X"000000000000000000000000000000000000000000000000FEDCBA9876543210";
INIT_01 : bit_vector (255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_02 : bit_vector (255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000";
... (cut)
INIT_3E : bit_vector (255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000";
INIT_3F : bit_vector (255 downto 0) :=
X"0000000000000000000000000000000000000000000000000000000000000000"
);
-- pragma translate_on
port {
  DI     : in std_logic_vector (31 downto 0);
  DIP    : in std_logic_vector (3 downto 0);
  ADDR   : in std_logic_vector (8 downto 0);
  EN     : in STD_LOGIC;
  WE     : in STD_LOGIC;
  SSR    : in STD_LOGIC;
  CLK    : in STD_LOGIC;
  DO     : out std_logic_vector (31 downto 0);
  DOP    : out std_logic_vector (3 downto 0)
};
end component;

-- Attribute Declarations:
attribute WRITE_MODE : string;
attribute INIT: string;
attribute SRVAL: string;
--
attribute INITP_00: string;
attribute INITP_01: string;
attribute INITP_02: string;
attribute INITP_03: string;
attribute INITP_04: string;
attribute INITP_05: string;
attribute INITP_06: string;
attribute INITP_07: string;
--
attribute INIT_00: string;
attribute INIT_01: string;
attribute INIT_02: string;
... (cut)
attribute INIT_3E: string;
attribute INIT_3F: string;
--
-- Attribute "Read during Write mode" = WRITE_FIRST(default)/READ_FIRST/NO_CHANGE
attribute WRITE_MODE of U_RAMB16_S36: label is "READ_FIRST";
attribute INIT of U_RAMB16_S36: label is "00000000000";
attribute SRVAL of U_RAMB16_S36: label is "012345678";
--
-- RAMB16 memory initialization for Alliance
-- Default value is "0" / Partial initialization strings are padded
-- with zeros to the left
attribute INITP_00 of U_RAMB16_S36: label is
*0000000000000000000000000000000000000000000000000000000000000000*;
attribute INITP_01 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INITP_02 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INITP_03 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INITP_04 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INITP_05 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INITP_06 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INITP_07 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
--
attribute INIT_00 of U_RAMB16_S36: label is
"000000000000000000000000000000000000000000000000FEDCBA9876543210";
attribute INIT_01 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INIT_02 of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
... (cut)
attribute INIT_3E of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
attribute INIT_3F of U_RAMB16_S36: label is
"0000000000000000000000000000000000000000000000000000000000000000";
--
-- Signal Declarations:
--
-- signal VCC : std_logic;
-- signal GND : std_logic;
signal CLK_BUFG: std_logic;
signal INV_SET_RESET : std_logic;
--
begin
-- VCC <= '1';
-- GND <= '0';
--
-- Instantiate the clock Buffer
U_BUFG: BUFG
port map (
I => CLK,
O => CLK_BUFG
);
--
-- Use of the free inverter on SSR pin
INV_SET_RESET <= NOT SET_RESET;

-- Block SelectRAM Instantiation
U_RAMB16_S36: RAMB16_S36
port map (
DI  => DATA_IN (31 downto 0), -- insert 32 bits data-in bus
(31 downto 0)>
DIP => DATA_IN (35 downto 32), -- insert 4 bits parity data-
in bus (or <35 downto 32>)
ADDR => ADDRESS (8 downto 0), -- insert 9 bits address bus
EN  => ENABLE, -- insert enable signal
WE  => WRITE_EN, -- insert write enable signal
SSR    => INV_SET_RESET, -- insert set/reset signal
CLK    => CLK_BUFG, -- insert clock signal
DO     => DATA_OUT (31 downto 0), -- insert 32 bits data-out bus
(<31 downto 0>)
DOP    => DATA_OUT (35 downto 32) -- insert 4 bits parity data-out bus (or <35 downto 32>)
}
--
end XC2V_RAMB_1_PORT_arch;
---------------------------------------------------------------

Verilog Template

// Module: XC2V_RAMB_1_PORT
// Description: 18Kb Block SelectRAM-II example
// Single Port 512 x 36 bits
// Use template "SelectRAM_A36.v"
//
// Device: Virtex-II Pro Family
//---------------------------------------------------------------

module XC2V_RAMB_1_PORT (CLK, SET_RESET, ENABLE, WRITE_EN, ADDRESS, DATA_IN, DATA_OUT);

input CLK, SET_RESET, ENABLE, WRITE_EN;
inout [35:0] DATA_IN;
in [8:0] ADDRESS;
output [35:0] DATA_OUT;
wire CLK_BUFG, INV_SET_RESET;
//Use of the free inverter on SSR pin
assign INV_SET_RESET = ~SET_RESET;

// initialize block ram for simulation
// synopsys translate_off
defparam
  //"Read during Write" attribute for functional simulation
  U_RAMB16_S36.WRITE_MODE = "READ_FIRST", //WRITE_FIRST(default)/READ_FIRST/ NO_CHANGE
  //Output value after configuration
  U_RAMB16_S36.INIT = 36'h000000000,
  //Output value if SSR active
  U_RAMB16_S36.SRVAL = 36'h012345678,
  //Plus bits initial content
  U_RAMB16_S36.INITP_00 =
    256'h0123456789ABCDEF000000000000000000000000000000000000000000000000000,
  U_RAMB16_S36.INITP_01 =
    256'h0000000000000000000000000000000000000000000000000000000000000000,
  U_RAMB16_S36.INITP_02 =
    256'h0000000000000000000000000000000000000000000000000000000000000000,
  U_RAMB16_S36.INITP_03 =
    256'h0000000000000000000000000000000000000000000000000000000000000000,
  U_RAMB16_S36.INITP_04 =
    256'h0000000000000000000000000000000000000000000000000000000000000000,
Block SelectRAM™ Memory

U_RAMB16_S36.INITP_05 =
256’h0000000000000000000000000000000000000000000000000000000000000000,
U_RAMB16_S36.INITP_06 =
256’h0000000000000000000000000000000000000000000000000000000000000000,
U_RAMB16_S36.INITP_07 =
256’h0000000000000000000000000000000000000000000000000000000000000000,

//Regular bits initial content
U_RAMB16_S36.INIT_00 =
256’h0123456789ABCDEF0000000000000000000000000000000000000000000000,
U_RAMB16_S36.INIT_01 =
256’h0000000000000000000000000000000000000000000000000000000000000000,
U_RAMB16_S36.INIT_02 =
256’h0000000000000000000000000000000000000000000000000000000000000000,
...<cut>
U_RAMB16_S36.INIT_3E =
256’h0000000000000000000000000000000000000000000000000000000000000000,
U_RAMB16_S36.INIT_3F =
256’h0000000000000000000000000000000000000000000000000000000000000000;
// synopsys translate_on

//Instantiate the clock Buffer
BUFG U_BUF ( .I(CLK), .O(CLK_BUFG));

//Block SelectRAM Instantiation
RAMB16_S36 U_RAMB16_S36 ( .DI(DATA_IN[31:0]),
   .DIP(DATA_IN-PARITY[35:32]),
   .ADDR(ADDRESS),
   .EN(ENABLE),
   .WE(WRITE_EN),
   .SSR(INV_SET_RESET),
   .CLK(CLK_BUFG),
   .DO(DATA_OUT[31:0]),
   .DOP(DATA_OUT-PARITY[35:32]));

// synthesis attribute declarations
/* synopsys attribute
WRITE_MODE "READ_FIRST"
INIT "000000000"
SRVAL "012345678"
*/

INITP_00
*0123456789ABCDEF000000000000000000000000000000000000000000000000000000,
INITP_01
*0000000000000000000000000000000000000000000000000000000000000000,
INITP_02
*0000000000000000000000000000000000000000000000000000000000000000,
INITP_03
*0000000000000000000000000000000000000000000000000000000000000000,
INITP_04
*0000000000000000000000000000000000000000000000000000000000000000,
INITP_05
*0000000000000000000000000000000000000000000000000000000000000000,
INITP_06
*0000000000000000000000000000000000000000000000000000000000000000,
INITP_07
*0000000000000000000000000000000000000000000000000000000000000000,
Distributed SelectRAM Memory

Introduction

In addition to 18Kb SelectRAM blocks, Virtex-II Pro / Virtex-II Pro X devices feature distributed SelectRAM modules. Each function generator or LUT of a CLB resource can implement a 16 x 1-bit synchronous RAM resource. Distributed SelectRAM memory writes synchronously and reads asynchronously. However, a synchronous read can be implemented using the register that is available in the same slice. This 16 x 1-bit RAM is cascadable for a deeper and/or wider memory implementation, with a minimal timing penalty incurred through specialized logic resources.

Distributed SelectRAM modules up to a size of 128 x 1 are available as primitives. Two 16 x 1 RAM resources can be combined to form a dual-port 16 x 1 RAM with one dedicated read/write port and a second read-only port. One port writes into both 16 x1 RAMs simultaneously, but the second port reads independently.

This section provides generic VHDL and Verilog reference code examples implementing n-bit-wide single-port and dual-port distributed SelectRAM memory.

Distributed SelectRAM memory enables many high-speed applications that require relatively small embedded RAM blocks, such as FIFOs, which are close to the logic that uses them.

Virtex-II Pro / Virtex-II Pro X Distributed SelectRAM memories can be generated using the CORE Generator Distributed Memory module (V2.0 or later). The user can also generate Distributed RAM-based Asynchronous and Synchronous FIFOs using the CORE Generator.

Single-Port and Dual-Port RAM

Data Flow

Distributed SelectRAM memory supports the following:

- Single-port RAM with synchronous write and asynchronous read
- Dual-port RAM with one synchronous write and two asynchronous read ports
As illustrated in the Figure 3-49, the dual port has one read/write port and an independent read port.

**Figure 3-49: Single-Port and Dual-Port Distributed SelectRAM**

Any read/write operation can occur simultaneously with and independently of a read operation on the other port.

**Write Operations**

The write operation is a single clock-edge operation, with a write enable that is active High by default. When the write enable is Low, no data is written into the RAM. When the write enable is High, the clock edge latches the write address and writes the data on D into the RAM.

**Read Operation**

The read operation is a combinatorial operation. The address port (single or dual port) is asynchronous with an access time equivalent to the logic delay.

**Read During Write**

When new data is synchronously written, the output reflects the data in the memory cell addressed (transparent mode). The timing diagram in Figure 3-50 illustrates a write operation, with the previous data read on the output port, before the clock edge and then the new data.
Characteristics

- A write operation requires only one clock edge.
- A read operation requires only the logic access time.
- Outputs are asynchronous and dependent only on the logic delay.
- Data and address inputs are latched with the write clock and have a setup-to-clock timing specification. There is no hold time requirement.
- For dual-port RAM, one address is the write and read address, the other address is an independent read address.

Library Primitives

Seven library primitives from 16 x 1-bit to 128 x 1-bit are available. Four primitives are single-port RAM and three primitives are True Dual-Port RAM, as shown in Table 3-16.

Table 3-16: Single-Port and Dual-Port Distributed SelectRAM

<table>
<thead>
<tr>
<th>Primitive</th>
<th>RAM Size</th>
<th>Type</th>
<th>Address Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM16X1S</td>
<td>16 bits</td>
<td>single-port</td>
<td>A3, A2, A1, A0</td>
</tr>
<tr>
<td>RAM32X1S</td>
<td>32 bits</td>
<td>single-port</td>
<td>A4, A3, A2, A1, A0</td>
</tr>
<tr>
<td>RAM64X1S</td>
<td>64 bits</td>
<td>single-port</td>
<td>A5, A4, A3, A2, A1, A0</td>
</tr>
<tr>
<td>RAM128X1S</td>
<td>128 bits</td>
<td>single-port</td>
<td>A6, A5, A4, A3, A2, A1, A0</td>
</tr>
<tr>
<td>RAM16X1D</td>
<td>16 bits</td>
<td>dual-port</td>
<td>A3, A2, A1, A0</td>
</tr>
<tr>
<td>RAM32X1D</td>
<td>32 bits</td>
<td>dual-port</td>
<td>A4, A3, A2, A1, A0</td>
</tr>
<tr>
<td>RAM64X1D</td>
<td>64 bits</td>
<td>dual-port</td>
<td>A5, A4, A3, A2, A1, A0</td>
</tr>
</tbody>
</table>

The input and output data are 1-bit wide. However, several distributed SelectRAM memories can be used to implement wide memory blocks.
Figure 3-51 shows generic single-port and dual-port distributed SelectRAM primitives. The A and DPRA signals are address buses.

![Figure 3-51: Single-Port and Dual-Port Distributed SelectRAM Primitive](image)

As shown in Table 3-17, wider library primitives are available for 2-bit, 4-bit, and 8-bit RAM.

<table>
<thead>
<tr>
<th>Table 3-17: Wider Library Primitives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primitve</td>
</tr>
<tr>
<td>------------</td>
</tr>
<tr>
<td>RAM16x2S</td>
</tr>
<tr>
<td>RAM32x2S</td>
</tr>
<tr>
<td>RAM64x2S</td>
</tr>
<tr>
<td>RAM16x4S</td>
</tr>
<tr>
<td>RAM32x4S</td>
</tr>
<tr>
<td>RAM16x8S</td>
</tr>
<tr>
<td>RAM32x8S</td>
</tr>
</tbody>
</table>

VHDL and Verilog Instantiation

VHDL and Verilog instantiations templates are available as examples (see “VHDL and Verilog Templates,” page 148).

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

The SelectRAM_xS templates (with x = 16, 32, 64, or 128) are single-port modules and instantiate the corresponding RAMxXIS primitive.

SelectRAM_xD templates (with x = 16, 32, or 64) are dual-port modules and instantiate the corresponding RAMxXID primitive.
Ports Signals

Each distributed SelectRAM port operates independently of the other while reading the same set of memory cells.

Clock - WCLK

The clock is used for the synchronous write. The data and the address input pins have setup time referenced to the WCLK pin.

Enable - WE

The enable pin affects the write functionality of the port. An inactive Write Enable prevents any writing to memory cells. An active Write Enable causes the clock edge to write the data input signal to the memory location pointed to by the address inputs.

Address - A0, A1, A2, A3 (A4, A5, A6)

The address inputs select the memory cells for read or write. The width of the port determines the required address inputs. Note that the address inputs are not a bus in VHDL or Verilog instantiations.

Data In - D

The data input provides the new data value to be written into the RAM.

Data Out - O, SPO, and DPO

The data out O (Single-Port or SPO) and DPO (Dual-Port) reflects the contents of the memory cells referenced by the address inputs. Following an active write clock edge, the data out (O or SPO) reflects the newly written data.

Inverting Control Pins

The two control pins (WCLK and WE) each have an individual inversion option. Any control signal, including the clock, can be active at 0 (negative edge for the clock) or at 1 (positive edge for the clock) without requiring other logic resources.

GSR

The global set/reset (GSR) signal does not affect distributed SelectRAM modules.

Attributes

Content Initialization - INIT

With the INIT attributes, users can define the initial memory contents after configuration. By default distributed SelectRAM memory is initialized with all zeros during the device configuration sequence. The initialization attribute INIT represents the specified memory contents. Each INIT is a hex-encoded bit vector. Table 3-18 shows the length of the INIT attribute for each primitive.
Initialization in VHDL or Verilog Codes

Distributed SelectRAM memory structures can be initialized in VHDL or Verilog code for both synthesis and simulation. For synthesis, the attributes are attached to the distributed SelectRAM instantiation and are copied in the EDIF output file to be compiled by Xilinx Alliance Series™ tools. The VHDL code simulation uses a `generic` parameter to pass the attributes. The Verilog code simulation uses a `defparam` parameter to pass the attributes.

The distributed SelectRAM instantiation templates (in VHDL and Verilog) illustrate these techniques (see “VHDL and Verilog Templates,” page 148).

Location Constraints

The CLB has four slices S0, S1, S2, and S3. As an example, in the bottom left CLB, the slices have the coordinates shown below:

<table>
<thead>
<tr>
<th>Slice S3</th>
<th>Slice S2</th>
<th>Slice S1</th>
<th>Slice S0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1Y1</td>
<td>X1Y0</td>
<td>X0Y1</td>
<td>X0Y0</td>
</tr>
</tbody>
</table>

Distributed SelectRAM instances can have LOC properties attached to them to constrain placement. The RAM16X1S primitive fits in any LUT of slices S0 or S1.

For example, the instance `U_RAM16` is placed in slice X0Y0 with the following LOC properties:

```
INST "U_RAM16" LOC = "SLICE_X0Y0";
```

The RAM16X1D primitive occupies half of two slices, as shown in Figure 3-52. The first slice (output SPO) implements the read/write port with the same address A[3:0] for read and write. The second slice implements the second read port with the address DPRA[3:0] and is written simultaneously with the first slice to the address A[3:0].

---

### Table 3-18: INIT Attributes Length

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Template</th>
<th>INIT Attribute Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAM16X1S</td>
<td>SelectRAM_16S</td>
<td>4 digits</td>
</tr>
<tr>
<td>RAM32X1S</td>
<td>SelectRAM_32S</td>
<td>8 digits</td>
</tr>
<tr>
<td>RAM64X1S</td>
<td>SelectRAM_64S</td>
<td>16 digits</td>
</tr>
<tr>
<td>RAM128X1S</td>
<td>SelectRAM_128S</td>
<td>32 digits</td>
</tr>
<tr>
<td>RAM16X1D</td>
<td>SelectRAM_16S</td>
<td>4 digits</td>
</tr>
<tr>
<td>RAM32X1D</td>
<td>SelectRAM_32S</td>
<td>8 digits</td>
</tr>
<tr>
<td>RAM64X1D</td>
<td>SelectRAM_64S</td>
<td>16 digits</td>
</tr>
</tbody>
</table>

---
In the same CLB module, the dual-port RAM16X1D either occupies half of slices S0 (X0Y0) and S2 (X1Y0), or half of slices S1 (X0Y1) and S3 (X1Y1).

If a dual-port 16 x 2-bit module is built, the two RAM16X1D primitives occupy two slices, as long as they share the same clock and write enable, as illustrated in Figure 3-53.
A RAM32X1S primitive fits in one slice, as shown in Figure 3-54.

Following the same rules, a RAM32X1D primitive fits in two slices, with one slice implementing the read/write port and the second slice implementing the second read port.

The RAM64X1S primitive occupies two slices and the RAM64X1D primitive occupies four slices (one CLB element), with two slices implementing the read/write port and two other slices implementing the second read port. The RAM64X1S read path is built on the MUXF5 and MUXF6 multiplexers.

The RAM128X1S primitive occupies four slices, equivalent to one CLB element.

Distributed SelectRAM placement locations use the slice location naming convention, allowing LOC properties to transfer easily from array to array.

Applications

Creating Larger RAM Structures

The memory compiler program generates wider and/or deeper memory structures using distributed SelectRAM instances. Along with an EDIF file for inclusion in a design, this program produces VHDL and Verilog instantiation templates and simulation models.

Table 3-19 shows the generic VHDL and Verilog distributed SelectRAM examples provided to implement \( n \)-bit-wide memories.

<table>
<thead>
<tr>
<th>Submodules</th>
<th>Primitive</th>
<th>Size</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC2V_RAM16XN_S_SUBM</td>
<td>RAM16X1S</td>
<td>16 words x ( n )-bit</td>
<td>single-port</td>
</tr>
<tr>
<td>XC2V_RAM32XN_S_SUBM</td>
<td>RAM32X1S</td>
<td>32 words x ( n )-bit</td>
<td>single-port</td>
</tr>
<tr>
<td>XC2V_RAM64XN_S_SUBM</td>
<td>RAM64X1S</td>
<td>64 words x ( n )-bit</td>
<td>single-port</td>
</tr>
<tr>
<td>XC2V_RAM128XN_S_SUBM</td>
<td>RAM128X1S</td>
<td>128 words x ( n )-bit</td>
<td>single-port</td>
</tr>
<tr>
<td>XC2V_RAM16XN_D_SUBM</td>
<td>RAM16X1D</td>
<td>16 words x ( n )-bit</td>
<td>dual-port</td>
</tr>
<tr>
<td>XC2V_RAM32XN_D_SUBM</td>
<td>RAM32X1D</td>
<td>32 words x ( n )-bit</td>
<td>dual-port</td>
</tr>
<tr>
<td>XC2V_RAM64XN_D_SUBM</td>
<td>RAM64X1D</td>
<td>64 words x ( n )-bit</td>
<td>dual-port</td>
</tr>
</tbody>
</table>
By using the read/write port for the write address and the second read port for the read address, a FIFO that can read and write simultaneously is easily generated. Simultaneous access doubles the effective throughput of the memory.

**VHDL and Verilog Templates**

VHDL and Verilog templates are available in the Libraries Guide for all single-port and dual-port primitives. The number in each template indicates the number of bits (for example, SelectRAM_16S is the template for the 16 x 1-bit RAM); S indicates single-port, and D indicates dual-port.

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

The following are single-port templates:
- SelectRAM_16S
- SelectRAM_32S
- SelectRAM_64S
- SelectRAM_128S

The following are dual-port templates:
- SelectRAM_16D
- SelectRAM_32D
- SelectRAM_64D

Templates for the SelectRAM_16S module are provided in VHDL and Verilog code as examples.

**VHDL Template**

```vhdl
-- Module: SelectRAM_16S
--
-- Description: VHDL instantiation template
--    Distributed SelectRAM
--    Single Port 16 x 1
--    can be used also for RAM16X1S_1
--
-- Device: Virtex-II Pro Family
--
-- Components Declarations:
--
-- component RAM16X1S
--    pragma translate_off
generic (  -- RAM initialization ("0" by default) for functional simulation:
        INIT : bit_vector := X"0000"
    );
--    pragma translate_on
port (    D : in std_logic;
    WE : in std_logic;
    WCLK : in std_logic;
    A0 : in std_logic;
    A1 : in std_logic;
```
A2 : in std_logic;
A3 : in std_logic;
O : out std_logic
);
end component;
--
---------------------------------------------------------------------
-- Architecture section:
--
-- Attributes for RAM initialization ("0" by default):
attribute INIT: string;
--
attribute INIT of U_RAM16X1S: label is "0000";
--
-- Distributed SelectRAM Instantiation
U_RAM16X1S: RAM16X1S
port map (
D => , -- insert input signal
WE => , -- insert Write Enable signal
WCLK => , -- insert Write Clock signal
A0 => , -- insert Address 0 signal
A1 => , -- insert Address 1 signal
A2 => , -- insert Address 2 signal
A3 => , -- insert Address 3 signal
O =>   -- insert output signal
);
--
---------------------------------------------------------------------
Verilog Template

//
// Module: SelectRAM_16S
//
// Description: Verilog instantiation template
// Distributed SelectRAM
// Single Port 16 x 1
// can be used also for RAM16X1S_1
//
// Device: Virtex-II Pro Family
//
//---------------------------------------------------------------------
//
// Syntax for Synopsys FPGA Express
// synopsys translate_off

defparam

   //RAM initialization ("0" by default) for functional simulation:
U_RAM16X1S.INIT = 16'h0000;
// synopsys translate_on

   //Distributed SelectRAM Instantiation
RAM16X1S U_RAM16X1S ( .D(), // insert input signal
 .WE(), // insert Write Enable signal
Look-Up Tables as Shift Registers (SRLs)

Introduction

Virtex-II Pro / Virtex-II Pro X can configure any look-up table (LUT) as a 16-bit shift register without using the flip-flops available in each slice. Shift-in operations are synchronous with the clock, and output length is dynamically selectable. A separate dedicated output allows the cascading of any number of 16-bit shift registers to create whatever size shift register is needed. Each CLB resource can be configured using the 8 LUTs as a 128-bit shift register.

This section provides generic VHDL and Verilog submodules and reference code examples for implementing from 16-bit up to 128-bit shift registers. These submodules are built from 16-bit shift-register primitives and from dedicated MUXF5, MUXF6, MUXF7, and MUXF8 multiplexers.

These shift registers enable the development of efficient designs for applications that require delay or latency compensation. Shift registers are also useful in synchronous FIFO and content-addressable memory (CAM) designs. To quickly generate a Virtex-II Pro / Virtex-II Pro X shift register without using flip-flops (i.e., using the SRL16 element(s)), use the CORE Generator RAM-based Shift Register module.

Shift Register Operations

Data Flow

Each shift register (SRL16 primitive) supports:

- Synchronous shift-in
- Asynchronous 1-bit output when the address is changed dynamically
- Synchronous shift-out when the address is fixed

In addition, cascadable shift registers (SRLC16) support synchronous shift-out output of the last (16th) bit. This output has a dedicated connection to the input of the next SRLC16 inside the CLB resource. Two primitives are illustrated in Figure 3-55.
Shift Operation

The shift operation is a single clock-edge operation, with an active High clock enable feature. When enable is High, the input (D) is loaded into the first bit of the shift register, and each bit is shifted to the next highest bit position. In a cascadable shift register configuration (such as SRLC16), the last bit is shifted out on the Q15 output.

The bit selected by the 4-bit address appears on the Q output.

Dynamic Read Operation

The Q output is determined by the 4-bit address. Each time a new address is applied to the 4-input address pins, the new bit position value is available on the Q output after the time delay to access the LUT. This operation is asynchronous and independent of the clock and clock enable signals.

Figure 3-56 illustrates the shift and dynamic read operations.
Static Read Operation

If the 4-bit address is fixed, the Q output always uses the same bit position. This mode implements any shift register length up to 16 bits in one LUT. Shift register length is (N+1) where N is the input address.

The Q output changes synchronously with each shift operation. The previous bit is shifted to the next position and appears on the Q output.

Characteristics

- A shift operation requires one clock edge.
- Dynamic-length read operations are asynchronous (Q output).
- Static-length read operations are synchronous (Q output).
- The data input has a setup-to-clock timing specification.
- In a cascadable configuration, the Q15 output always contains the last bit value.
- The Q15 output changes synchronously after each shift operation.

Library Primitives and Submodules

Eight library primitives are available that offer optional clock enable (CE), inverted clock (CLK) and cascadable output (Q15) combinations.
Look-Up Tables as Shift Registers (SRLs)

Table 3-20 lists all of the available primitives for synthesis and simulation.

**Table 3-20:** Shift Register Primitives

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Length</th>
<th>Control</th>
<th>Address Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRL16</td>
<td>16 bits</td>
<td>CLK</td>
<td>A3,A2,A1,A0</td>
<td>Q</td>
</tr>
<tr>
<td>SRL16E</td>
<td>16 bits</td>
<td>CLK, CE</td>
<td>A3,A2,A1,A0</td>
<td>Q</td>
</tr>
<tr>
<td>SRL16_1</td>
<td>16 bits</td>
<td>CLK</td>
<td>A3,A2,A1,A0</td>
<td>Q</td>
</tr>
<tr>
<td>SRL16E_1</td>
<td>16 bits</td>
<td>CLK</td>
<td>A3,A2,A1,A0</td>
<td>Q</td>
</tr>
<tr>
<td>SRLC16</td>
<td>16 bits</td>
<td>CLK</td>
<td>A3,A2,A1,A0</td>
<td>Q, Q15</td>
</tr>
<tr>
<td>SRLC16E</td>
<td>16 bits</td>
<td>CLK, CE</td>
<td>A3,A2,A1,A0</td>
<td>Q, Q15</td>
</tr>
<tr>
<td>SRLC16_1</td>
<td>16 bits</td>
<td>CLK</td>
<td>A3,A2,A1,A0</td>
<td>Q, Q15</td>
</tr>
<tr>
<td>SRLC16E_1</td>
<td>16 bits</td>
<td>CLK</td>
<td>A3,A2,A1,A0</td>
<td>Q, Q15</td>
</tr>
</tbody>
</table>

In addition to the 16-bit primitives, three submodules that implement 32-bit, 64-bit, and 128-bit cascadable shift registers are provided in VHDL and Verilog code. Table 3-21 lists available submodules.

**Table 3-21:** Shift Register Submodules

<table>
<thead>
<tr>
<th>Submodule</th>
<th>Length</th>
<th>Control</th>
<th>Address Inputs</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>SRLC32E_SUBM</td>
<td>32 bits</td>
<td>CLK, CE</td>
<td>A4,A3,A2,A1,A0</td>
<td>Q, Q31</td>
</tr>
<tr>
<td>SRLC64E_SUBM</td>
<td>64 bits</td>
<td>CLK, CE</td>
<td>A5, A4, A3,A2,A1,A0</td>
<td>Q, Q63</td>
</tr>
<tr>
<td>SRLC128E_SUBM</td>
<td>128 bits</td>
<td>CLK, CE</td>
<td>A6, A5, A4, A3,A2,A1,A0</td>
<td>Q, Q127</td>
</tr>
</tbody>
</table>

The submodules are based on SRLC16E primitives, which are associated with dedicated multiplexers (MUXF5, MUXF6, and so forth). This implementation allows a fast static- and dynamic-length mode, even for very large shift registers.

Figure 3-57 represents the cascadable shift registers (32-bit and 64-bit) implemented by the submodules in Table 3-21.
Figure 3-57: Shift-Register Submodules (32-bit, 64-bit)
A 128-bit shift register is built on the same scheme and uses MUXF7 (address input A6). All clock enable (CE) and clock (CLK) inputs are connected to one global clock enable and one clock signal per submodule. If a global static- or dynamic-length mode is not required, the SRLC16E primitive can be cascaded without multiplexers.

Initialization in VHDL and Verilog Code

A shift register can be initialized in VHDL or Verilog code for both synthesis and simulation. For synthesis, the attribute is attached to the 16-bit shift register instantiation and is copied in the EDIF output file to be compiled by Xilinx Alliance Series tools. The VHDL code simulation uses a `generic` parameter to pass the attributes. The Verilog code simulation uses a `defparam` parameter to pass the attributes.

The V2_SRL16E shift register instantiation code examples (in VHDL and Verilog) illustrate these techniques (“VHDL and Verilog Templates,” page 159). V2_SRL16E.vhd and .v files are not a part of the documentation.

Port Signals

Clock - CLK

Either the rising edge or the falling edge of the clock is used for the synchronous shift-in. The data and clock enable input pins have set-up times referenced to the chosen edge of CLK.

Data In - D

The data input provides new data (one bit) to be shifted into the shift register.

Clock Enable - CE (optional)

The clock enable pin affects shift functionality. An inactive clock enable pin does not shift data into the shift register and does not write new data. Activating the clock enable allows the data in (D) to be written to the first location and all data to be shifted by one location. When available, new data appears on output pins (Q) and the cascadable output pin (Q15).

Address - A0, A1, A2, A3

Address inputs select the bit (range 0 to 15) to be read. The n\textsuperscript{th} bit is available on the output pin (Q). Address inputs have no effect on the cascadable output pin (Q15), which is always the last bit of the shift register (bit 15).

Data Out - Q

The data output Q provides the data value (1 bit) selected by the address inputs.

Data Out - Q15 (optional)

The data output Q15 provides the last bit value of the 16-bit shift register. New data becomes available after each shift-in operation.
Inverting Control Pins

The two control pins (CLK, CE) have an individual inversion option. The default is the rising clock edge and active High clock enable.

GSR

The global set/reset (GSR) signal has no impact on shift registers.

Attributes

Content Initialization - INIT

The INIT attribute defines the initial shift register contents. The INIT attribute is a hex-encoded bit vector with four digits (0000). The left-most hexadecimal digit is the most significant bit. By default the shift register is initialized with all zeros during the device configuration sequence, but any other configuration value can be specified.

Location Constraints

Each CLB resource has four slices: S0, S1, S2, and S3. As an example, in the bottom left CLB resource, each slice has the coordinates shown in Table 3-22.

Table 3-22: Slice Coordinates in the Bottom-Left CLB Resource

<table>
<thead>
<tr>
<th>Slice S3</th>
<th>Slice S2</th>
<th>Slice S1</th>
<th>Slice S0</th>
</tr>
</thead>
<tbody>
<tr>
<td>X1Y1</td>
<td>X1Y0</td>
<td>X0Y1</td>
<td>X0Y0</td>
</tr>
</tbody>
</table>

To constrain placement, shift register instances can have LOC properties attached to them. Each 16-bit shift register fits in one LUT.

A 32-bit shift register in static or dynamic address mode fits in one slice (two LUTs and one MUXF5). This shift register can be placed in any slice.

A 64-bit shift register in static or dynamic address mode fits in two slices. These slices are either S0 and S1, or S2 and S3. Figure 3-58 illustrates the position of the four slices in a CLB resource.

The dedicated CLB shift chain runs from the top slice to the bottom slice. The data input pin must either be in slice S1 or in S3. The address selected as the output pin (Q) is the MUXF6 output.

A 128-bit shift register in static or dynamic address mode fits in a four-slice CLB resource. The data input pin has to be in slice S3. The address selected as the output pin (Q) is the MUXF7 output.
Look-Up Tables as Shift Registers (SRLs)

Fully Synchronous Shift Registers

All shift-register primitives and submodules do not use the register(s) available in the same slice(s). To implement a fully synchronous read and write shift register, output pin Q must be connected to a flip-flop. Both the shift register and the flip-flop share the same clock, as shown in Figure 3-59.

Figure 3-58: Shift Register Placement
This configuration provides a better timing solution and simplifies the design. Because the flip-flop must be considered to be the last register in the shift-register chain, the static or dynamic address should point to the desired length minus one. If needed, the cascadable output can also be registered in a flip-flop.

**Static-Length Shift Registers**

The cascadable 16-bit shift register implements any static length mode shift register without the dedicated multiplexers (MUXF5, MUXF6,…). Figure 3-60 illustrates a 40-bit shift register. Only the last SRLC16E primitive needs to have its address inputs tied to “0111”. Alternatively, shift register length can be limited to 39 bits (address tied to “0110”) and a flip-flop can be used as the last register. (In an SRLC16E primitive, the shift register length is the address input + 1.)
VHDL and Verilog Instantiation

VHDL and Verilog instantiation templates are available in the Libraries Guide for all primitives and submodules.

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

The ShiftRegister_C_x (with x = 16, 32, 64, 128, or 256) templates are cascadable modules and instantiate the corresponding SRLCxE primitive (16) or submodule (32, 64, 128, or 256).

The ShiftRegister_16 template can be used to instantiate an SRL16 primitive.

VHDL and Verilog Templates

In template names, the number indicates the number of bits (for example, SHIFT_REGISTER_16 is the template for the 16-bit shift register) and the “C” extension means the template is cascadable.

The following are templates for primitives:

- SHIFT_REGISTER_16
- SHIFT_REGISTER_16_C

The following are templates for submodules:

- SHIFT_REGISTER_32_C (submodule: SRLC32E_SUBM)
- SHIFT_REGISTER_64_C (submodule: SRLC64E_SUBM)
- SHIFT_REGISTER_128_C (submodule: SRLC128E_SUBM)

The corresponding submodules have to be synthesized with the design.

Templates for the SHIFT_REGISTER_16_C module are provided in VHDL and Verilog code as an example.

VHDL Template:

```vhdl
-- Module: SHIFT_REGISTER_C_16
-- Description: VHDL instantiation template
-- CASCADABLE 16-bit shift register with enable (SRLC16E)
-- Device: Virtex-II Pro Family
---------------------------------------------------------------------
-- Components Declarations:
--
component SRLC16E
-- pragma translate_off
generic (
-- Shift Register initialization ("0" by default) for functional
simulation:
    INIT : bit_vector := X"0000"
);
-- pragma translate_on
port (   
    D : in std_logic;
    CE : in std_logic;
    CLK : in std_logic;
    A0 : in std_logic;
    A1 : in std_logic;
    A2 : in std_logic;
    A3 : in std_logic;
```
Q : out std_logic;
Q15 : out std_logic
);
end component;
-- Architecture Section:
--
-- Attributes for Shift Register initialization (*0* by default):
attribute INIT: string;
--
attribute INIT of U_SRLC16E: label is "0000";
--
-- ShiftRegister Instantiation
U_SRLC16E: SRLC16E
port map (
D => , -- insert input signal
CE => , -- insert Clock Enable signal (optional)
CLK => , -- insert Clock signal
A0 => , -- insert Address 0 signal
A1 => , -- insert Address 1 signal
A2 => , -- insert Address 2 signal
A3 => , -- insert Address 3 signal
Q => , -- insert output signal
Q15 =>   -- insert cascadable output signal
);

Verilog Template:

// Module: SHIFT_REGISTER_16
// Description: Verilog instantiation template
// Cascadable 16-bit Shift Register with Clock Enable (SRLC16E)
// Device: Virtex-II Pro Family
//-------------------------------------------------------------------
// Syntax for Synopsys FPGA Express
// synopsys translate_off
//
defparam

//Shift Register initialization (*0* by default) for functional simulation:
U_SRLC16E.INIT = 16'h0000;
// synopsys translate_on

//SelectShiftRegister-II Instantiation
SRLC16E U_SRLC16E   ( .D(),
       .A0(),
       .A1(),
       .A2(),
       .A3(),
       .CLK(),
       .CE(),
       .Q(),
       .Q15()
);

// synthesis attribute declarations
/* synopsys attribute
INIT "0000"
*/
Large Multiplexers

Introduction

Virtex-II Pro / Virtex-II Pro X slices contain dedicated two-input multiplexers (one MUXF5 and one MUXFX per slice). These multiplexers combine the 4-input LUT outputs or the outputs of other multiplexers. Using the multiplexers MUXF5, MUXF6, MUXF7 and MUXF8 allows to combine 2, 4, 8 and 16 LUTs. Specific routing resources are associated with these 2-input multiplexers to guarantee a fast implementation of any combinatorial function built upon LUTs and MUXFX.

The combination of the LUTs and the MUXFX offers a unique solution to the design of wide-input functions. This section illustrates the implementation of large multiplexers up to 32:1. Any slice can implement a 4:1 multiplexer, any CLB can implement a 16:1 multiplexer, and 2 CLBs can implement a 32:1 multiplexer. Such multiplexers are just one example of wide-input combinatorial function taking advantage of the MUXFX feature. Many other logic functions can be mapped in the LUT and MUXFX features.

This section provides generic VHDL and Verilog reference code implementing multiplexers. These submodules are built from LUTs and the dedicated MUXF5, MUXF6, MUXF7, and MUXF8 multiplexers. To automatically generate large multiplexers using these dedicated elements, use the CORE Generator Bit Multiplexer and Bus Multiplexer modules.

For applications like comparators, encoder-decoders or “case” statement in VHDL or Verilog, these resources offer an optimal solution.

Virtex-II Pro/Virtex-II Pro X CLB Resources

Slice Multiplexers

Each Virtex-II Pro / Virtex-II Pro X slice has a MUXF5 to combine the outputs of the 2 LUTs and an extra MUXFX. Figure 3-61 illustrates any single combinatorial function with up to 5 inputs, or in some functions up to 9 inputs, within one slice.

![Figure 3-61: LUTs and MUXF5 in a Slice](UG002_C2_016_081500)

Each Virtex-II Pro / Virtex-II Pro X CLB contains 4 slices. The second MUXFX implements a MUXF6, MUXF7 or MUXF8 according to the position of the slice in the CLB. These MUXFX are designed to allow LUTs combination up to 16 LUTs in two adjacent CLBs.
Figure 3-62 shows the relative position of the slices in the CLB.

Slices S0 and S2 have a MUXF6, designed to combine the outputs of two MUXF5 resources. Figure 3-63 illustrates any single combinatorial function up to 6 inputs, or in some functions up to 18 inputs, in slices S0 and S1, or in slices S2 and S3.
The slice S1 has a MUXF7, designed to combine the outputs of two MUXF6. Figure 3-64 illustrates any single combinatorial function up to 7 inputs, or in some functions up to 35 inputs, in a Virtex-II CLB.

Figure 3-64: LUTs and (MUXF5, MUXF6, and MUXF7) in One CLB
The slice S3 of each CLB has a MUXF8. Any single combinatorial functions of up to 8 inputs, or some functions up to 68 inputs, fit in two CLBs as shown in Figure 3-65. The outputs of two MUXF7 are combined through dedicated routing resources between two adjacent CLBs in a column.

![Figure 3-65: MUXF8 Combining Two Adjacent CLBs](image-url)
Wide-Input Multiplexers

Each LUT can implement a 2:1 multiplexer. In each slice, the MUXF5 and two LUTs can implement a 4:1 multiplexer. As shown in Figure 3-66, the MUXF6 and two slices can implement a 8:1 multiplexer. The MUXF7 and the four slices of any CLB can implement a 16:1 and the MUXF8 and two CLBs can implement a 32:1 multiplexer.

Characteristics

- Implementation in one level of logic (LUT) and dedicated MUXFX
- Full combinatorial path
Library Primitives and Submodules

Four library primitives are available that offer access to the dedicated MUXFX in each slice. In the example shown in Table 3-23, MUXF7 is available only in slice S1.

Table 3-23: MUXFX Resources

<table>
<thead>
<tr>
<th>Primitive</th>
<th>Slice</th>
<th>Control</th>
<th>Input (I0, I1)</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUXF5</td>
<td>S0, S1, S2, S3</td>
<td>S</td>
<td>I0, I1</td>
<td>O</td>
</tr>
<tr>
<td>MUXF6</td>
<td>S0, S2</td>
<td>S</td>
<td>I0, I1</td>
<td>O</td>
</tr>
<tr>
<td>MUXF7</td>
<td>S1</td>
<td>S</td>
<td>I0, I1</td>
<td>O</td>
</tr>
<tr>
<td>MUXF8</td>
<td>S3</td>
<td>S</td>
<td>I0, I1</td>
<td>O</td>
</tr>
</tbody>
</table>

In addition to the primitives, five submodules that implement multiplexers from 2:1 to 32:1 are provided in VHDL and Verilog code. Synthesis tools can automatically infer the above primitives (MUXF5, MUXF6, MUXF7, and MUXF8); however, the submodules described in this section used instantiation of the new MUXFX to guarantee an optimized result. Table 3-24 lists available submodules:

Table 3-24: Available Submodules

<table>
<thead>
<tr>
<th>Submodule</th>
<th>Multiplexer</th>
<th>Control</th>
<th>Input (DATA_I[1:0], DATA_I[3:0], DATA_I[8:0], DATA_I[15:0], DATA_I[31:0])</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>MUX_2_1_SUBM</td>
<td>2:1</td>
<td>SELECT_I</td>
<td>DATA_I[1:0]</td>
<td>DATA_O</td>
</tr>
<tr>
<td>MUX_4_1_SUBM</td>
<td>4:1</td>
<td>SELECT_I</td>
<td>DATA_I[3:0]</td>
<td>DATA_O</td>
</tr>
<tr>
<td>MUX_8_1_SUBM</td>
<td>8:1</td>
<td>SELECT_I</td>
<td>DATA_I[8:0]</td>
<td>DATA_O</td>
</tr>
<tr>
<td>MUX_16_1_SUBM</td>
<td>16:1</td>
<td>SELECT_I</td>
<td>DATA_I[15:0]</td>
<td>DATA_O</td>
</tr>
<tr>
<td>MUX_32_1_SUBM</td>
<td>32:1</td>
<td>SELECT_I</td>
<td>DATA_I[31:0]</td>
<td>DATA_O</td>
</tr>
</tbody>
</table>

Port Signals

Data In - DATA_I

The data input provides the data to be selected by the SELECT_I signal(s).

Control In - SELECT_I

The select input signal or bus determines the DATA_I signal to be connected to the output DATA_O. For example, the MUX_4_1_SUBM multiplexer has a 2-bit SELECT_I bus and a 4-bit DATA_I bus. Table 3-25 shows the DATA_I selected for each SELECT_I value.

Table 3-25: Selected Inputs

<table>
<thead>
<tr>
<th>SELECT_I[1:0]</th>
<th>DATA_O</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0</td>
<td>DATA_I[0]</td>
</tr>
<tr>
<td>0 1</td>
<td>DATA_I[1]</td>
</tr>
<tr>
<td>1 0</td>
<td>DATA_I[2]</td>
</tr>
<tr>
<td>1 1</td>
<td>DATA_I[3]</td>
</tr>
</tbody>
</table>

www.xilinx.com  Virtex-II Pro and Virtex-II Pro X FPGA User Guide  UG012 (v4.0) 23 March 2005
Data Out - DATA_O

The data output O provides the data value (1 bit) selected by the control inputs.

Applications

Multiplexers are used in various applications. These are often inferred by synthesis tools when a “case” statement is used (see the example below). Comparators, encoder-decoders and wide-input combinatorial functions are optimized when they are based on one level of LUTs and dedicated MUXFX resources of the Virtex-II Pro / Virtex-II Pro X CLBs.

VHDL and Verilog Instantiation

The primitives (MUXF5, MUXF6, and so forth) can be instantiated in VHDL or Verilog code, to design wide-input functions.

The submodules (MUX_2_1_SUBM, MUX_4_1_SUBM, and so forth) can be instantiated in VHDL or Verilog code to implement multiplexers. However the corresponding submodule must be added to the design directory as hierarchical submodule. For example, if a module is using the MUX_16_1_SUBM, the MUX_16_1_SUBM.vhd file (VHDL code) or MUX_16_1_SUBM.v file (Verilog code) must be compiled with the design source code. The submodule code can also be “cut and pasted” into the designer source code.

VHDL and Verilog Submodules

VHDL and Verilog submodules are available to implement multiplexers up to 32:1. They illustrate how to design with the MUXFX resources. When synthesis infers the corresponding MUXFX resource(s), the VHDL or Verilog code is behavioral code (“case” statement). Otherwise, the equivalent “case” statement is provided in comments and the correct MUXFX are instantiated. However, most synthesis tools support the inference of all of the MUXFX. The following examples can be used as guidelines for designing other wide-input functions.

The following submodules are available:
- MUX_2_1_SUBM (behavioral code)
- MUX_4_1_SUBM
- MUX_8_1_SUBM
- MUX_16_1_SUBM
- MUX_32_1_SUBM

The corresponding submodules have to be synthesized with the design

The submodule MUX_16_1_SUBM in VHDL and Verilog are provided as example.

VHDL Template

```vhdl
-- Module: MUX_16_1_SUBM
-- Description: Multiplexer 16:1
--
-- Device: Virtex-II Pro Family
---------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;

-- Syntax for Synopsys FPGA Express
```
-- pragma translate_off
library UNISIM;
use UNISIM.VCOMPONENTS.ALL;
-- pragma translate_on

entity MUX_16_1_SUBM is
  port (
    DATA_I: in std_logic_vector (15 downto 0);
    SELECT_I: in std_logic_vector (3 downto 0);
    DATA_O: out std_logic
  );
end MUX_16_1_SUBM;

architecture MUX_16_1_SUBM_arch of MUX_16_1_SUBM is
  -- Component Declarations:
  component MUXF7
    port (
      I0: in std_logic;
      I1: in std_logic;
      S: in std_logic;
      O: out std_logic
    );
  end component;
  --
  -- Signal Declarations:
  signal DATA_MSB : std_logic;
  signal DATA_LSB : std_logic;
  --
  begin
    -- If synthesis tools support MUXF7:
    SELECT_PROCESS: process (SELECT_I, DATA_I)
    begin
      case SELECT_I is
        when "0000" => DATA_O <= DATA_I (0);
        when "0001" => DATA_O <= DATA_I (1);
        when "0010" => DATA_O <= DATA_I (2);
        when "0011" => DATA_O <= DATA_I (3);
        when "0100" => DATA_O <= DATA_I (4);
        when "0101" => DATA_O <= DATA_I (5);
        when "0110" => DATA_O <= DATA_I (6);
        when "0111" => DATA_O <= DATA_I (7);
        when "1000" => DATA_O <= DATA_I (8);
        when "1001" => DATA_O <= DATA_I (9);
        when "1010" => DATA_O <= DATA_I (10);
        when "1011" => DATA_O <= DATA_I (11);
        when "1100" => DATA_O <= DATA_I (12);
        when "1101" => DATA_O <= DATA_I (13);
        when "1110" => DATA_O <= DATA_I (14);
        when "1111" => DATA_O <= DATA_I (15);
        when others => DATA_O <= 'X';
      end case;
    end process SELECT_PROCESS;
    --
    -- If synthesis tools DO NOT support MUXF7:
    SELECT_PROCESS_LSB: process (SELECT_I, DATA_I)
    begin
      case SELECT_I (2 downto 0) is
        when "000" => DATA_LSB <= DATA_I (0);
        when others => DATA_LSB <= 'X';
      end case;
    end process SELECT_PROCESS_LSB;
    --
end MUX_16_1_SUBM;
Large Multiplexers

when "001" => DATA_LSB <= DATA_I (1);
when "010" => DATA_LSB <= DATA_I (2);
when "011" => DATA_LSB <= DATA_I (3);
when "100" => DATA_LSB <= DATA_I (4);
when "101" => DATA_LSB <= DATA_I (5);
when "110" => DATA_LSB <= DATA_I (6);
when "111" => DATA_LSB <= DATA_I (7);
when others => DATA_LSB <= 'X';
end case;
end process SELECT_PROCESS_LSB;

--

SELECT_PROCESS_MSB: process (SELECT_I, DATA_I)
begin
  case SELECT_I (2 downto 0) is
    when "000" => DATA_MSB <= DATA_I (8);
    when "001" => DATA_MSB <= DATA_I (9);
    when "010" => DATA_MSB <= DATA_I (10);
    when "011" => DATA_MSB <= DATA_I (11);
    when "100" => DATA_MSB <= DATA_I (12);
    when "101" => DATA_MSB <= DATA_I (13);
    when "110" => DATA_MSB <= DATA_I (14);
    when "111" => DATA_MSB <= DATA_I (15);
    when others => DATA_MSB <= 'X';
  end case;
end process SELECT_PROCESS_MSB;

-- MUXF7 instantiation
U_MUXF7: MUXF7
  port map (
    I0 => DATA_LSB,
    I1 => DATA_MSB,
    S => SELECT_I (3),
    O => DATA_O
  );

--
end MUX_16_1_SUBM_arch;

--

Verilog Template

// Module: MUX_16_1_SUBM
//
// Description: Multiplexer 16:1
// Device: Virtex-II Pro Family
//-------------------------------------------------------------
//
module MUX_16_1_SUBM (DATA_I, SELECT_I, DATA_O);

input [15:0]DATA_I;
input [3:0]SELECT_I;

output DATA_O;
wire [2:0]SELECT;
reg DATA_LSB;
reg DATA_MSB;
assign SELECT[2:0] = SELECT_I[2:0];
```vhdl
/*
// If synthesis tools supports MUXF7 :
always @(DATA_I or SELECT_I)

    case (SELECT_I)
        4'b0000 : DATA_O <= DATA_I[0];
        4'b0001 : DATA_O <= DATA_I[1];
        4'b0010 : DATA_O <= DATA_I[2];
        4'b0011 : DATA_O <= DATA_I[3];
        4'b0100 : DATA_O <= DATA_I[4];
        4'b0101 : DATA_O <= DATA_I[5];
        4'b0110 : DATA_O <= DATA_I[6];
        4'b0111 : DATA_O <= DATA_I[7];
        4'b1000 : DATA_O <= DATA_I[8];
        4'b1001 : DATA_O <= DATA_I[9];
        4'b1010 : DATA_O <= DATA_I[10];
        4'b1011 : DATA_O <= DATA_I[11];
        4'b1100 : DATA_O <= DATA_I[12];
        4'b1101 : DATA_O <= DATA_I[13];
        4'b1110 : DATA_O <= DATA_I[14];
        4'b1111 : DATA_O <= DATA_I[15];
        default : DATA_O <= 1'bx;
    endcase
 */

always @(SELECT or DATA_I)

    case (SELECT)
        3'b000 : DATA_LSB <= DATA_I[0];
        3'b001 : DATA_LSB <= DATA_I[1];
        3'b010 : DATA_LSB <= DATA_I[2];
        3'b011 : DATA_LSB <= DATA_I[3];
        3'b100 : DATA_LSB <= DATA_I[4];
        3'b101 : DATA_LSB <= DATA_I[5];
        3'b110 : DATA_LSB <= DATA_I[6];
        3'b111 : DATA_LSB <= DATA_I[7];
        default : DATA_LSB <= 1'bx;
    endcase

always @(SELECT or DATA_I)

    case (SELECT)
        3'b000 : DATA_MSB <= DATA_I[8];
        3'b001 : DATA_MSB <= DATA_I[9];
        3'b010 : DATA_MSB <= DATA_I[10];
        3'b011 : DATA_MSB <= DATA_I[11];
        3'b100 : DATA_MSB <= DATA_I[12];
        3'b101 : DATA_MSB <= DATA_I[13];
        3'b110 : DATA_MSB <= DATA_I[14];
        3'b111 : DATA_MSB <= DATA_I[15];
        default : DATA_MSB <= 1'bx;
    endcase

// MUXF7 instantiation
MUXF7 U_MUXF7
    (.I0(DATA_LSB),
     .I1(DATA_MSB),
     .S(SELECT_I[3]));
```
Sum of Products (SOP) Logic

Introduction

Virtex-II Pro / Virtex-II Pro X slices contain a dedicated two-input multiplexer (MUXCY) and a two-input OR gate (ORCY) to perform operations involving wide AND and OR gates. These combine the four-input LUT outputs. These gates can be cascaded in a chain to provide the wide AND functionality across slices. The output from the cascaded AND gates can then be combined with the dedicated ORCY to produce the Sum of Products (SOP).

Virtex-II Pro/Virtex-II Pro X CLB Resources

Each Virtex-II Pro / Virtex-II Pro X slice has a MUXCY, which uses the output from the LUTs as a SELECT signal. Depending on the width of data desired, several slices can be used to provide the SOP output. Figure 3-67 illustrates the logic involved in designing a 16-input AND gate. It utilizes the 4-input LUT to provide the necessary SELECT signal for the MUXCY. Only when all of the input signals are High, can the VCC at the bottom reach the output. This use of carry logic helps to perform AND functions at high speed and saves logic resources.

Figure 3-67: Implementing a 16-bit Wide AND Gate Using MUXCY & ORCY
The output from the chain of AND gates is passed as one of the inputs of the dedicated OR gate, ORCY. To calculate the SOP, these AND chains can be cascaded vertically across several CLBs, depending on the width of the input data. Figure 3-68 illustrates how the AND outputs are then passed in through the ORCY gates in a horizontal cascade, the sum of which is the Sum of Products.

Figure 3-68: 64-bit Input SOP Design

**VHDL Parameters**

**AND_WIDTH Parameter**

The width of each AND gate used in the cascade.

**PROD_TERM Parameter**

The number of AND gates used along each vertical cascade.

**AND_IN Parameter**

Data input to the AND gates. The total width of data is calculated from the product of AND_WIDTH and PROD_TERM.
SOP_OUT Parameter

The Sum of Products (SOP) output data from the cascade chain.

Applications

These logic gates can be used in various applications involving very wide AND gates and Sum of Products (SOP) functions.

VHDL and Verilog Instantiation

To implement wide-input AND functions, MUXCY and ORCY primitives can be instantiated in VHDL or Verilog code. The submodule code provided can be used to implement wide-input AND gates for any width of input data.

VHDL and Verilog Submodules

VHDL and Verilog submodules are available to implement the cascade chain of wide-input AND gates and OR gates to calculate the Sum of Products (SOP). The VHDL module provided uses a generic case, where the width of data and the product terms can be specified in the case. The Verilog module provides a 64-bit input example, using four wide AND chains, each of which handle 16 bits of data.

VHDL Templates

```vhdl
-- Module : AND_CHAIN
-- Description : 16 input AND gate
--
-- Device : Virtex-II Pro Family
---------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
--library UNISIM;
--use UNISIM.VCOMPONENTS.ALL;

entity AND_CHAIN is
    generic (
        input_width : integer); --must be a 4x value
    port ( 
        data_in : in std_logic_vector( input_width-1 downto 0);
        carry_in : in std_logic;
        out_andor_chain : out std_logic);
end AND_CHAIN;

architecture AND_CHAIN_arch of AND_CHAIN is

component ORCY
    port( i : std_logic;
         ci : in std_logic;
         o : out std_logic);
end component;

component AND_LOGIC
    port( sel_data : in std_logic_vector(3 downto 0);
         data_cin : in std_logic;
         data_out : out std_logic);
end component;
```
signal VCC, GND : std_logic;
signal cout : std_logic_vector(input_width/4 downto 0);
signal out_and_chain : std_logic;

begin

VCC <= '1';
GND <= '0';

--initialization of first input for MUXCY
cout(0) <= VCC;

and_chain_x : for i in (input_width/4) - 1 downto 0 generate
  AND_LOGIC_inst : AND_LOGIC
  port map ( 
    sel_data => data_in((4 * i + 3) downto (4 * i)),
    data_cin => cout(i),
    data_out => cout(i + 1));
end generate;

out_and_chain <= cout(input_width/4);

orcy_inst : ORCY
  port map( i => out_and_chain,
     ci => carry_in,
     o => out_andor_chain);

end AND_CHAIN_arch;

---------------------------------------------------------------------
-- Module AND_LOGIC
-- Description : 4-input AND gate
--
-- Device : Virtex-II Pro Family
---------------------------------------------------------------------
library IEEE;
use IEEE.std_logic_1164.all;
--library UNISIM;
--use UNISIM.VCOMPONENTS.ALL;

entity AND_LOGIC is
  port(
    sel_data : in std_logic_vector(3 downto 0); -- data for select
    data_in : in std_logic_vector(4 downto 0)); -- data for MUXCY from LUT
end AND_LOGIC;

architecture AND_LOGIC_arch of AND_LOGIC is

component MUXCY
  port( 
    DI : in std_logic;
    CI : in std_logic;
    s : in std_logic;
    o : out std_logic);
end component;
signal GND : std_logic;
signal sel:std_logic;

begin

GND <= '0';
sel <= sel_data(0) and sel_data(1) and sel_data(2) and sel_data(3);

--Wide AND gate using MUXCY
MUX : MUXCY
  port map (  
    DI => GND,
    CI => data_cin,
    s => sel,
    o => data_out);

end AND_LOGIC_arch;

---------------------------------------------------------------------
-- Module : SOP_SUBM
-- Description : Implementing SOP using MUXCY and ORCY
--
-- Device : Virtex-II Pro Family
---------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
--library UNISIM;
--use UNISIM.VCOMPONENTS.ALL;

entity SOP_SUBM is
  generic(
      and_width : integer :=16 ;
      prod_term : integer := 4 );
  port(
      and_in : in std_logic_vector(and_width * prod_term - 1 downto 0);
      sop_out : out std_logic);
end SOP_SUBM;

architecture SOP_SUBM_arch of SOP_SUBM is

component AND_CHAIN
  generic (  
    input_width : integer); --must be a 4x value
  port (  
    data_in : in std_logic_vector( input_width-1 downto 0);
    carry_in : in std_logic;
    out_andor_chain : out std_logic);
end component;

signal VCC, GND : std_logic;
signal carry : std_logic_vector(prod_term downto 0);

begin

VCC <= '1';
GND <= '0';

carry(0) <= GND;
andor_inst : for i in 0 to (prod_term - 1) generate

and_chainx : AND_CHAIN
    generic map(
        input_width => and_width)
    port map(
        data_in => and_in((and_width * i + (and_width -1)) downto
        (and_width * i)),
        carry_in => carry(i),
        out_andor_chain => carry(i + 1));
end generate;
sop_out <= carry(prod_term);
end SOP_SUBM_arch;

Verilog Templates

// Module : AND_CHAIN
// Description : 16 input AND gate
//
// Device : Virtex-II Family
//-------------------------------------------------------------------
module AND_CHAIN(data_in, carry_in, out_andor_chain);
    input [15:0] data_in;
    input carry_in;
    output out_andor_chain;
    wire VCC = 1'b1;
    wire out_and_chain;
    wire dat_out1, data_out2, data_out3;
AND_LOGIC_OR u4(.sel_data(data_in[15:12]), .data_cin(data_out3),
    .carry_in(carry_in), .data_out(out_andor_chain));
AND_LOGIC u3(.sel_data(data_in[11:8]), .data_cin(data_out2),
    .data_out(data_out3));
AND_LOGIC u2(.sel_data(data_in[7:4]), .data_cin(data_out1),
    .data_out(data_out2));
AND_LOGIC u1(.sel_data(data_in[3:0]), .data_cin(VCC),
    .data_out(data_out1));
endmodule

//-------------------------------------------------------------------
// Module AND_LOGIC
// Description : 4-input AND gate
//
// Device : Virtex-II Family
//-------------------------------------------------------------------
// Module : init_and
//
module AND_LOGIC(sel_data, data_cin, data_out);
    input[3:0] sel_data;
    input data_cin;
    output data_out;
    wire GND = 1'b0;
    wire VCC = 1'b1;
MUXCY muxcy_inst (.DI(GND), .CI(data_cin), .S(and_out), .O(data_out));
endmodule

// Module AND_LOGIC + ORCY
module AND_LOGIC_OR(sel_data, data_cin, carry_in, data_out);
```verilog
input[3:0] sel_data;
input data_cin;
input carry_in;
output data_out;
wire data_mux_out;
wire GND = 1'b0;
wire VCC = 1'b1;
wire and_out;
MUXCY muxcy_inst (.DI(GND), .CI(data_cin), .S(and_out),
.O(data_mux_out)) /* synthesis RLOC="x0y0" */;
ORCY u5(.I(carry_in), .CI(data_mux_out), .O(data_out)) /* synthesis
RLOC="x0y0" */;
endmodule
```

```verilog
// Module : SOP_SUBM
// Description : Implementing SOP using MUXCY and ORCY
//
// Device : Virtex-II Family
//---------------------------------------------------------------
module SOP_SUBM(and_in, sop_out);
input [63:0] and_in;
output sop_out;
wire out_andor_chain1, out_andor_chain2, out_andor_chain3;
wire GND = 1'b0;
AND_CHAIN u4(.data_in(and_in[63:48]), .carry_in(out_andor_chain3),
.out_andor_chain(sop_out));
AND_CHAIN u3(.data_in(and_in[47:32]), .carry_in(out_andor_chain2),
.out_andor_chain(out_andor_chain3));
AND_CHAIN u2(.data_in(and_in[31:16]), .carry_in(out_andor_chain1),
.out_andor_chain(out_andor_chain2));
AND_CHAIN u1(.data_in(and_in[15:0]), .carry_in(GND),
.out_andor_chain(out_andor_chain1));
endmodule
```

**Embedded Multipliers**

**Introduction**

Virtex-II Pro / Virtex-II Pro X devices feature a large number of embedded 18-bit X 18-bit two’s-complement embedded multipliers. The embedded multipliers offer fast, efficient means to create 18-bit signed by 18-bit signed multiplication products. The multiplier blocks share routing resources with the Block SelectRAM memory, allowing for increased efficiency for many applications. Cascading of multipliers can be implemented with additional logic resources in local Virtex-II Pro / Virtex-II Pro X slices.

Applications such as signed-signed, signed-unsigned, and unsigned-unsigned multiplication, logical, arithmetic, and barrel shifters, two’s-complement and magnitude return are easily implemented.

Using the CORE Generator, the designer can quickly generate multipliers that make use of the embedded 18-bit x 18-bit two’s-complement multipliers (V2.0 or later) of the Multiplier core for Virtex-II Pro / Virtex-II Pro X devices.
Two’s-Complement Signed Multiplier

Data Flow

Each embedded multiplier block (MULT18X18 primitive) supports two independent dynamic data input ports: 18-bit signed or 17-bit unsigned. The MULT18X18 primitive is illustrated in Figure 3-69.

In addition, efficient cascading of multipliers up to 35-bit X 35-bit signed can be accomplished by using 4 embedded multipliers, one 36-bit adder, and one 53-bit adder. See Figure 3-71.

Library Primitives and Submodules

One library primitive (MULT18X18) is available. Table 3-26 lists the attributes of this primitive.

Table 3-26: Multiplier Primitives

<table>
<thead>
<tr>
<th>Primitive</th>
<th>A width</th>
<th>B width</th>
<th>P width</th>
<th>Signed/Unsigned</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULT18X18</td>
<td>18</td>
<td>18</td>
<td>36</td>
<td>Signed (Two’s complement)</td>
<td>Combinatorial</td>
</tr>
<tr>
<td>MULT18X18S</td>
<td>18</td>
<td>18</td>
<td>36</td>
<td>Signed (Two’s complement)</td>
<td>Registered</td>
</tr>
</tbody>
</table>

The registered version of the multiplier adds a clock input C, an active-High clock enable CE, and a synchronous reset R (see Figure 3-70). The registers are implemented in the multiplier itself and do not require any other resources. The control inputs C, CE, and R all have built-in programmable polarity. The data inputs, clock enable, and reset all must meet a setup time before the clock edge, and the data on the P outputs changes after the clock-to-output delay.

The pin names used in the Xilinx implementation tools, such as the FPGA Editor, are identical to those used in the library primitives.
In addition to the primitive, 15 submodules that implement various widths of signed and unsigned multipliers and two’s-complement return functions are provided in VHDL and Verilog code. Multipliers using cascaded MULT18X18 primitives are included with registers between stages causing three cycles of latency. Multipliers that make use of the embedded Virtex-II Pro / Virtex-II Pro X 18-bit by 18-bit two’s complement multipliers can be easily generated using V2.0 of the CORE Generator Multiplier module. Table 3-27 lists cascaded multiplier submodules.

Table 3-27: Embedded Multiplier Submodules - Cascaded MULT18X18

<table>
<thead>
<tr>
<th>Submodule</th>
<th>A Width</th>
<th>B Width</th>
<th>P Width</th>
<th>Signed/Unsigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULT35X35_S</td>
<td>35</td>
<td>35</td>
<td>70</td>
<td>Signed</td>
</tr>
<tr>
<td>MULT34X34_U</td>
<td>34</td>
<td>34</td>
<td>68</td>
<td>Unsigned</td>
</tr>
</tbody>
</table>

Figure 3-71 represents the cascaded scheme used to implement a 35-bit by 35-bit signed multiplier utilizing four embedded multipliers and two adders.

Figure 3-71: MULT35X35_S Submodule

The fixed adder is 53 bits wide (17 LSBs are always 0 on one input).

The 34-bit by 34-bit unsigned submodule is constructed in a similar manner with the most significant bit on each operand being tied to logic low. Table 3-28 lists multipliers and two’s-complement return functions that utilize one MULT18X18 primitive and are not registered.
Multipliers of form MULT_aXaS_bXbU use one embedded multiplier to implement two multipliers with separate outputs. The submodules listed above use optimized pin assignments to achieve shortest possible through-delay.

Figure 3-72 and Figure 3-73 represent 4-bit by 4-bit signed multiplier and 4-bit by 4-bit unsigned multiplier implementations, respectively.

<table>
<thead>
<tr>
<th>Submodule</th>
<th>A width</th>
<th>B width</th>
<th>P width</th>
<th>Signed/Unsigned</th>
</tr>
</thead>
<tbody>
<tr>
<td>MULT17X17_U</td>
<td>17</td>
<td>17</td>
<td>34</td>
<td>Unsigned</td>
</tr>
<tr>
<td>MULT8X8_S</td>
<td>8</td>
<td>8</td>
<td>16</td>
<td>Signed</td>
</tr>
<tr>
<td>MULT8X8_U</td>
<td>8</td>
<td>8</td>
<td>16</td>
<td>Unsigned</td>
</tr>
<tr>
<td>MULT4X4_S</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>Signed</td>
</tr>
<tr>
<td>MULT4X4_U</td>
<td>4</td>
<td>4</td>
<td>8</td>
<td>Unsigned</td>
</tr>
<tr>
<td>MULT_6X6S_5X5U</td>
<td>6</td>
<td>5</td>
<td>12</td>
<td>Signed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unsigned</td>
</tr>
<tr>
<td>MULT_5X5S_6X6U</td>
<td>5</td>
<td>6</td>
<td>10</td>
<td>Signed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Unsigned</td>
</tr>
<tr>
<td>MULT_5X5U_5X5U</td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>Unsigned</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>10</td>
<td>Unsigned</td>
</tr>
<tr>
<td>MULT_4X4S_7X7U</td>
<td>4</td>
<td>7</td>
<td>8</td>
<td>Signed</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>7</td>
<td>14</td>
<td>Unsigned</td>
</tr>
<tr>
<td>MULT_4X4S_3X3S</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>Signed</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>Signed</td>
</tr>
<tr>
<td>TWOS_CMP18</td>
<td>18</td>
<td>-</td>
<td>18</td>
<td>-</td>
</tr>
<tr>
<td>TWOS_CMP9</td>
<td>9</td>
<td>-</td>
<td>9</td>
<td>-</td>
</tr>
<tr>
<td>MAGNTD_18</td>
<td>18</td>
<td>-</td>
<td>17</td>
<td>-</td>
</tr>
</tbody>
</table>
Submodule MAGNTD_18 performs a magnitude return (i.e., absolute value) of a two’s-complement number. An incoming negative number returns with a positive number, while an incoming positive number remains unchanged. Submodules TWOS_CMP18 and TWOS_CMP9 perform a two’s-complement return function. The incoming number in two’s-complement form (either signed or unsigned) is complemented when the DO_COMP pin is asserted High. Additional slice logic can be used with these submodules.
to efficiently convert sign-magnitude to two’s-complement or vice-versa. Figure 3-74 shows the connections to a MULT18X18 to create the submodule TWOS_CMP9.

VHDL and Verilog Instantiation

VHDL and Verilog instantiation templates are available in the Libraries Guide for all primitives and submodules.

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signals names.

Port Signals

Data In - A

The data input provides new data (up to 18 bits) to be used as one of the multiplication operands.

Data In - B

The data input provides new data (up to 18 bits) to be used as one of the multiplication operands.

Data Out - P

The data output bus P provides the data value (up to 36 bits) of two’s-complement multiplication for operands A and B.

Location Constraints

Each embedded multiplier has location coordinates of the form XrowYcolumn. To constrain placement, multiplier instances can have LOC properties attached to MULT18X18 embedded multiplier instances can have LOC properties attached to them to constrain placement. MULT18X18 placement locations differ from the convention used for naming CLB locations, allowing LOC properties to transfer easily from array to array.

The LOC properties use the following form:

LOC = MULT18X18_X#Y#
For example, MULT18X18_X0Y0 is the bottom-left MULT18X18 location on the device.

Routing with BlockRAM

The following input pins are shared among adjacent BlockRAM and multiplier:

- RAMB16 MULT18x18
- DIA16 A0
- DIA17 A1
- DIA18 A2
- DIA19 A3
- DIA20 A4
- DIA21 A5
- DIA22 A6
- DIA23 A7
- DIA24 A8
- DIA25 A9
- DIA26 A10
- DIA27 A11
- DIA28 A12
- DIA29 A13
- DIA30 A14
- DIA31 A15
- DIB16 B0
- DIB17 B1
- DIB18 B2
- DIB19 B3
- DIB20 B4
- DIB21 B5
- DIB22 B6
- DIB23 B7
- DIB24 B8
- DIB25 B9
- DIB26 B10
- DIB27 B11
- DIB28 B12
- DIB29 B13
- DIB30 B14
- DIB31 B15

If the above BlockRAM and multiplier pins do not have common source, the implementation tools will not place the blocks adjacently.
VHDL and Verilog Templates

VHDL and Verilog templates are available for the primitive and submodules. The following is a template for the primitive:

- SIGNED_MULT_18X18 (primitive: MULT18X18)

The following are templates for submodules:

- SIGNED_MULT_35X35 (submodule: MULT35X35_S)
- UNSIGNED_MULT_34X34 (submodule: MULT34X34_U)
- UNSIGNED_MULT_17X17 (submodule: MULT17X17_U)
- SIGNED_MULT_8X8 (submodule: MULT8X8_S)
- UNSIGNED_MULT_8X8 (submodule: MULT8X8_U)
- SIGNED_MULT_4X4 (submodule: MULT4X4_S)
- UNSIGNED_MULT_4X4 (submodule: MULT4X4_U)
- TWOS_COMPLEMENTS_18BIT (submodule: TWOS_CMP18)
- TWOS_COMPLEMENTS_9BIT (submodule: TWOS_CMP9)
- MAGNITUDE_18BIT (submodule: MAGNTD_18)

The corresponding submodules have to be synthesized with the design. Templates for the SIGNED_MULT_18X18 module are provided in VHDL and Verilog code as an example.

**VHDL Template:**

```vhdl
-- Module: SIGNED_MULT_18X18
-- Description: VHDL instantiation template
-- 18-bit X 18-bit embedded signed multiplier (asynchronous)
-- Device: Virtex-II Pro Family
---------------------------------------------------------------------
-- Components Declarations
component MULT18X18
port(
    A : in std_logic_vector (17 downto 0);
    B : in std_logic_vector (17 downto 0);
    P : out std_logic_vector (35 downto 0)
);
end component;
-- Architecture Section
--
U_MULT18X18 : MULT18X18
  port map (  
    A => , -- insert input signal #1  
    B => , -- insert input signal #2  
    P =>   -- insert output signal
  );
```

**Verilog Template:**

```verilog
// Module: SIGNED_MULT_18X18
```
Single-Ended SelectIO-Ultra Resources

Summary

The Virtex-II Pro/Virtex-II Pro X FPGA Series includes a highly configurable, high-performance single-ended SelectIO™-Ultra resource that supports a wide variety of I/O standards. The SelectIO-Ultra resource includes a robust set of features, including programmable control of output drive strength, slew rate, and input delay and hold time. Taking advantage of the flexibility of SelectIO-Ultra features and the design considerations described in this document can improve and simplify system-level design.

Introduction

As FPGAs continue to grow in size and capacity, the larger and more complex systems designed for them demand an increased variety of I/O standards. Furthermore, as system clock speeds continue to increase, the need for high-performance I/O becomes more important. Chip-to-chip delays have an increasingly substantial impact on overall system speed. The task of achieving the desired system performance is becoming more difficult with the proliferation of low-voltage I/O standards. SelectIO-Ultra resolves this potential problem by providing a highly configurable, high-performance alternative to I/O resources used in more conventional programmable devices.

Virtex-II Pro/Virtex-II Pro X SelectIO-Ultra blocks can support up to 22 single-ended I/O standards. Supporting such a variety of I/O standards allows support for a wide variety of applications.

Each Input/Output Block (IOB) includes six registers, two each from the input, output, and 3-state signals within the IOB. These registers are optionally configured as either a D-type flip-flop or as a level-sensitive latch. The purpose of having six registers is to allow designers to design double-data-rate (DDR) logic in the I/O blocks. Each pair of the flip-flop (FF) has different clocks so that the flip-flops can be driven by two clocks with a 180-degree phase shift to achieve DDR. All I/O flip-flops still share the same reset/preset line.

The input buffer has an optional delay element used to guarantee a zero hold time requirement for input signals registered within the IOB.

Virtex-II Pro/Virtex-II Pro X SelectIO-Ultra features also provide dedicated resources for input reference voltage (VREF) and input output source voltage (VCCO), along with a convenient banking system that simplifies board design. Virtex-II Pro/Virtex-II Pro X CMOS inputs and outputs are powered from VCCO. Differential amplifier inputs, such as GTL and SSTL, are powered from VCCAUX.
Fundamentals

Modern bus applications, pioneered by the largest and most influential components in the digital electronics industry, are commonly introduced with a new I/O standard tailored specifically to the needs of that application. The bus I/O standards provide specifications to other vendors who create products designed to interface with these applications. Each standard often has its own specifications for current, voltage, I/O buffering, and termination techniques.

The ability to provide the flexibility and time-to-market advantages of programmable logic is increasingly dependent on the capability of the programmable logic device to support an ever increasing variety of I/O standards.

SelectIO-Ultra resources feature highly configurable input and output buffers that provide support for a wide variety of I/O standards. An input buffer can be configured as either a simple buffer or as a differential amplifier input. An output buffer can be configured as either a push-pull output or as an open drain output. Table 3-29 illustrates all of the supported single-ended I/O standards in Virtex-II Pro / Virtex-II Pro X devices. Each buffer type can support a variety of current and voltage requirements.

Table 3-29: Supported Single-Ended I/O Standards

<table>
<thead>
<tr>
<th>I/O Standard</th>
<th>Input Reference Voltage ( (V_{\text{REF}}) )</th>
<th>Input Source Voltage ( (V_{\text{CCO}}) )</th>
<th>Output Source Voltage ( (V_{\text{CCO}}) )</th>
<th>Board Termination Voltage ( (V_{\text{TT}}) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>LV TTL</td>
<td>N/A</td>
<td>3.3</td>
<td>3.3</td>
<td>N/A</td>
</tr>
<tr>
<td>LV CMOS 33</td>
<td>N/A</td>
<td>3.3</td>
<td>3.3</td>
<td>N/A</td>
</tr>
<tr>
<td>LV CMOS 25</td>
<td>N/A</td>
<td>2.5</td>
<td>2.5</td>
<td>N/A</td>
</tr>
<tr>
<td>LV CMOS 18</td>
<td>N/A</td>
<td>1.8</td>
<td>1.8</td>
<td>N/A</td>
</tr>
<tr>
<td>LV CMOS 15</td>
<td>N/A</td>
<td>1.5</td>
<td>1.5</td>
<td>N/A</td>
</tr>
<tr>
<td>PCI 33.3</td>
<td>N/A</td>
<td>Note (1)</td>
<td>Note (1)</td>
<td>N/A</td>
</tr>
<tr>
<td>PCI 66.3</td>
<td>N/A</td>
<td>Note (1)</td>
<td>Note (1)</td>
<td>N/A</td>
</tr>
<tr>
<td>PCI-X</td>
<td>N/A</td>
<td>Note (1)</td>
<td>Note (1)</td>
<td>N/A</td>
</tr>
<tr>
<td>GTL</td>
<td>0.8</td>
<td>Note (2)</td>
<td>Note (2)</td>
<td>1.2</td>
</tr>
<tr>
<td>GTLP</td>
<td>1.0</td>
<td>Note (2)</td>
<td>Note (2)</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL I</td>
<td>0.75</td>
<td>N/A</td>
<td>1.5</td>
<td>0.75</td>
</tr>
<tr>
<td>HSTL II</td>
<td>0.75</td>
<td>N/A</td>
<td>1.5</td>
<td>0.75</td>
</tr>
<tr>
<td>HSTL III</td>
<td>0.9</td>
<td>N/A</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL IV</td>
<td>0.9</td>
<td>N/A</td>
<td>1.5</td>
<td>1.5</td>
</tr>
<tr>
<td>HSTL I 18</td>
<td>0.9</td>
<td>N/A</td>
<td>1.8</td>
<td>0.9</td>
</tr>
<tr>
<td>HSTL II 18</td>
<td>0.9</td>
<td>N/A</td>
<td>1.8</td>
<td>0.9</td>
</tr>
<tr>
<td>HSTL III 18</td>
<td>1.08</td>
<td>N/A</td>
<td>1.8</td>
<td>1.8</td>
</tr>
<tr>
<td>HSTL IV 18</td>
<td>1.08</td>
<td>N/A</td>
<td>1.8</td>
<td>1.8</td>
</tr>
<tr>
<td>SSTL2 I</td>
<td>1.25</td>
<td>N/A</td>
<td>2.5</td>
<td>1.25</td>
</tr>
<tr>
<td>SSTL2 II</td>
<td>1.25</td>
<td>N/A</td>
<td>2.5</td>
<td>1.25</td>
</tr>
</tbody>
</table>
Overview of Supported I/O Standards

This section provides a brief overview of I/O standards supported by all Virtex-II Pro / Virtex-II Pro X devices.

While most I/O standards specify a range of allowed voltages, this document records typical voltage values only. Detailed information on each specification can be found on the Electronic Industry Alliance JEDEC website at http://www.jedec.org.

LVTTL - Low-Voltage TTL

The low-voltage TTL, or LVTTL, standard is a general purpose EIA/JESDSA standard for 3.3V applications that use an LVTTL input buffer and a push-pull output buffer. This standard requires a 3.3V input and output source voltage (VCCO), but does not require the use of a reference voltage (VREF) or a termination voltage (VTT).

LVCMOS33 - 3.3-Volt Low-Voltage CMOS

This standard is an extension of the LVCMOS standard (JESD 8.-5). It is used in general-purpose 3.3V applications. The standard requires a 3.3V input/output source voltage (VCCO), but does not require a reference voltage (VREF) or termination voltage (VTT).

LVCMOS25 - 2.5-Volt Low-Voltage CMOS

This standard is an extension of the LVCMOS standard (JESD 8.-5). It is used in general purpose 2.5 volts or lower applications. This standard requires a 2.5V input/output source voltage (VCCO), but does not require the use of a reference voltage (VREF) or a board termination voltage (VTT).

LVCMOS18 - 1.8-Volt Low-Voltage CMOS

This standard is an extension of the LVCMOS standard. It is used in general purpose 1.8V applications. The use of a reference voltage (VREF) or board termination voltage (VTT) is not required.

LVCMOS15 - 1.5-Volt Low-Voltage CMOS

This standard is an extension of the LVCMOS standard. It is used in general purpose 1.5V applications. The use of a reference voltage (VREF) or a board termination voltage (VTT) is not required.

---

Table 3-29:  **Supported Single-Ended I/O Standards (Continued)**

<table>
<thead>
<tr>
<th>I/O Standard</th>
<th>Input Reference Voltage (VREF)</th>
<th>Input Source Voltage (VCCO)</th>
<th>Output Source Voltage (VCCO)</th>
<th>Board Termination Voltage (VTT)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTL18_I (3)</td>
<td>0.9</td>
<td>N/A</td>
<td>1.8</td>
<td>0.9</td>
</tr>
<tr>
<td>SSTL18_II</td>
<td>0.9</td>
<td>N/A</td>
<td>1.8</td>
<td>0.9</td>
</tr>
</tbody>
</table>

Notes:
1. For PCI and PCI-X standards, refer to XAPP653.
2. VCCO of GTL or GTLP should not be lower than the termination voltage or the voltage seen at the I/O pad. For example, connect VCCO to 1.5V if the pin High level is 1.5V
3. SSTL18_I is not a JEDEC-supported standard.
PCI - Peripheral Component Interface

The PCI standard specifies support for 33 MHz, 66 MHz, and 133 MHz PCI bus applications. It uses an LVTTL input buffer and a push-pull output buffer. This standard does not require the use of a reference voltage ($V_{REF}$) or a board termination voltage ($V_{TT}$). However, it does require 3.3V input/output source voltage ($V_{CCO}$).

For further information on interfacing a Virtex-II Pro / Virtex-II Pro X device to a PCI bus, see the following Xilinx application notes. The information contained in both these application notes must be taken into account when implementing PCI33_3, PCI66_3, and PCI-X.

- XAPP653, Virtex-II Pro 3.3V PCI Reference Design
- XAPP646, Connecting Virtex-II Devices to a 3.3V/5V PCI Bus

Note: XAPP653 is recommended for PCI applications, as it is PCI-compliant and more cost-effective than XAPP646.

GTL - Gunning Transceiver Logic Terminated

The GTL standard is a high-speed bus standard (JESD8.3) invented by Xerox. Xilinx has implemented the terminated variation for this standard. This standard requires a differential amplifier input buffer and a open Drain output buffer.

GTLP - Gunning Transceiver Logic Plus

The Gunning Transceiver Logic Plus, or GTLP standard is a high-speed bus standard (JESD8.3) first used by the Pentium Pro Processor.

HSTL - High-Speed Transceiver Logic

The high-speed Transceiver Logic, or HSTL standard is a general purpose high-speed, 1.5V bus standard sponsored by IBM (EIA/JESD8-6). This standard has four variations or classes. Virtex-II Pro / Virtex-II Pro X SelectIO-Ultra supports all four classes. This standard requires a differential amplifier input buffer and a push-pull output buffer.

SSTL2 - Stub Series Terminated Logic for 2.5V

The Stub Series Terminated Logic for 2.5V, or SSTL2 standard is a general purpose 2.5V memory bus standard also sponsored by Hitachi and IBM (JESD8-8). This standard has two classes, I and II. Virtex-II Pro / Virtex-II Pro X SelectIO-Ultra supports both classes for the SSTL2 standard. This standard requires a differential amplifier input buffer and a push-pull output buffer.

SSTL18 - Stub Series Terminated Logic for 1.8V

The Stub Series Terminated Logic for 1.8V, or SSTL18 is a general purpose 1.8V memory bus standard. Similar to SSTL2, it has two classes, I and II. (Only Class II is supported by JESD8-15.) Virtex-II Pro / Virtex-II Pro X SelectIO-Ultra supports both classes for the SSTL18 standard. This standard requires a differential amplifier input buffer and a push-pull output buffer.
Library Symbols

The Xilinx library includes an extensive list of symbols designed to provide support for the variety of SelectIO-Ultra features. Most of these symbols represent variations of the five generic SelectIO-Ultra symbols.

- IBUF (input buffer)
- IBUFG (clock input buffer)
- OBUF (output buffer)
- OBUFT (3-state output buffer)
- IOBUF (input/output buffer)

IBUF

Signals used as inputs to a Virtex-II Pro / Virtex-II Pro X device must source an input buffer (IBUF) via an external input port. The generic IBUF symbol is shown in Figure 3-75. The extension to the base name defines which I/O standard the IBUF uses. The assumed standard is LVCMOS when the generic IBUF has no specified extension.

![IBUF Symbol](x133_01_111699)

*Figure 3-75: Input Buffer (IBUF) Symbols*

Table 3-30 details variations of the IBUF symbol for single-ended Virtex-II Pro / Virtex-II Pro X I/O standards:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Symbol</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBUF</td>
<td>IBUF_GTL</td>
<td>IBUF_SSTL2_I</td>
</tr>
<tr>
<td>IBUF_LVCMOS15</td>
<td>IBUF_GTLP</td>
<td>IBUF_SSTL2_II</td>
</tr>
<tr>
<td>IBUF_LVCMOS18</td>
<td>IBUF_HSTL_I</td>
<td>IBUF_PCI33_3</td>
</tr>
<tr>
<td>IBUF_LVCMOS25</td>
<td>IBUF_HSTL_II</td>
<td>IBUF_PCI66_3</td>
</tr>
<tr>
<td>IBUF_LVCMOS33</td>
<td>IBUF_HSTL_III</td>
<td>IBUF_PCIE</td>
</tr>
<tr>
<td></td>
<td></td>
<td>IBUF_HSTL_IV</td>
</tr>
</tbody>
</table>

When the IBUF symbol supports an I/O standard that requires a differential amplifier input, the IBUF is automatically configured as a differential amplifier input buffer. The low-voltage I/O standards with a differential amplifier input require an external reference voltage input $V_{REF}$.

The voltage reference signal is “banked” within the Virtex-II Pro / Virtex-II Pro X device on a half-edge basis, such that, for all packages, there are eight independent $V_{REF}$ banks internally. For a representation of the I/O banks, see Figure 3-76 and Figure 3-77.

Within each bank approximately one of every six I/O pins is automatically configured as a $V_{REF}$ input. After placing a differential amplifier input signal within a given $V_{REF}$ bank, the same external source must drive all I/O pins configured as a $V_{REF}$ input.
IBUF placement restrictions require that any differential amplifier input signals within a bank be of the same standard. How to specify a specific location for the IBUF via the LOC property is described below. Table 3-31 summarizes compatibility requirements of Virtex-II Pro / Virtex-II Pro X input standards.

An optional delay element in the input data path is associated with each IBUF. When the IBUF drives a flip-flop within the IOB, the delay element is activated by default to ensure a zero hold-time requirement at the device input pin. The IOBDELAY = NONE property overrides this default, reducing the input set-up time, but risking a hold-time requirement.

When the IBUF does not drive a flip-flop within the IOB, the delay element is deactivated by default to provide a shorter input set-up time. To delay the input signal, activate the delay element with the IOBDELAY = BOTH property.

![IBUF placement in Virtex-II Pro and Virtex-II Pro X FPGA](image)

**Figure 3-76:** I/O Banks: Top View for Flip-Chip Packages (FF)

![IBUF placement in Virtex-II Pro and Virtex-II Pro X FPGA](image)

**Figure 3-77:** I/O Banks: Top View for Wire-Bond Package (FG)

<table>
<thead>
<tr>
<th>Rule</th>
<th>Standards with the same $V_{CCO}$ and $V_{REF}$ can be placed within the same bank.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 2</td>
<td>Standards that don't require a $V_{REF}$ can be placed within the same bank with the standards that have the same $V_{CCO}$ values</td>
</tr>
</tbody>
</table>

**Table 3-31: Xilinx Input Standard Compatibility Requirements**
Each bank has its own $V_{CCO}$ and $V_{REF}$ voltage. Details on compatible input standards for each $V_{CCO}$ / $V_{REF}$ voltage combination are available in the Data Sheet.

OBUF

An OBUF must drive outputs through an external output port. Figure 3-78 shows the generic output buffer (OBUF) symbol.

![Output Buffer (OBUF) Symbol](image)

The extension to the base name defines which I/O standard the OBUF uses. With no extension specified for the generic OBUF symbol, the assumed standard is slew rate limited LVCMOS25 with 12mA drive strength.

The LVCMOS OBUFs can additionally support one of two slew rate modes to minimize bus transients. By default, the slew rate for each output buffer is reduced to minimize power bus transients, when switching non-critical signals.

LVCMOS output buffers have selectable drive strengths. The format for these OBUF symbol names is as follows:

```
OBUF_<slew_rate>_<drive_strength>
```

<slew_rate> is either F (fast) or S (slow) and <drive_strength> is specified in milliamperes. For LVCMOS25, the supported drive strengths are 2, 4, 6, 8, 12, 16, and 24. For LVCMOS15 and LVCMOS18, the supported drive strengths are 2, 4, 6, 8, 12, and 16.

Table 3-32 details variations of the OBUF symbol.

<table>
<thead>
<tr>
<th>OBUF</th>
<th>OBUF_LVCMOS18</th>
<th>OBUF_LVCMOS25_F_24</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBUF_S_2</td>
<td>OBUF_LVCMOS18_S_2</td>
<td>OBUF_LVCMOS33_S_4</td>
</tr>
<tr>
<td>OBUF_S_4</td>
<td>OBUF_LVCMOS18_S_4</td>
<td>OBUF_LVCMOS33_S_6</td>
</tr>
<tr>
<td>OBUF_S_6</td>
<td>OBUF_LVCMOS18_S_6</td>
<td>OBUF_LVCMOS33_S_8</td>
</tr>
<tr>
<td>OBUF_S_8</td>
<td>OBUF_LVCMOS18_S_8</td>
<td>OBUF_LVCMOS33_S_12</td>
</tr>
<tr>
<td>OBUF_S_12</td>
<td>OBUF_LVCMOS18_S_12</td>
<td>OBUF_LVCMOS33_S_16</td>
</tr>
<tr>
<td>OBUF_S_16</td>
<td>OBUF_LVCMOS18_S_16</td>
<td>OBUF_LVCMOS33_S_24</td>
</tr>
<tr>
<td>OBUF_S_24</td>
<td>OBUF_LVCMOS18_F_2</td>
<td>OBUF_LVCMOS33_F_2</td>
</tr>
<tr>
<td>OBUF_F_2</td>
<td>OBUF_LVCMOS18_F_4</td>
<td>OBUF_LVCMOS33_F_4</td>
</tr>
<tr>
<td>OBUF_F_4</td>
<td>OBUF_LVCMOS18_F_6</td>
<td>OBUF_LVCMOS33_F_6</td>
</tr>
<tr>
<td>OBUF_F_6</td>
<td>OBUF_LVCMOS18_F_8</td>
<td>OBUF_LVCMOS33_F_8</td>
</tr>
<tr>
<td>OBUF_F_8</td>
<td>OBUF_LVCMOS18_F_12</td>
<td>OBUF_LVCMOS33_F_12</td>
</tr>
<tr>
<td>OBUF_F_12</td>
<td>OBUF_LVCMOS18_F_16</td>
<td>OBUF_LVCMOS33_F_16</td>
</tr>
</tbody>
</table>
Table 3-32: Variations of the OBUF Symbol  (Continued)

<table>
<thead>
<tr>
<th>OBUF_F_16</th>
<th>OBUF_LVCMOS25</th>
<th>OBUF_LVCMOS33_F_24</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBUF_F_24</td>
<td>OBUF_LVCMOS25_S_2</td>
<td>OBUF_PCI33_3</td>
</tr>
<tr>
<td>OBUF_LVCMOS15</td>
<td>OBUF_LVCMOS25_S_4</td>
<td>OBUF_PCI66-3</td>
</tr>
<tr>
<td>OBUF_LVCMOS15_S_2</td>
<td>OBUF_LVCMOS25_S_6</td>
<td>OBUF_PCIIX</td>
</tr>
<tr>
<td>OBUF_LVCMOS15_S_4</td>
<td>OBUF_LVCMOS25_S_8</td>
<td>OBUF_GTL</td>
</tr>
<tr>
<td>OBUF_LVCMOS15_S_6</td>
<td>OBUF_LVCMOS25_S_12</td>
<td>OBUF_GTPL</td>
</tr>
<tr>
<td>OBUF_LVCMOS15_S_8</td>
<td>OBUF_LVCMOS25_S_16</td>
<td>OBUF_HSTL_I</td>
</tr>
<tr>
<td>OBUF_LVCMOS15_S_12</td>
<td>OBUF_LVCMOS25_S_24</td>
<td>OBUF_HSTL_II</td>
</tr>
<tr>
<td>OBUF_LVCMOS15_S_16</td>
<td>OBUF_LVCMOS25_F_2</td>
<td>OBUF_HSTL_III</td>
</tr>
<tr>
<td>OBUF_LVCMOS15_F_2</td>
<td>OBUF_LVCMOS25_F_4</td>
<td>OBUF_HSTL_IV</td>
</tr>
<tr>
<td>OBUF_LVCMOS15_F_4</td>
<td>OBUF_LVCMOS25_F_6</td>
<td>OBUF_SSTL18_1</td>
</tr>
<tr>
<td>OBUF_LVCMOS15_F_6</td>
<td>OBUF_LVCMOS25_F_8</td>
<td>OBUF_SSTL18_II</td>
</tr>
<tr>
<td>OBUF_LVCMOS15_F_8</td>
<td>OBUF_LVCMOS25_F_12</td>
<td>OBUF_SSTL2_I</td>
</tr>
<tr>
<td>OBUF_LVCMOS15_F_12</td>
<td>OBUF_LVCMOS25_F_16</td>
<td>OBUF_SSTL2_II</td>
</tr>
<tr>
<td>OBUF_LVCMOS15_F_16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

OBUF placement restrictions require that within a given $V_{CCO}$ bank each OBUF share the same output source drive voltage. Input buffers with the same $V_{CCO}$ and output buffers that do not require $V_{CCO}$ can be placed within any $V_{CCO}$ bank. Table 3-33 summarizes Virtex-II Pro / Virtex-II Pro X output compatibility requirements. The LOC property can specify a location for the OBUF.

Table 3-33: Output Standards Compatibility Requirements

<table>
<thead>
<tr>
<th>Rule</th>
<th>Requirement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 1</td>
<td>Only outputs with standards which share compatible $V_{CCO}$ can be used within the same bank.</td>
</tr>
<tr>
<td>Rule 2</td>
<td>There are no placement restrictions for outputs with standards that do not require a $V_{CCO}$</td>
</tr>
</tbody>
</table>

Each bank has its own $V_{CCO}$ voltage. Details on compatible output standards for each $V_{CCO}$ voltage combination are available in the Data Sheet.
OBUFT

The generic 3-state output buffer OBUFT, shown in Figure 3-79, typically implements 3-state outputs or bidirectional I/O.

![OBUFT Symbol](image)

Figure 3-79: 3-State Output Buffer Symbol (OBUFT)

The extension to the base name defines which I/O standard OBUFT uses. With no extension specified for the generic OBUFT symbol, the assumed standard is slew rate limited LVCMOS with 12mA drive strength.

The LVCMOS OBUFT additionally can support one of two slew rate modes to minimize bus transients. By default, the slew rate for each output buffer is reduced to minimize power bus transients, when switching non-critical signals.

LVCMOS 3-state buffers have selectable drive strengths. The format for these OBUFT symbol names is as follows:

```
OBUFT_<slew_rate>_<drive_strength>
```

<slew_rate> is either F (fast) or S (slow) and <drive_strength> is specified in milliamperes. For LVCMOS25, the supported drive strengths are 2, 4, 6, 8, 12, 16, and 24. For LVCMOS15 and LVCMOS18, the supported drive strengths are 2, 4, 6, 8, 12, and 16. Table 3-34 details variations of the OBUFT symbol.

Table 3-34: Variations of the OBUFT Symbol

<table>
<thead>
<tr>
<th>OBUFT</th>
<th>OBUFT_LVCMOS18</th>
<th>OBUFT_LVCMOS25_F_24</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBUFT_S_2</td>
<td>OBUFT_LVCMOS18_S_2</td>
<td>OBUFT_LVCMOS33</td>
</tr>
<tr>
<td>OBUFT_S_4</td>
<td>OBUFT_LVCMOS18_S_4</td>
<td>OBUFT_LVCMOS33_S_2</td>
</tr>
<tr>
<td>OBUFT_S_6</td>
<td>OBUFT_LVCMOS18_S_6</td>
<td>OBUFT_LVCMOS33_S_4</td>
</tr>
<tr>
<td>OBUFT_S_8</td>
<td>OBUFT_LVCMOS18_S_8</td>
<td>OBUFT_LVCMOS33_S_6</td>
</tr>
<tr>
<td>OBUFT_S_12</td>
<td>OBUFT_LVCMOS18_S_12</td>
<td>OBUFT_LVCMOS33_S_8</td>
</tr>
<tr>
<td>OBUFT_S_16</td>
<td>OBUFT_LVCMOS18_S_16</td>
<td>OBUFT_LVCMOS33_S_12</td>
</tr>
<tr>
<td>OBUFT_S_24</td>
<td>OBUFT_LVCMOS18_F_2</td>
<td>OBUFT_LVCMOS33_S_16</td>
</tr>
<tr>
<td>OBUFT_F_2</td>
<td>OBUFT_LVCMOS18_F_4</td>
<td>OBUFT_LVCMOS33_S_24</td>
</tr>
<tr>
<td>OBUFT_F_4</td>
<td>OBUFT_LVCMOS18_F_6</td>
<td>OBUFT_LVCMOS33_F_2</td>
</tr>
<tr>
<td>OBUFT_F_6</td>
<td>OBUFT_LVCMOS18_F_8</td>
<td>OBUFT_LVCMOS33_F_4</td>
</tr>
<tr>
<td>OBUFT_F_8</td>
<td>OBUFT_LVCMOS18F_12</td>
<td>OBUFT_LVCMOS33_F_6</td>
</tr>
<tr>
<td>OBUFT_F_12</td>
<td>OBUFT_LVCMOS18_F_16</td>
<td>OBUFT_LVCMOS33_F_8</td>
</tr>
<tr>
<td>OBUFT_F_16</td>
<td>OBUFT_LVCMOS25</td>
<td>OBUFT_LVCMOS33_F_12</td>
</tr>
</tbody>
</table>
ROBUFT placement restrictions require that within a given VCCO bank each OBUFT share the same output source drive voltage. Input buffers with the same VCCO and output buffers that do not require VCCO can be placed within any VCCO bank. The LOC property can specify a location for the OBUFT.

3-state output buffers and bidirectional buffers can have either a weak pull-up resistor, a weak pull-down resistor, or a weak “keeper” circuit. Control this feature by adding the appropriate symbol to the output net of the OBUFT (PULLUP, PULLDOWN, or KEEPER).

The weak “keeper” circuit requires the input buffer within the IOB to sample the I/O signal. Thus, OBUFTs programmed for an I/O standard that requires a VREF have automatic placement of a VREF in the bank with an OBUFT configured with a weak “keeper” typically implement a bidirectional I/O. In this case, the IBUF (and the corresponding VREF) are placed explicitly.

Table 3-34: Variations of the OBUFT Symbol (Continued)

<table>
<thead>
<tr>
<th>OBUFT_F_24</th>
<th>OBUFT_LVC莫斯25_S_2</th>
<th>OBUFT_LVC莫斯33_F_16</th>
</tr>
</thead>
<tbody>
<tr>
<td>OBUFT_LVCmos15</td>
<td>OBUFT_LVCmos25_S_4</td>
<td>OBUFT_LVCmos33_F_24</td>
</tr>
<tr>
<td>OBUFT_LVCmos15_S_2</td>
<td>OBUFT_LVCmos25_S_6</td>
<td>OBUFT_PCI33_3</td>
</tr>
<tr>
<td>OBUFT_LVCmos15_S_4</td>
<td>OBUFT_LVCmos25_S_8</td>
<td>OBUFT_PCI66-3</td>
</tr>
<tr>
<td>OBUFT_LVCmos15_S_6</td>
<td>OBUFT_LVCmos25_S_12</td>
<td>OBUFT_PCIIX</td>
</tr>
<tr>
<td>OBUFT_LVCmos15_S_8</td>
<td>OBUFT_LVCmos25_S_16</td>
<td>OBUFT_GTL</td>
</tr>
<tr>
<td>OBUFT_LVCmos15_S_12</td>
<td>OBUFT_LVCmos25_S_24</td>
<td>OBUFT_GTLP</td>
</tr>
<tr>
<td>OBUFT_LVCmos15_S_16</td>
<td>OBUFT_LVCmos25_F_2</td>
<td>OBUFT_HSTL_I</td>
</tr>
<tr>
<td>OBUFT_LVCmos15_F_2</td>
<td>OBUFT_LVCmos25_F_4</td>
<td>OBUFT_HSTL_II</td>
</tr>
<tr>
<td>OBUFT_LVCmos15_F_4</td>
<td>OBUFT_LVCmos25_F_6</td>
<td>OBUFT_HSTL_III</td>
</tr>
<tr>
<td>OBUFT_LVCmos15_F_6</td>
<td>OBUFT_LVCmos25_F_8</td>
<td>OBUFT_HSTL_IV</td>
</tr>
<tr>
<td>OBUFT_LVCmos15_F_8</td>
<td>OBUFT_LVCmos25_F_12</td>
<td>OBUFT_SSTL2_I</td>
</tr>
<tr>
<td>OBUFT_LVCmos15_F_12</td>
<td>OBUFT_LVCmos25_F_16</td>
<td>OBUFT_SSTL2_II</td>
</tr>
<tr>
<td>OBUFT_LVCmos15_F_16</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
IOBUF

Use the IOBUF symbol for bidirectional signals that require both an input buffer and a 3-state output buffer with an active High 3-state pin. Figure 3-80 shows the generic input/output IOBUF buffer.

![Input/Output Buffer Symbol (IOBUF)](image)

Figure 3-80: Input/Output Buffer Symbol (IOBUF)

The extension to the base name defines which I/O standard the IOBUF uses. With no extension specified for the generic IOBUF symbol, the assumed standard is LVCMOS input buffer and slew rate limited LVCMOS with 12mA drive strength for the output buffer.

The LVCMOS IOBUF can additionally support one of two slew rate modes to minimize bus transients. By default, the slew rate for each output buffer is reduced to minimize power bus transients, when switching non-critical signals.

LVCMOS output buffers have selectable drive strengths. The format for these IOBUF symbol names is as follows:

`IOBUF_<slew_rate>_<_drive_strength>`

`<slew_rate>` is either F (fast) or S (slow) and `<drive_strength>` is specified in milliamperes. For LVCMOS25, the supported drive strengths are 2, 4, 6, 8, 12, 16, and 24. For LVCMOS15 and LVCMOS18, the supported drive strengths are 2, 4, 6, 8, 12, and 16. Table 3-35 details variations of the IOBUF symbol.

### Table 3-35: Variations of the IOBUF Symbol

<table>
<thead>
<tr>
<th>IOBUF</th>
<th>IOBUF_LVCMOS18</th>
<th>IOBUF_LVCMOS33</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBUF</td>
<td>IOBUF_LVCMOS18</td>
<td>IOBUF_LVCMOS33</td>
</tr>
<tr>
<td>IOBUF_S_2</td>
<td>IOBUF_LVCMOS18_S_2</td>
<td>IOBUF_LVCMOS33_S_2</td>
</tr>
<tr>
<td>IOBUF_S_4</td>
<td>IOBUF_LVCMOS18_S_4</td>
<td>IOBUF_LVCMOS33_S_4</td>
</tr>
<tr>
<td>IOBUF_S_6</td>
<td>IOBUF_LVCMOS18_S_6</td>
<td>IOBUF_LVCMOS33_S_6</td>
</tr>
<tr>
<td>IOBUF_S_8</td>
<td>IOBUF_LVCMOS18_S_8</td>
<td>IOBUF_LVCMOS33_S_8</td>
</tr>
<tr>
<td>IOBUF_S_12</td>
<td>IOBUF_LVCMOS18_S_12</td>
<td>IOBUF_LVCMOS33_S_12</td>
</tr>
<tr>
<td>IOBUF_S_16</td>
<td>IOBUF_LVCMOS18_S_16</td>
<td>IOBUF_LVCMOS33_S_16</td>
</tr>
<tr>
<td>IOBUF_S_24</td>
<td>IOBUF_LVCMOS18_F_2</td>
<td>IOBUF_LVCMOS33_S_24</td>
</tr>
<tr>
<td>IOBUF_F_2</td>
<td>IOBUF_LVCMOS18_F_2</td>
<td>IOBUF_LVCMOS33_F_2</td>
</tr>
<tr>
<td>IOBUF_F_4</td>
<td>IOBUF_LVCMOS18_F_4</td>
<td>IOBUF_LVCMOS33_F_4</td>
</tr>
<tr>
<td>IOBUF_F_6</td>
<td>IOBUF_LVCMOS18_F_8</td>
<td>IOBUF_LVCMOS33_F_6</td>
</tr>
</tbody>
</table>
When the IOBUF symbol supports an I/O standard that requires a differential amplifier input, IOBUF is automatically configured as a differential amplifier input buffer. Low-voltage I/O standards with a differential amplifier input require an external reference voltage input VREF.

The voltage reference signal is “banked” within the Virtex-II Pro / Virtex-II Pro X device on a half-edge basis, such that for all packages there are eight independent VREF banks internally. For a representation of the I/O banks, see Figure 3-77. Within each bank approximately one of every twelve I/O pins is automatically configured as a VREF input. After placing a differential amplifier input signal within a given VREF bank, the same external source must drive all I/O pins configured as a VREF input.

IOBUF placement restrictions require any differential amplifier input signals within a bank be of the same standard.

Additional restrictions on Virtex-II Pro / Virtex-II Pro X SelectIO-Ultra IOBUF placement require that within a given VCCO bank each IOBUF share the same output source drive voltage. Input buffers with the same VCCO and output buffers that do not require VCCO can be placed within any VCCO bank. The LOC property can specify a location for the IOBUF.

An optional delay element is associated with the input path in each IOBUF. When the IOBUF drives an input flip-flop within the IOB, the delay element is activated by default to ensure the zero hold-time requirement. Override this default with the IOBDELAY = NONE property.

<table>
<thead>
<tr>
<th>IOBUF_SYMBOL</th>
<th>VREF_SYMBOL</th>
<th>VCCO_SYMBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>IOBUF_F_8</td>
<td>IOBUF_LVCMOS18F_12</td>
<td>IOBUF_LVCMOS33_F_8</td>
</tr>
<tr>
<td>IOBUF_F_12</td>
<td>IOBUF_LVCMOS18_F_16</td>
<td>IOBUF_LVCMOS33_F_12</td>
</tr>
<tr>
<td>IOBUF_F_16</td>
<td>IOBUF_LVCMOS25</td>
<td>IOBUF_LVCMOS33_F_16</td>
</tr>
<tr>
<td>IOBUF_F_24</td>
<td>IOBUF_LVCMOS25_S_2</td>
<td>IOBUF_LVCMOS33_F_24</td>
</tr>
<tr>
<td>IOBUF_LVCMOS15</td>
<td>IOBUF_LVCMOS25_S_4</td>
<td>IOBUF_PCI133_3</td>
</tr>
<tr>
<td>IOBUF_LVCMOS15_S_2</td>
<td>IOBUF_LVCMOS25_S_6</td>
<td>IOBUF_PCI166-3</td>
</tr>
<tr>
<td>IOBUF_LVCMOS15_S_4</td>
<td>IOBUF_LVCMOS25_S_8</td>
<td>IOBUF_PCI1X</td>
</tr>
<tr>
<td>IOBUF_LVCMOS15_S_6</td>
<td>IOBUF_LVCMOS25_S_12</td>
<td>IOBUF_GTL</td>
</tr>
<tr>
<td>IOBUF_LVCMOS15_S_8</td>
<td>IOBUF_LVCMOS25_S_16</td>
<td>IOBUF_GTLP</td>
</tr>
<tr>
<td>IOBUF_LVCMOS15_S_12</td>
<td>IOBUF_LVCMOS25_S_24</td>
<td>IOBUF_HSTL_I</td>
</tr>
<tr>
<td>IOBUF_LVCMOS15_S_16</td>
<td>IOBUF_LVCMOS25_F_2</td>
<td>IOBUF_HSTL_II</td>
</tr>
<tr>
<td>IOBUF_LVCMOS15_S_2</td>
<td>IOBUF_LVCMOS25_S_6</td>
<td>IOBUF_HSTL_III</td>
</tr>
<tr>
<td>IOBUF_LVCMOS15_S_4</td>
<td>IOBUF_LVCMOS25_F_6</td>
<td>IOBUF_HSTL_IV</td>
</tr>
<tr>
<td>IOBUF_LVCMOS15_S_6</td>
<td>IOBUF_LVCMOS25_F_8</td>
<td>IOBUF_SSTL18_I</td>
</tr>
<tr>
<td>IOBUF_LVCMOS15_S_8</td>
<td>IOBUF_LVCMOS25_F_12</td>
<td>IOBUF_SSTL18_II</td>
</tr>
<tr>
<td>IOBUF_LVCMOS15_S_12</td>
<td>IOBUF_LVCMOS25_F_16</td>
<td>IOBUF_SSTL2_1</td>
</tr>
<tr>
<td>IOBUF_LVCMOS15_S_16</td>
<td>IOBUF_LVCMOS25_F_24</td>
<td>IOBUF_SSTL2_II</td>
</tr>
</tbody>
</table>
In the case when the IOBUF does not drive an input flip-flop within the IOB, the delay element is deactivated by default to provide higher performance. To delay the input signal, activate the delay element with the IOBDELAY = BOTH property.

3-state output buffers and bidirectional buffers can have a weak pull-up resistor, a weak pull-down resistor, or a weak “keeper” circuit. Control this feature by adding the appropriate symbol to the output net of the IOBUF (PULLUP, PULLDOWN, or KEEPER).

**SelectIO-Ultra Properties**

Access to some SelectIO-Ultra features (for example, location constraints, input delay, output drive strength, and slew rate) is available through properties associated with these features.

**Input Delay Properties**

An optional delay element is associated with the input path in each IBUF. When the IBUF drives an input flip-flop within the IOB, the delay element activates by default to ensure the zero hold-time requirement. Override this default with the IOBDELAY = NONE property.

In the case when the IBUF does not drive an input flip-flop within the IOB, the delay element is deactivated by default to provide higher performance. To delay the input signal, activate the delay element with the IOBDELAY = BOTH property.

**IOB Flip-Flop/Latch Properties**

The Virtex-II Pro / Virtex-II Pro X Series I/O block (IOB) includes two optional registers on the input path, two optional registers on the output path, and two optional registers on the 3-state control pin. The design implementation software automatically takes advantage of these registers when the following option for the MAP program is specified.

```
Map -pr b <filename>
```

Alternatively, the IOB = TRUE property can be placed on a register to force the mapper to place the register in an IOB.

The two registers for each path makes designing double-data-rate (DDR) logic much simpler. Each pair of the registers has separate clock inputs, which can be driven by either the positive edge or the negative edge of the clock. Users can use both edges of the clocks to clock data in and out from the IOB. For details on DDR, see “Double-Data-Rate (DDR) I/O,” page 241.

**Location Constraints**

Specify the location of each SelectIO-Ultra symbol with the location constraint LOC attached to the SelectIO-Ultra symbol. The external port identifier indicates the value of the location constrain. The format of the port identifier depends on the package chosen for the specified design.

The LOC properties use the following form:

- LOC=A42;
- LOC=P37;
Output Slew Rate Property

As mentioned above, a variety of symbol names provide the option of choosing the desired slew rate for the output buffers. In the case of the LVCMOS output buffers (OBUF, OBUFT, and IOBUF), slew rate control can be alternatively programmed with the SLEW = property. By the default, the slew rate for each output buffer is reduced to minimize power bus transients when switching non-critical signals. The SLEW = property has one of the two following values:

- SLEW = SLOW
- SLEW = FAST

Output Drive Strength Property

The desired output drive strength can be additionally specified by choosing the appropriate library symbol. The Xilinx library also provides an alternative method for specifying this feature. For the LVCMOS output buffers (OBUF, OBUFT, and IOBUF), the desired drive strength can be specified with the DRIVE = property. This property could have one of the following values:

- DRIVE = 2
- DRIVE = 4
- DRIVE = 6
- DRIVE = 8
- DRIVE = 12
- DRIVE = 16
- DRIVE = 24

Design Considerations

Reference Voltage (V_{REF}) Pins

Low-voltage I/O standards with a differential amplifier input buffer require an input reference voltage (V_{REF}). Provide the V_{REF} as an external signal to the device.

The voltage reference signal is “banked” within the Virtex-II Pro/Virtex-II Pro X device on a half-edge basis such that for all packages there are eight independent V_{REF} banks internally. See Figure 3-77 for a representation of the I/O banks. Within each bank approximately one of every twelve I/O pins is automatically configured as a V_{REF} input. After placing a differential amplifier input signal within a given V_{REF} bank, the same external source must drive all I/O pins configured as a V_{REF} input.

Within each V_{REF} bank, any input buffers that require a V_{REF} signal must be of the same type. Output buffers that have the same V_{CCO} values as the input buffers can be placed within the same V_{REF} bank.

Output Drive Source Voltage (V_{CCO}) Pins

Many of the low-voltage I/O standards supported by SelectIO-Ultra devices require a different output drive source voltage (V_{CCO}). As a result each device can often have to support multiple output drive source voltages.

Output buffers within a given V_{CCO} bank must share the same output drive source voltage. Input buffers for LVTTL, LVCMOS15, LVCMOS18, LVCMOS25, LVCMOS33, PCI33_3, PCI66_3, and PCI-X use the V_{CCO} voltage for input V_{CCO} voltage.
It is recommended to power $V_{CCO}$ even for non-$V_{CCO}$ dependent input standards such as LVDS, LVPECL, SSTL, HSTL, and GTL. The input clamp diodes, which are always present regardless of I/O standard, become forward-biased if pin voltage exceeds $V_{CCO}$ by the diode threshold voltage (~0.5V). Leaving $V_{CCO}$ disconnected, or setting it to an excessively low voltage, can lead to a distorted input signal. To avoid this, $V_{CCO}$ should be greater than the input high voltage (plus any overshoot), less the 0.5V diode voltage.

Furthermore, if a bank is unused, tie $V_{CCO}$ to $V_{CCAUX}$ or any available $V_{CCO}$. Do not leave it floating or connected to ground.

Transmission Line Effects

The delay of an electrical signal along a wire is dominated by the rise and fall times when the signal travels a short distance. Transmission line delays vary with inductance and capacitance. But a well-designed board can experience delays of approximately 180 ps per inch. Transmission line effects, or reflections, typically start at 1.5” for fast (1.5 ns) rise and fall times. Poor (or non-existent) termination or changes in the transmission line impedance cause these reflections and can cause additional delay in longer traces. As a system speeds continue to increase, the effect of I/O delays can become a limiting factor and therefore transmission line termination becomes increasingly more important.

Termination Techniques

A variety of termination techniques reduce the impact of transmission line effects.

The following are output termination techniques:

- None
- Series
- Parallel (Shunt)
- Series and Parallel (Series-Shunt)

The following are input termination techniques:

- None
- Parallel (Shunt)
These termination techniques can be applied in any combination. A generic example of each combination of termination methods appears in Figure 3-81.

**Figure 3-81: Overview of Standard Input and Output Termination Methods**

Simultaneous Switching Guidelines

Ground bounce can occur with high-speed digital ICs when multiple outputs change states simultaneously, causing undesired transient behavior on an output or in the internal logic. This is also referred to as the Simultaneous-Switching Output (SSO) problem.

Ground bounce is primarily due to current changes in the combined inductance of ground pins, bond wires, and group metallization. The IC internal ground level deviates from the external system ground level for a short duration (a few nanoseconds) after multiple outputs change state simultaneously.

Ground bounce affects stable low outputs and all inputs because they interpret the incoming signal by comparing it to the internal ground. If the ground bounce amplitude exceeds the actual instantaneous noise margin, then a non-changing input can be interpreted as changing. Table 3-36 provides the guidelines for the maximum number of simultaneously switching outputs allowed per output power/ground pair to avoid the effects of ground bounce. (See Note 1 at the end of the table for qualifications.) Refer to Table 3-37 and Table 3-38 for the number of effective output power/ground pairs for each Virtex-II Pro / Virtex-II Pro X device and package combination.

Note on LVDCI Drivers

Table 3-36 shows LVDCI guidelines only for 50Ω impedance. For higher reference resistor (RR) values, less drive strength is needed, and the SSO limit increases linearly. To calculate the SSO limit for an LVDCI driver with different reference resistors, the following formula is used:

\[ \text{New SSO Limit} = \frac{\text{New RR Value}}{50\Omega} \times \text{SSO Limit for 50Ω} \]
Example:

If using the LVDCI_18 driver with 65Ω reference resistors, the LVDCI_18 SSO limit for 50Ω impedance is first taken from the table. The SSO limit for LVDCI_18 at 50Ω is 11 per power/ground pin pair. Therefore, the SSO limit for LVDCI_18 at 65Ω is 
\[(65Ω / 50Ω) \times 11 = 14\]
per power/ground pin pair.

Since some of the ground pins are shared inside the package, the effective power/ground pairs per bank may be fewer than the physical power/ground pair pins. The following tables show the number of equivalent power/ground pairs.

Table 3-36: Guidelines for Maximum Number of Simultaneously Switching Outputs per Power/Ground Pair

<table>
<thead>
<tr>
<th>Standard</th>
<th>Package: FG, FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVTTIL_S2</td>
<td>68</td>
</tr>
<tr>
<td>LVTTIL_S4</td>
<td>41</td>
</tr>
<tr>
<td>LVTTIL_S6</td>
<td>29</td>
</tr>
<tr>
<td>LVTTIL_S8</td>
<td>22</td>
</tr>
<tr>
<td>LVTTIL_S12</td>
<td>15</td>
</tr>
<tr>
<td>LVTTIL_S16</td>
<td>11</td>
</tr>
<tr>
<td>LVTTIL_S24</td>
<td>7</td>
</tr>
<tr>
<td>LVTTIL_F2</td>
<td>40</td>
</tr>
<tr>
<td>LVTTIL_F4</td>
<td>24</td>
</tr>
<tr>
<td>LVTTIL_F6</td>
<td>17</td>
</tr>
<tr>
<td>LVTTIL_F8</td>
<td>13</td>
</tr>
<tr>
<td>LVTTIL_F12</td>
<td>10</td>
</tr>
<tr>
<td>LVTTIL_F16</td>
<td>8</td>
</tr>
<tr>
<td>LVTTIL_F24</td>
<td>5</td>
</tr>
<tr>
<td>LVDCI_15 50Ω impedance</td>
<td>10</td>
</tr>
<tr>
<td>LVDCI_DV2_15 25Ω impedance</td>
<td>5</td>
</tr>
<tr>
<td>LVCMOS15_S2</td>
<td>51</td>
</tr>
<tr>
<td>LVCMOS15_S4</td>
<td>31</td>
</tr>
<tr>
<td>LVCMOS15_S6</td>
<td>22</td>
</tr>
<tr>
<td>LVCMOS15_S8</td>
<td>17</td>
</tr>
<tr>
<td>LVCMOS15_S12</td>
<td>11</td>
</tr>
<tr>
<td>LVCMOS15_S16</td>
<td>8</td>
</tr>
<tr>
<td>LVCMOS15_F2</td>
<td>30</td>
</tr>
<tr>
<td>LVCMOS15_F4</td>
<td>18</td>
</tr>
<tr>
<td>LVCMOS15_F6</td>
<td>13</td>
</tr>
<tr>
<td>LVCMOS15_F8</td>
<td>10</td>
</tr>
</tbody>
</table>
Table 3-36: Guidelines for Maximum Number of Simultaneously Switching Outputs per Power/Ground Pair (Continued)

<table>
<thead>
<tr>
<th>Standard</th>
<th>Package: FG, FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVCMOS15_F12</td>
<td>8</td>
</tr>
<tr>
<td>LVCMOS15_F16</td>
<td>6</td>
</tr>
<tr>
<td>LVDCI_18, 50Ω impedance</td>
<td>11</td>
</tr>
<tr>
<td>LVDCI_DV2_18, 25Ω impedance</td>
<td>5</td>
</tr>
<tr>
<td>LVCMOS18_S2</td>
<td>58</td>
</tr>
<tr>
<td>LVCMOS18_S4</td>
<td>35</td>
</tr>
<tr>
<td>LVCMOS18_S6</td>
<td>25</td>
</tr>
<tr>
<td>LVCMOS18_S8</td>
<td>19</td>
</tr>
<tr>
<td>LVCMOS18_S12</td>
<td>13</td>
</tr>
<tr>
<td>LVCMOS18_S16</td>
<td>10</td>
</tr>
<tr>
<td>LVCMOS18_F2</td>
<td>34</td>
</tr>
<tr>
<td>LVCMOS18_F4</td>
<td>20</td>
</tr>
<tr>
<td>LVCMOS18_F6</td>
<td>15</td>
</tr>
<tr>
<td>LVCMOS18_F8</td>
<td>11</td>
</tr>
<tr>
<td>LVCMOS18_F12</td>
<td>9</td>
</tr>
<tr>
<td>LVCMOS18_F16</td>
<td>7</td>
</tr>
<tr>
<td>LVDCI_25, 50 Ω impedance</td>
<td>13</td>
</tr>
<tr>
<td>LVDCI_DV2_, 25 25 Ω impedance</td>
<td>6</td>
</tr>
<tr>
<td>LVCMOS25_S2</td>
<td>68</td>
</tr>
<tr>
<td>LVCMOS25_S4</td>
<td>41</td>
</tr>
<tr>
<td>LVCMOS25_S6</td>
<td>29</td>
</tr>
<tr>
<td>LVCMOS25_S8</td>
<td>22</td>
</tr>
<tr>
<td>LVCMOS25_S12</td>
<td>15</td>
</tr>
<tr>
<td>LVCMOS25_S16</td>
<td>11</td>
</tr>
<tr>
<td>LVCMOS25_S24</td>
<td>7</td>
</tr>
<tr>
<td>LVCMOS25_F2</td>
<td>40</td>
</tr>
<tr>
<td>LVCMOS25_F4</td>
<td>24</td>
</tr>
<tr>
<td>LVCMOS25_F6</td>
<td>17</td>
</tr>
<tr>
<td>LVCMOS25_F8</td>
<td>13</td>
</tr>
<tr>
<td>LVCMOS25_F12</td>
<td>10</td>
</tr>
<tr>
<td>LVCMOS25_F16</td>
<td>8</td>
</tr>
<tr>
<td>LVCMOS25_F24</td>
<td>5</td>
</tr>
<tr>
<td>LVDCI_33, 50Ω impedance</td>
<td>13</td>
</tr>
</tbody>
</table>
### Table 3-36: Guidelines for Maximum Number of Simultaneously Switching Outputs per Power/Ground Pair \(^{(1)}\) (Continued)

<table>
<thead>
<tr>
<th>Standard</th>
<th>Package: FG, FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVDCI_DV2_33, 25Ω impedance</td>
<td>6</td>
</tr>
<tr>
<td>LVCMOS33_S2</td>
<td>68</td>
</tr>
<tr>
<td>LVCMOS33_S4</td>
<td>41</td>
</tr>
<tr>
<td>LVCMOS33_S6</td>
<td>29</td>
</tr>
<tr>
<td>LVCMOS33_S8</td>
<td>22</td>
</tr>
<tr>
<td>LVCMOS33_S12</td>
<td>15</td>
</tr>
<tr>
<td>LVCMOS33_S16</td>
<td>11</td>
</tr>
<tr>
<td>LVCMOS33_S24</td>
<td>7</td>
</tr>
<tr>
<td>LVCMOS33_F2</td>
<td>40</td>
</tr>
<tr>
<td>LVCMOS33_F4</td>
<td>24</td>
</tr>
<tr>
<td>LVCMOS33_F6</td>
<td>17</td>
</tr>
<tr>
<td>LVCMOS33_F8</td>
<td>13</td>
</tr>
<tr>
<td>LVCMOS33_F12</td>
<td>10</td>
</tr>
<tr>
<td>LVCMOS33_F16</td>
<td>8</td>
</tr>
<tr>
<td>LVCMOS33_F24</td>
<td>5</td>
</tr>
<tr>
<td>PCI_33, PCI_66, PCIX</td>
<td>8</td>
</tr>
<tr>
<td>GTL</td>
<td>4</td>
</tr>
<tr>
<td>GTL_DCI</td>
<td>4</td>
</tr>
<tr>
<td>GTLP</td>
<td>4</td>
</tr>
<tr>
<td>GTLP_DCI</td>
<td>4</td>
</tr>
<tr>
<td>HSTLI</td>
<td>20</td>
</tr>
<tr>
<td>HSTLI_DCI</td>
<td>20</td>
</tr>
<tr>
<td>HSTLIi</td>
<td>10</td>
</tr>
<tr>
<td>HSTLIi_DCI</td>
<td>6</td>
</tr>
<tr>
<td>HSTLIii</td>
<td>8</td>
</tr>
<tr>
<td>HSTLIii_DCI</td>
<td>8</td>
</tr>
<tr>
<td>HSTLIV</td>
<td>4</td>
</tr>
<tr>
<td>HSTLIV_DCI</td>
<td>4</td>
</tr>
<tr>
<td>SSTL18_I</td>
<td>20</td>
</tr>
<tr>
<td>SSTL18_I_DCI</td>
<td>20</td>
</tr>
<tr>
<td>SSTL18_II</td>
<td>13</td>
</tr>
<tr>
<td>SSTL18_II_DCI</td>
<td>6</td>
</tr>
<tr>
<td>SSTL2_I</td>
<td>15</td>
</tr>
</tbody>
</table>
### Guidelines for Maximum Number of Simultaneously Switching Outputs per Power/Ground Pair (Continued)

<table>
<thead>
<tr>
<th>Standard</th>
<th>Package: FG, FF</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSTL2_I_DCI</td>
<td>15</td>
</tr>
<tr>
<td>SSTL2_II</td>
<td>10</td>
</tr>
<tr>
<td>SSTL2_II_DCI</td>
<td>5</td>
</tr>
<tr>
<td>HSTL_I_18</td>
<td>16</td>
</tr>
<tr>
<td>HSTL_I_DCI_18</td>
<td>16</td>
</tr>
<tr>
<td>HSTL_II_18</td>
<td>8</td>
</tr>
<tr>
<td>HSTL_II_DCI_18</td>
<td>6</td>
</tr>
<tr>
<td>HSTL_III_18</td>
<td>6</td>
</tr>
<tr>
<td>HSTL_III_DCI_18</td>
<td>6</td>
</tr>
<tr>
<td>HSTL_IV_18</td>
<td>3</td>
</tr>
<tr>
<td>HSTL_IV_DCI_18</td>
<td>3</td>
</tr>
<tr>
<td>LVPECL_25</td>
<td>8</td>
</tr>
<tr>
<td>BLVDS_25</td>
<td>8</td>
</tr>
<tr>
<td>DIFF_HSTL_II</td>
<td>10</td>
</tr>
<tr>
<td>DIFF_HSTL_II_DCI</td>
<td>6</td>
</tr>
<tr>
<td>DIFF_HSTL_II_18</td>
<td>8</td>
</tr>
<tr>
<td>DIFF_HSTL_II_18_DC</td>
<td>6</td>
</tr>
<tr>
<td>DIFF_SSTL_II</td>
<td>10</td>
</tr>
<tr>
<td>DIFF_SSTL_II_DCI</td>
<td>6</td>
</tr>
<tr>
<td>DIFF_SSTL_II_18</td>
<td>12</td>
</tr>
<tr>
<td>DIFF_SSTL_II_18_DC</td>
<td>6</td>
</tr>
</tbody>
</table>

### Notes:
1. The maximum number of simultaneously switching outputs per power/ground pair may be less than the number given in this table if the external bypass capacitor solution has a high series inductance. This table presumes the use of ultra-low inductance bypass capacitors. Refer to “VCC Decoupling,” page 481.
Table 3-37 shows the number of pairs in banks 0, 1, 4, and 5 (top and bottom edges of the die), while Table 3-38 shows the numbers in banks 2, 3, 6, and 7. This is because MGTs reside on the top and bottom edge only.

**Table 3-37: Equivalent Pwr/Gnd Pairs per Bank, Top/Bottom**

<table>
<thead>
<tr>
<th>Package</th>
<th>For Banks 0, 1, 4, and 5 (Top/Bottom Sides):</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2VP2</td>
</tr>
<tr>
<td>FG256/FGG256</td>
<td>2</td>
</tr>
<tr>
<td>FG456/FGG456</td>
<td>3</td>
</tr>
<tr>
<td>FG676/FGG676</td>
<td></td>
</tr>
<tr>
<td>FF672</td>
<td>4</td>
</tr>
<tr>
<td>FF896</td>
<td></td>
</tr>
<tr>
<td>FF1152</td>
<td></td>
</tr>
<tr>
<td>FF1148</td>
<td></td>
</tr>
<tr>
<td>FF1517</td>
<td></td>
</tr>
<tr>
<td>FF1704</td>
<td></td>
</tr>
<tr>
<td>FF1696</td>
<td></td>
</tr>
</tbody>
</table>

**Table 3-38: Equivalent Pwr/Gnd Pairs per Bank, Left/Right**

<table>
<thead>
<tr>
<th>Package</th>
<th>For Banks 2, 3, 6, and 7 (Left/Right Sides):</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2VP2</td>
</tr>
<tr>
<td>FG256/FGG256</td>
<td>2</td>
</tr>
<tr>
<td>FG456/FGG456</td>
<td>2</td>
</tr>
<tr>
<td>FG676/FGG676</td>
<td></td>
</tr>
<tr>
<td>FF672</td>
<td>3</td>
</tr>
<tr>
<td>FF896</td>
<td></td>
</tr>
<tr>
<td>FF1152</td>
<td></td>
</tr>
<tr>
<td>FF1148</td>
<td></td>
</tr>
<tr>
<td>FF1517</td>
<td></td>
</tr>
<tr>
<td>FF1704</td>
<td></td>
</tr>
<tr>
<td>FF1696</td>
<td></td>
</tr>
</tbody>
</table>

Refer to [XAPP689](XAPP689) for further details on managing ground bounce in large FPGAs.
Application Examples

Creating a design with the SelectIO-Ultra feature requires either assignment of the IOSTANDARD attribute in the constraint file or instantiation of the desired library symbol within the design code.

To enter the IOSTANDARD attribute in the constraint file (UCF file), the following syntax can be used:

```
NET <pad net name> IOSTANDARD=<the name of the standard>
```

For example, to enter LVCMOS25 standard, use

```
NET <pad net name> IOSTANDARD=LVCMOS25;
```

To instantiate a library symbol in the HDL code, use the proper input or output buffer name, and follow the standard syntax of instantiation.

For example, to instantiate a GTL input buffer in VHDL, the following syntax can be used:

```
GTL_buffer : IBUF_GTL port map (I=>data_in, O=>data_gtl_in);
```

At the board level, designers need to know the termination techniques required for each I/O standard.

This section describes some common application examples illustrating the termination techniques recommended by each of the single-ended standard supported by the SelectIO-Ultra features.

Termination Examples

Circuit examples involving typical termination techniques for each of the SelectIO-Ultra standards follow. For a full range of accepted values for the DC voltage specifications for each standard, refer to the table associated with each figure.

The resistors used in each termination technique example and the transmission lines depicted represent board level components and are not meant to represent components on the device.
GTL

A sample circuit illustrating a valid termination technique for GTL is shown in Figure 3-82.

![GTL Terminated](image)

Figure 3-82: **GTL Terminated**

Table 3-39 lists DC voltage specifications.

### Table 3-39: GTL Voltage Specifications

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{CCO} )</td>
<td>-</td>
<td>N/A</td>
<td>-</td>
</tr>
<tr>
<td>( V_{REF} = N \times V_{TT}(1) )</td>
<td>0.74</td>
<td>0.8</td>
<td>0.86</td>
</tr>
<tr>
<td>( V_{TT} )</td>
<td>1.14</td>
<td>1.2</td>
<td>1.26</td>
</tr>
<tr>
<td>( V_{IH} \geq V_{REF} + 0.05 )</td>
<td>0.79</td>
<td>0.85</td>
<td>-</td>
</tr>
<tr>
<td>( V_{IL} \leq V_{REF} - 0.05 )</td>
<td>-</td>
<td>0.75</td>
<td>0.81</td>
</tr>
<tr>
<td>( V_{OH} )</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( V_{OL} )</td>
<td>-</td>
<td>0.2</td>
<td>0.4</td>
</tr>
<tr>
<td>( I_{OH} ) at ( V_{OH} ) (mA)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( I_{OL} ) at ( V_{OL} ) (mA) at 0.4V</td>
<td>32</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( I_{OL} ) at ( V_{OL} ) (mA) at 0.2V</td>
<td>-</td>
<td>-</td>
<td>40</td>
</tr>
</tbody>
</table>

**Notes:**
1. \( N \) must be greater than or equal to 0.653 and less than or equal to 0.68.
GTL +

Figure 3-83 shows a sample circuit illustrating a valid termination technique for GTLP.

![Figure 3-83: GTLP Terminated](image)

Table 3-40 lists DC voltage specifications.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{CCO}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{REF} = N \times V_{TT(1)}$</td>
<td>0.88</td>
<td>1.0</td>
<td>1.12</td>
</tr>
<tr>
<td>$V_{TT}$</td>
<td>1.35</td>
<td>1.5</td>
<td>1.65</td>
</tr>
<tr>
<td>$V_{IH} \geq V_{REF} + 0.1$</td>
<td>0.98</td>
<td>1.1</td>
<td>-</td>
</tr>
<tr>
<td>$V_{IL} \leq V_{REF} - 0.1$</td>
<td>-</td>
<td>0.9</td>
<td>1.02</td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>0.3</td>
<td>0.45</td>
<td>0.6</td>
</tr>
<tr>
<td>$I_{OH} at V_{OH}$ (mA)</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$I_{OL} at V_{OL}$ (mA) at 0.6V</td>
<td>36</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$I_{OL} at V_{OL}$ (mA) at 0.3V</td>
<td>-</td>
<td>-</td>
<td>48</td>
</tr>
</tbody>
</table>

Notes:
1. $N$ must be greater than or equal to 0.653 and less than or equal to 0.68.
HSTL Class I

Figure 3-84 shows a sample circuit illustrating a valid termination technique for HSTL_I.

![Terminated HSTL Class I Circuit](image)

**Figure 3-84: Terminated HSTL Class I**

Table 3-41 lists DC voltage specifications.

**Table 3-41: HSTL Class I Voltage Specification**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{CCO}$</td>
<td>1.40</td>
<td>1.50</td>
<td>1.60</td>
</tr>
<tr>
<td>$V_{REF}$</td>
<td>0.68</td>
<td>0.75</td>
<td>0.90</td>
</tr>
<tr>
<td>$V_{TT}$</td>
<td>-</td>
<td>$V_{CCO} \times 0.5$</td>
<td>-</td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>$V_{REF} + 0.1$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>-</td>
<td>-</td>
<td>$V_{REF} - 0.1$</td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>$V_{CCO} - 0.4$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
</tr>
<tr>
<td>$I_{OH}$ at $V_{OH}$ (mA)</td>
<td>-8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$I_{OL}$ at $V_{OL}$ (mA)</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
HSTL Class II

Figure 3-85 shows a sample circuit illustrating a valid termination technique for HSTL-II.

![Terminated HSTL Class II](image)

Figure 3-85: Terminated HSTL Class II

Table 3-42 lists DC voltage specifications.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{CCO} )</td>
<td>1.40</td>
<td>1.50</td>
<td>1.60</td>
</tr>
<tr>
<td>( V_{REF} ) (i)</td>
<td>-</td>
<td>0.75</td>
<td>-</td>
</tr>
<tr>
<td>( V_{TT} )</td>
<td>-</td>
<td>( V_{CCO} \times 0.5 )</td>
<td>-</td>
</tr>
<tr>
<td>( V_{IH} )</td>
<td>( V_{REF} + 0.1 )</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( V_{IL} )</td>
<td>-</td>
<td>-</td>
<td>( V_{REF} - 0.1 )</td>
</tr>
<tr>
<td>( V_{OH} )</td>
<td>( V_{CCO} - 0.4 )</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( V_{OL} )</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
</tr>
<tr>
<td>( I_{OH \text{ at } V_{OH}} \text{ (mA)} )</td>
<td>-16</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( I_{OL \text{ at } V_{OL}} \text{ (mA)} )</td>
<td>16</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Notes:

1. Per EIA/JESD8-6, “The value of \( V_{REF} \) is to be selected by the user to provide optimum noise margin in the use conditions specified by the user.”
HSTL Class III

Figure 3-86 shows a sample circuit illustrating a valid termination technique for HSTL_III.

![Terminated HSTL Class III Circuit](UG012_C2_011_121101)

**Figure 3-86:** Terminated HSTL Class III

Table 3-43 lists DC voltage specifications.

**Table 3-43:** HSTL Class III Voltage Specification

<table>
<thead>
<tr>
<th>Parameter</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{CCO}$</td>
<td>1.40</td>
<td>1.50</td>
<td>1.60</td>
</tr>
<tr>
<td>$V_{REF}^{(1)}$</td>
<td>-</td>
<td>0.90</td>
<td>-</td>
</tr>
<tr>
<td>$V_{TT}$</td>
<td>-</td>
<td>$V_{CCO}$</td>
<td>-</td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>$V_{REF} + 0.1$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>-</td>
<td>-</td>
<td>$V_{REF} - 0.1$</td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>$V_{CCO} - 0.4$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
</tr>
<tr>
<td>$I_{OH}$ at $V_{OH}$ (mA)</td>
<td>-8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$I_{OL}$ at $V_{OL}$ (mA)</td>
<td>24</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Notes:**

1. Per EIA/JESD8-6, “The value of $V_{REF}$ is to be selected by the user to provide optimum noise margin in the use conditions specified by the user.”
HSTL Class IV

Figure 3-87 shows a sample circuit illustrating a valid termination technique for HSTL_IV.

![Terminated HSTL Class IV](image)

Figure 3-87: Terminated HSTL Class IV

Table 3-44 lists DC voltage specifications.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{CCO}$</td>
<td>1.40</td>
<td>1.50</td>
<td>1.60</td>
</tr>
<tr>
<td>$V_{REF}$</td>
<td>-</td>
<td>0.90</td>
<td>-</td>
</tr>
<tr>
<td>$V_{TT}$</td>
<td>-</td>
<td>$V_{CCO}$</td>
<td>-</td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>$V_{REF} + 0.1$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>-</td>
<td>-</td>
<td>$V_{REF} - 0.1$</td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>$V_{CCO} - 0.4$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
</tr>
<tr>
<td>$I_{OH}$ at $V_{OH}$ (mA)</td>
<td>-8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$I_{OL}$ at $V_{OL}$ (mA)</td>
<td>48</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Notes:
1. Per EIA/JESD8-6, “The value of $V_{REF}$ is to be selected by the user to provide optimum noise margin in the use conditions specified by the user.”
HSTL Class I (1.8V)

Figure 3-88 shows a sample circuit illustrating a valid termination technique for HSTL_I.

![Terminated HSTL Class I (1.8V)](image)

**Figure 3-88:** Terminated HSTL Class I (1.8V)

Table 3-45 lists DC voltage specifications.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>V&lt;br&gt;CCO</td>
<td>1.7</td>
<td>1.8</td>
<td>1.9</td>
</tr>
<tr>
<td>V&lt;br&gt;REF</td>
<td>0.8</td>
<td>0.9</td>
<td>1.1</td>
</tr>
<tr>
<td>V&lt;br&gt;TT</td>
<td>-</td>
<td>V&lt;br&gt;CCO x 0.5</td>
<td>-</td>
</tr>
<tr>
<td>V&lt;br&gt;IH</td>
<td>V&lt;br&gt;REF + 0.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V&lt;br&gt;IL</td>
<td>-</td>
<td>-</td>
<td>V&lt;br&gt;REF - 0.1</td>
</tr>
<tr>
<td>V&lt;br&gt;OH</td>
<td>V&lt;br&gt;CCO - 0.4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V&lt;br&gt;OL</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
</tr>
<tr>
<td>I&lt;br&gt;OH at V&lt;br&gt;OH (mA)</td>
<td>-8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I&lt;br&gt;OL at V&lt;br&gt;OL (mA)</td>
<td>8</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
HSTL Class II (1.8V)

Figure 3-89 shows a sample circuit illustrating a valid termination technique for HSTL_IID.

![Figure 3-89: Terminated HSTL Class II (1.8V)](image)

Table 3-46 lists DC voltage specifications.

**Table 3-46: HSTL Class II (1.8V) Voltage Specification**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{CCO} )</td>
<td>1.7</td>
<td>1.8</td>
<td>1.9</td>
</tr>
<tr>
<td>( V_{REF} )</td>
<td>-</td>
<td>0.9</td>
<td>-</td>
</tr>
<tr>
<td>( V_{TT} )</td>
<td>-</td>
<td>( V_{CCO} \times 0.5 )</td>
<td>-</td>
</tr>
<tr>
<td>( V_{IH} )</td>
<td>( V_{REF} + 0.1 )</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( V_{IL} )</td>
<td>-</td>
<td>-</td>
<td>( V_{REF} - 0.1 )</td>
</tr>
<tr>
<td>( V_{OH} )</td>
<td>( V_{CCO} - 0.4 )</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( V_{OL} )</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
</tr>
<tr>
<td>( I_{OH} )</td>
<td>-16</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( I_{OL} )</td>
<td>16</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Notes:**

1. Per EIA/JESD8-6, “The value of \( V_{REF} \) is to be selected by the user to provide optimum noise margin in the use conditions specified by the user.”
HSTL Class III (1.8V)

Figure 3-90 shows a sample circuit illustrating a valid termination technique for HSTL_III.

![Terminated HSTL Class III (1.8V)](image)

Table 3-47 lists DC voltage specifications.

**Table 3-47: HSTL Class III (1.8V) Voltage Specification**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{CCO}$</td>
<td>1.7</td>
<td>1.8</td>
<td>1.9</td>
</tr>
<tr>
<td>$V_{REF}$(1)</td>
<td>-</td>
<td>1.1</td>
<td>-</td>
</tr>
<tr>
<td>$V_{TT}$</td>
<td>-</td>
<td>$V_{CCO}$</td>
<td>-</td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>$V_{REF} + 0.1$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>-</td>
<td>-</td>
<td>$V_{REF} - 0.1$</td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>$V_{CCO} - 0.4$</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
</tr>
<tr>
<td>$I_{OH}$ at $V_{OH}$ (mA)</td>
<td>-8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$I_{OL}$ at $V_{OL}$ (mA)</td>
<td>24</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Notes:**

1. Per EIA/JESD8-6, “The value of $V_{REF}$ is to be selected by the user to provide optimum noise margin in the use conditions specified by the user.”
HSTL Class IV (1.8V)

Figure 3-91 shows a sample circuit illustrating a valid termination technique for HSTL_IV.

![Figure 3-91: Terminated HSTL Class IV (1.8V)](image)

Table 3-48 lists DC voltage specifications.

**Table 3-48: HSTL Class IV (1.8V) Voltage Specification**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>MIN</th>
<th>TYP</th>
<th>MAX</th>
</tr>
</thead>
<tbody>
<tr>
<td>VCCO</td>
<td>1.7</td>
<td>1.8</td>
<td>1.9</td>
</tr>
<tr>
<td>VREF</td>
<td>-</td>
<td>1.1</td>
<td>-</td>
</tr>
<tr>
<td>VTT</td>
<td>-</td>
<td>VCCO</td>
<td>-</td>
</tr>
<tr>
<td>VIH</td>
<td>VREF + 0.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VIL</td>
<td>-</td>
<td>-</td>
<td>VREF - 0.1</td>
</tr>
<tr>
<td>VOH</td>
<td>VCCO - 0.4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>VOL</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
</tr>
<tr>
<td>IOH at VOH (mA)</td>
<td>-8</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>IOL at VOL (mA)</td>
<td>48</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Notes:**
1. Per EIA/JESD8-6, “The value of VREF is to be selected by the user to provide optimum noise margin in the use conditions specified by the user.”
SSTL2_I

Figure 3-92 shows a sample circuit illustrating a valid termination technique for SSTL2_I.

![Terminated SSTL2_I](image)

Figure 3-92: Terminated SSTL2_I

Table 3-49 lists DC voltage specifications.

### Table 3-49: SSTL2_I Voltage Specifications

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>V\textsubscript{CCO}</td>
<td>2.3</td>
<td>2.5</td>
<td>2.7</td>
</tr>
<tr>
<td>V\textsubscript{REF} = 0.5 × V\textsubscript{CCO}</td>
<td>1.13</td>
<td>1.25</td>
<td>1.38</td>
</tr>
<tr>
<td>V\textsubscript{TT} = V\textsubscript{REF} + N\textsuperscript{(1)}</td>
<td>1.11</td>
<td>1.25</td>
<td>1.39</td>
</tr>
<tr>
<td>V\textsubscript{IH} ≥ V\textsubscript{REF} + 0.18</td>
<td>1.3</td>
<td>1.43</td>
<td>3.0\textsuperscript{(2)}</td>
</tr>
<tr>
<td>V\textsubscript{IL} = V\textsubscript{REF} − 0.18</td>
<td>−0.3\textsuperscript{(3)}</td>
<td>1.07</td>
<td>1.2</td>
</tr>
<tr>
<td>V\textsubscript{OH} ≥ V\textsubscript{REF} + 0.61\textsuperscript{(4)}</td>
<td>1.74</td>
<td>1.84</td>
<td>1.99</td>
</tr>
<tr>
<td>V\textsubscript{OL} = V\textsubscript{REF} − 0.61\textsuperscript{(4)}</td>
<td>0.52</td>
<td>0.66</td>
<td>0.77</td>
</tr>
<tr>
<td>I\textsubscript{OH} at V\textsubscript{OH} (mA)</td>
<td>−8.1</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I\textsubscript{OL} at V\textsubscript{OL} (mA)</td>
<td>8.1</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

### Notes:
1. N must be greater than or equal to -0.04 and less than or equal to 0.04.
2. V\textsubscript{IH} maximum is V\textsubscript{CCO} + 0.3.
3. V\textsubscript{IL} minimum does not conform to the formula.
4. V\textsubscript{OH} and V\textsubscript{OL} will be different for SSTL2_I\_DCI, because it uses a controlled-impedance driver.
SSTL2\_II

Figure 3-93 shows a sample circuit illustrating a valid termination technique for SSTL2\_II.

![Terminated SSTL2 II](image)

**Figure 3-93:** Terminated SSTL2 II

Table 3-50 lists DC voltage specifications.

**Table 3-50: SSTL2\_II Voltage Specifications**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>( V_{\text{CCO}} )</td>
<td>2.3</td>
<td>2.5</td>
<td>2.7</td>
</tr>
<tr>
<td>( V_{\text{REF}} = 0.5 \times V_{\text{CCO}} )</td>
<td>1.13</td>
<td>1.25</td>
<td>1.38</td>
</tr>
<tr>
<td>( V_{\text{TT}} = V_{\text{REF}} + N^{(1)} )</td>
<td>1.11</td>
<td>1.25</td>
<td>1.42</td>
</tr>
<tr>
<td>( V_{\text{IH}} \geq V_{\text{REF}} + 0.15 )</td>
<td>1.30</td>
<td>1.43</td>
<td>3.0(^{(2)})</td>
</tr>
<tr>
<td>( V_{\text{IL}} \leq V_{\text{REF}} - 0.15 )</td>
<td>-0.3(^{(3)})</td>
<td>1.07</td>
<td>1.2</td>
</tr>
<tr>
<td>( V_{\text{OH}} \geq V_{\text{REF}} + 0.8^{(4)} )</td>
<td>1.93</td>
<td>2.06</td>
<td>-</td>
</tr>
<tr>
<td>( V_{\text{OL}} \leq V_{\text{REF}} - 0.8^{(4)} )</td>
<td>-</td>
<td>0.44</td>
<td>0.55</td>
</tr>
<tr>
<td>( I_{\text{OH}} ) at ( V_{\text{OH}} ) (mA)</td>
<td>-16.2</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>( I_{\text{OL}} ) at ( V_{\text{OL}} ) (mA)</td>
<td>16.2</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

**Notes:**

1. \( N \) must be greater than or equal to \(-0.04\) and less than or equal to 0.04.
2. \( V_{\text{IH}} \) maximum is \( V_{\text{CCO}} + 0.3 \).
3. \( V_{\text{IL}} \) minimum does not conform to the formula.
4. \( V_{\text{OH}} \) and \( V_{\text{OL}} \) will be different for SSTL2\_II\_DCI, because it uses a controlled-impedance driver.
SSTL18_II

Figure 3-94 shows a sample circuit illustrating a valid termination technique for SSTL18_II.

Figure 3-94: Terminated SSTL18_II

Table 3-51 lists DC voltage specifications.

Table 3-51: SSTL18_II Voltage Specifications (JESD8-15)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{CCO}$</td>
<td>1.7</td>
<td>1.8</td>
<td>1.9</td>
</tr>
<tr>
<td>$V_{REF} = 0.5 \times V_{CCO}$</td>
<td>0.833</td>
<td>0.9</td>
<td>0.969</td>
</tr>
<tr>
<td>$V_{TT} = V_{REF} + N^{(1)}$</td>
<td>0.793</td>
<td>0.9</td>
<td>1.009</td>
</tr>
<tr>
<td>$V_{IH} \geq V_{REF} + 0.15$</td>
<td>0.958</td>
<td></td>
<td>2.2</td>
</tr>
<tr>
<td>$V_{IL} \leq V_{REF} - 0.15$</td>
<td>-0.3</td>
<td></td>
<td>0.844</td>
</tr>
<tr>
<td>$V_{OH} \geq V_{REF} + 0.8^{(4)}$</td>
<td>1.396</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$V_{OL} \leq V_{REF} - 0.8^{(4)}$</td>
<td></td>
<td></td>
<td>0.406</td>
</tr>
<tr>
<td>$I_{OH}$ at $V_{OH}$ (mA)</td>
<td>-13.4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$I_{OL}$ at $V_{OL}$ (mA)</td>
<td>13.4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
PCI33_3, PCI66_3, and PCI-X

Table 3-52 lists DC voltage specifications.

Table 3-52: PCI33_3, PCI66_3, and PCI-X Voltage Specifications

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_{CCO}</td>
<td>3.0</td>
<td>3.3</td>
<td>3.5</td>
</tr>
<tr>
<td>V_{REF}</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V_{TT}</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V_{IH} = 0.5 \times V_{CCO}</td>
<td>1.5</td>
<td>1.65</td>
<td>V_{CCO} + 0.5</td>
</tr>
<tr>
<td>V_{IL} = 0.3 \times V_{CCO}</td>
<td>-0.5</td>
<td>0.99</td>
<td>1.08</td>
</tr>
<tr>
<td>V_{OH} = 0.9 \times V_{CCO}</td>
<td>2.7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V_{OL} = 0.1 \times V_{CCO}</td>
<td>-</td>
<td>-</td>
<td>0.36</td>
</tr>
<tr>
<td>I_{OH} at V_{OH} (mA)</td>
<td>(Note 1)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I_{OL} at V_{OL} (mA)</td>
<td>(Note 1)</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Notes:
1. Tested according to the relevant specification.

LVTTL

Table 3-53 lists DC voltage specifications.

Table 3-53: LVTTL Voltage Specifications

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>V_{CCO}</td>
<td>3.0</td>
<td>3.3</td>
<td>3.6</td>
</tr>
<tr>
<td>V_{REF}</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V_{TT}</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V_{IH}</td>
<td>2.0</td>
<td>-</td>
<td>3.6</td>
</tr>
<tr>
<td>V_{IL}</td>
<td>-0.5</td>
<td>-</td>
<td>0.8</td>
</tr>
<tr>
<td>V_{OH}</td>
<td>2.4</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>V_{OL}</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
</tr>
<tr>
<td>I_{OH} at V_{OH} (mA)</td>
<td>-24</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>I_{OL} at V_{OL} (mA)</td>
<td>24</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Notes:
1. V_{OL} and V_{OH} for lower drive currents are sample tested.
LVCMOS15

Table 3-54 lists DC voltage specifications.

Table 3-54: LVCMOS15 Voltage Specifications

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{CCO}$</td>
<td>-</td>
<td>1.5</td>
<td>-</td>
</tr>
<tr>
<td>$V_{REF}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{TT}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{IH} = 0.7 \times V_{CCO}$</td>
<td>1.05</td>
<td>-</td>
<td>1.65</td>
</tr>
<tr>
<td>$V_{IL} = 0.2 \times V_{CCO}$</td>
<td>-0.5</td>
<td>-</td>
<td>0.3</td>
</tr>
<tr>
<td>$V_{OH} = V_{CCO} - 0.45$</td>
<td>-</td>
<td>1.05</td>
<td>-</td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
</tr>
<tr>
<td>$I_{OH}$ at $V_{OH}$ (mA)</td>
<td>-16</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$I_{OL}$ at $V_{OL}$ (mA)</td>
<td>16</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

LVCMOS18

Table 3-55 lists DC voltage specifications.

Table 3-55: LVCMOS18 Voltage Specifications

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{CCO}$</td>
<td>1.7</td>
<td>1.8</td>
<td>1.9</td>
</tr>
<tr>
<td>$V_{REF}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{TT}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{IH} = 0.7 \times V_{CCO}$</td>
<td>1.19</td>
<td>-</td>
<td>1.95</td>
</tr>
<tr>
<td>$V_{IL} = 0.2 \times V_{CCO}$</td>
<td>-0.5</td>
<td>-</td>
<td>0.4</td>
</tr>
<tr>
<td>$V_{OH} = V_{CCO} - 0.4$</td>
<td>1.3</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
</tr>
<tr>
<td>$I_{OH}$ at $V_{OH}$ (mA)</td>
<td>-16</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$I_{OL}$ at $V_{OL}$ (mA)</td>
<td>16</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
LVCMOS25

Table 3-56 lists DC voltage specifications.

Table 3-56: LVCMOS25 Voltage Specifications

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{CCO}$</td>
<td>2.3</td>
<td>2.5</td>
<td>2.7</td>
</tr>
<tr>
<td>$V_{REF}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{TT}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>1.7</td>
<td>-</td>
<td>2.7</td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>-0.5</td>
<td>-</td>
<td>0.7</td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>1.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
</tr>
<tr>
<td>$I_{OH}$ at $V_{OH}$ (mA)</td>
<td>~24</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$I_{OL}$ at $V_{OL}$ (mA)</td>
<td>24</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

LVCMOS33

Table 3-57 lists DC voltage specifications.

Table 3-57: LVCMOS33 Voltage Specifications

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{CCO}$</td>
<td>3.0</td>
<td>3.3</td>
<td>3.6</td>
</tr>
<tr>
<td>$V_{REF}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{TT}$</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{IH}$</td>
<td>2.0</td>
<td>-</td>
<td>3.6</td>
</tr>
<tr>
<td>$V_{IL}$</td>
<td>-0.5</td>
<td>-</td>
<td>0.8</td>
</tr>
<tr>
<td>$V_{OH}$</td>
<td>2.6</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$V_{OL}$</td>
<td>-</td>
<td>-</td>
<td>0.4</td>
</tr>
<tr>
<td>$I_{OH}$ at $V_{OH}$ (mA)</td>
<td>~24</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$I_{OL}$ at $V_{OL}$ (mA)</td>
<td>24</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>
Digitally Controlled Impedance (DCI)

Introduction

As FPGAs get bigger and system clock speeds get faster, PC board design and manufacturing becomes more difficult. With ever faster edge rates, maintaining signal integrity becomes a critical issue. Designers must make sure that most PC board traces are terminated properly to avoid reflections or ringing.

To terminate a trace, resistors are traditionally added to make the output and/or input match the impedance of the receiver or driver to the impedance of the trace. However, due to the increase in device I/O counts, adding resistors close to the device pins increases the board area and component count, and in some cases might even be physically impossible. To address these issues and to achieve better signal integrity, Xilinx developed a new I/O technology for the Virtex-II and Virtex-II Pro / Virtex-II Pro X device families, Digitally Controlled Impedance (DCI).

DCI adjusts the output impedance or input termination to accurately match the characteristic impedance of the transmission line. DCI actively adjusts the impedance of the I/O to equal an external reference resistance. This compensates for changes in I/O impedance due to process variation. It also continuously adjusts the impedance of the I/O to compensate for variations of temperature and supply voltage fluctuations.

In the case of controlled impedance drivers, DCI controls the driver impedance to match two reference resistors, or optionally, to match half the value of these reference resistors. DCI eliminates the need for external series termination resistors.

DCI provides the parallel or series termination for transmitters or receivers. This eliminates the need for termination resistors on the board, reduces board routing difficulties and component count, and improves signal integrity by eliminating stub reflection. Stub reflection occurs when termination resistors are located too far from the end of the transmission line. With DCI, the termination resistors are as close as possible to the output driver or the input buffer, thus, eliminating stub reflections.

Xilinx DCI

DCI uses two multi-purpose reference pins in each bank to control the impedance of the driver or the parallel termination value for all of the I/Os of that bank. The N reference pin (VRN) must be pulled up to VCCO by a reference resistor, and the P reference pin (VRP) must be pulled down to ground by another reference resistor. The value of each reference resistor should be equal to the characteristic impedance of the PC board traces, or should be twice that value (configuration option).

When a DCI I/O standard is used on a particular bank, the two multi-purpose reference pins cannot be used as regular I/Os. However, if DCI I/O standards are not used in the bank, these pins are available as regular I/O pins. Check the Virtex-II Pro / Virtex-II Pro X pinout for detailed pin descriptions.

DCI adjusts the impedance of the I/O by selectively turning transistors in the I/Os on or off. The impedance is adjusted to match the external reference resistors. The impedance adjustment process has two phases. The first phase, which compensates for process variations, is done during the device startup sequence. The second phase, which maintains the impedance in response to temperature and supply voltage changes, begins immediately after the first phase and continues indefinitely, even while the part is operating. By default, the DONE pin does not go High until the first phase of the impedance adjustment process has completed.
If users do not wish to have the second phase enabled, they should use the FreezeDCI option in BitGen. If FreezeDCI is used, Phase One is the only impedance adjustment.

For controlled impedance output drivers, the impedance can be adjusted either to match the reference resistors or half the resistance of the reference resistors. For on-chip termination, the termination is always adjusted to match the reference resistors.

DCI can configure output drivers to be the following types:
1. Controlled Impedance Driver (Source Termination)
2. Controlled Impedance Driver with Half Impedance (Source Termination)

It can also configure inputs to have the following types of on-chip terminations:
1. Input termination to $V_{CCO}$ (Single Termination)
2. Input termination to $V_{CCO}/2$ (Split Termination, Thevenin equivalent)

For bidirectional operation, both ends of the line can be DCI-terminated permanently:
1. Driver with termination to $V_{CCO}$ (Single Termination)
2. Driver with termination to $V_{CCO}/2$ (Split Termination, Thevenin equivalent)

Alternatively, bidirectional point-to-point lines can use controlled-impedance drivers (with 3-state buffers) on both ends.
Controlled Impedance Driver (Source Termination)

Some I/O standards, such as LVCMOS, must have a drive impedance that matches the characteristic impedance of the driven line. DCI can provide controlled impedance output drivers that eliminate reflections without an external source termination. The impedance is set by external reference resistors, whose resistance should be equal to the trace impedance.

The DCI I/O standards that support Controlled Impedance Driver are: LVDCI_15, LVDCI_18, LVDCI_25, and LVDCI_33. Figure 3-95 illustrates a controlled impedance driver inside Virtex-II Pro / Virtex-II Pro X device.

![Controlled Impedance Driver](UG012_C2_047_121101)

Figure 3-95: Controlled Impedance Driver

Controlled Impedance Driver with Half Impedance (Source Termination)

DCI can also provide drivers with one half of the impedance of the reference resistors. The DCI I/O standards that support controlled impedance driver with half-impedance are LVDCI_DV2_15, LVDCI_DV2_18, and LVDCI_DV2_25.

Figure 3-96 illustrates a controlled driver with half impedance inside a Virtex-II Pro / Virtex-II Pro X device. Note that reference resistors \( R \) must be \( 2 \cdot Z_0 \) in order to match an impedance of \( Z_0 \).

![Controlled Impedance Driver with Half Impedance](UG012_C2_052_121101)

Figure 3-96: Controlled Impedance Driver with Half Impedance
Input Termination to \( V_{CCO} \) (Single Termination)

Some I/O standards, such as HSTL Class III, IV, etc., require an input termination to \( V_{CCO} \). See Figure 3-97.

![Figure 3-97: Input Termination to \( V_{CCO} \) without DCI](image1)

DCI can provide this termination to \( V_{CCO} \) using single termination. The termination resistance is set by the reference resistors. For GTL and HSTL standards, they should be controlled by 50-ohm reference resistors. The DCI I/O standards that support single termination are: GTL_DCI, GTLP_DCI, HSTL_III_DCI, HSTL_III_DCI_18, HSTL_IV_DCI, and HSTL_IV_DCI_18.

Figure 3-98 illustrates single termination inside a Virtex-II Pro / Virtex-II Pro X device.

![Figure 3-98: Input Termination Using DCI Single Termination](image2)
Input Termination to $V_{CCO}/2$ (Split Termination)

Some I/O standards, such as HSTL Class I and II, require an input termination voltage of $V_{CCO}/2$. See Figure 3-99.

This is equivalent to having a split termination composed of two resistors. One terminates to $V_{CCO}$, the other to ground. The resistor values are $2R$. DCI provides termination to $V_{CCO}/2$ using split termination. The termination resistance is set by the external reference resistors, i.e., the resistors to $V_{CC}$ and ground are each twice the reference resistor value. If users are planning to use HSTL or SSTL standards, the reference resistors should be 50-ohms. The DCI I/O standards that support split termination are: HSTL_I_DCI, HSTL_I_DCI_18, HSTL_II_DCI, HSTL_II_DCI_18, SSTL2_I_DCI, and SSTL2_II_DCI.

Figure 3-100 illustrates split termination inside a Virtex-II Pro / Virtex-II Pro X device.

Figure 3-99: Input Termination to $V_{CCO}/2$ without DCI

Figure 3-100: Input Termination to $V_{CCO}/2$ Using DCI Split Termination
Chapter 3: Design Considerations

Driver with Termination to $V_{CCO}$ (Single Termination)

Some I/O standards, such as HSTL Class IV, require an output termination to $V_{CCO}$. Figure 3-101 illustrates the output termination to $V_{CCO}$.

![Figure 3-101: Driver with Termination to $V_{CCO}$ without DCI](image)

DCI can provide this termination to $V_{CCO}$ using single termination. In this case, DCI only controls the impedance of the termination, but not the driver. If users are planning to use GTL or HSTL standards, the external reference resistors should be 50-ohms. The DCI I/O standards that support a driver with single termination are: GTL_DCI, GTLP_DCI, HSTL_IV_DCI, and HSTL_IV_DCI_18.

Figure 3-102 illustrates a driver with single termination inside a Virtex-II Pro / Virtex-II Pro X device.

![Figure 3-102: Driver with Termination to $V_{CCO}$ Using DCI Single Termination](image)
Driver with Termination to $V_{CCO}/2$ (Split Termination)

Some I/O standards, such as HSTL Class II, require an output termination to $V_{CCO}/2$. See Figure 3-103.

![Diagram of Driver with Termination to $V_{CCO}/2$ without DCI]

DCI can provide this termination to $V_{CCO}/2$ using split termination. It only controls the impedance of the termination, but not the driver. For HSTL or SSTL standards, the external reference resistors should be 50-ohms. The DCI I/O standards that support a Driver with split termination are: HSTL_II_DCI, HSTL_II_DCI_18, and SSTL2_II_DCI.

Figure 3-104 illustrates a driver with split termination inside a Virtex-II Pro / Virtex-II Pro X device.

![Diagram of Driver with Termination to $V_{CCO}/2$ Using DCI Split Termination]
Software Support

This section lists the valid DCI I/O buffer library components and describes how to use DCI in the Xilinx software.

DCI I/O Buffer Library Components

The DCI input buffer library components, including global clock buffer, are the following:

- IBUFGDS_LVDSEXT_25_DCI
- IBUFGDS_LVDS_25_DCI
- IBUFDS_LVDSEXT_25_DCI
- IBUFDS_LVDS_25_DCI
- IBUF_LVDS_25_DCI
- IBUF_LVDSEXT_25_DCI
- IBUF_LVDS_25_DCI
- IBUF_LVDSEXT_25_DCI
- IBUF_GTLP_DCI
- IBUF_GTL_DCI
- IBUF_HSTL_I_DCI
- IBUF_HSTL_II_DCI
- IBUF_HSTL_III_DCI
- IBUF_HSTL_IV_DCI
- IBUF_HSTL_I_DCI_18
- IBUF_HSTL_II_DCI_18
- IBUF_HSTL_III_DCI_18
- IBUF_HSTL_IV_DCI_18
- IBUF_LVDCI_15
- IBUF_LVDCI_18
- IBUF_LVDCI_25
- IBUF_LVDCI_33
- IBUF_LVDCI_DV2_15
- IBUF_LVDCI_DV2_18
- IBUF_LVDCI_DV2_25
- IBUF_SSTL2_I_DCI
- IBUF_SSTL2_II_DCI
- IBUF_GTLP_DCI
- IBUF_GTL_DCI
- IBUF_HSTL_I_DCI
- IBUF_HSTL_II_DCI
- IBUF_HSTL_III_DCI
- IBUF_HSTL_IV_DCI
- IBUF_LVDCI_15
- IBUF_LVDCI_18
- IBUF_LVDCI_25
- IBUF_LVDCI_33
- IBUF_LVDCI_DV2_15
Digitally Controlled Impedance (DCI)

- IBUF_LVDCI_DV2_18
- IBUF_LVDCI_DV2_25
- IBUF_SSTL2_I_DCI
- IBUF_SSTL2_II_DCI

The following are DCI output buffer library components:

- OBUF_GTLP_DCI
- OBUF_GTL_DCI
- OBUF_HSTL_I_DCI
- OBUF_HSTL_II_DCI
- OBUF_HSTL_III_DCI
- OBUF_HSTL_IV_DCI
- OBUF_HSTL_I_DCI_18
- OBUF_HSTL_II_DCI_18
- OBUF_HSTL_III_DCI_18
- OBUF_HSTL_IV_DCI_18
- OBUF_LVDCI_15
- OBUF_LVDCI_18
- OBUF_LVDCI_25
- OBUF_LVDCI_33
- OBUF_LVDCI_DV2_15
- OBUF_LVDCI_DV2_18
- OBUF_LVDCI_DV2_25
- OBUF_SSTL2_I_DCI
- OBUF_SSTL2_II_DCI

The following are DCI 3-state output buffer library components:

- OBUFT_GTLP_DCI
- OBUFT_GTL_DCI
- OBUFT_HSTL_I_DCI
- OBUFT_HSTL_II_DCI
- OBUFT_HSTL_III_DCI
- OBUFT_HSTL_IV_DCI
- OBUFT_HSTL_I_DCI_18
- OBUFT_HSTL_II_DCI_18
- OBUFT_HSTL_III_DCI_18
- OBUFT_HSTL_IV_DCI_18
- OBUFT_LVDCI_15
- OBUFT_LVDCI_18
- OBUFT_LVDCI_25
- OBUFT_LVDCI_33
- OBUFT_LVDCI_DV2_15
- OBUFT_LVDCI_DV2_18
• OBUFT_LVDCI_DV2_25
• OBUFT_SSTL2_I_DCI
• OBUFT_SSTL2_II_DCI

The following are DCI I/O buffer library components:
• IOBUF_GTLP_DCI
• IOBUF_GTL_DCI
• IOBUF_HSTL_II_DCI
• IOBUF_HSTL_IV_DCI
• IOBUF_SSTL2_II_DCI
• IOBUF_HSTL_II_DCI_18
• IOBUF_HSTL_IV_DCI_18
• IOBUF_LVDCI_15
• IOBUF_LVDCI_18
• IOBUF_LVDCI_25
• IOBUF_LVDCI_33
• IOBUF_LVDCI_DV2_15
• IOBUF_LVDCI_DV2_18
• IOBUF_LVDCI_DV2_25

How to Use DCI in the Software

There are two ways for users to use DCI for Virtex-II Pro/Virtex-II Pro X devices:
1. Use the IOSTANDARD attribute in the constraint file.
2. Instantiate DCI input or output buffers in the HDL code.

IOSTANDARD Attribute

The IOSTANDARD attribute can be entered through the NCF or UCF file. The syntax is as follows:

```
NET <net name> IOSTANDARD = LVDCI_25;
```

Where `<net name>` is the name between the IPAD and IBUF or OPAD or OBUF. For HDL designs, this name is the same as the port name.

The following are valid DCI attributes for output drivers:

• LVDCI_15
• LVDCI_18
• LVDCI_25
• LVDCI_33
• LVDCI_DV2_15
• LVDCI_DV2_18
• LVDCI_DV2_25
The following are valid DCI attributes for terminations:

- GTL_DCI
- GTLP_DCI
- HSTL_I_DCI
- HSTL_IDCI
- HSTL_II_DCI
- HSTL_II_DCI
- HSTL_III_DCI
- HSTL_III_DCI
- HSTL_IV_DCI
- HSTL_IV_DCI
- HSTL_I_DCI_18
- HSTL_II_DCI_18
- HSTL_III_DCI_18
- HSTL_IV_DCI_18
- SSTL2_I_DCI
- SSTL2_IDCI

**VHDL Example**

Instantiating DCI input and output buffers is the same as instantiating any other I/O buffers. Users must make sure that the correct I/O buffer names are used and follow the standard syntax of instantiation.

For example, to instantiate a HSTL Class I output DCI buffer, the following syntax can be used:

```
HSTL_DCI_buffer: OBUF_HSTL_I_DCI port map (I=>data_out, O=>data_out_DCI);
```

Below is an example VHDL code that instantiates four 2.5V LVDCI drivers and four HSTL Class I outputs.

```
-- Module: DCI_TEST

-- Description: VHDL example for DCI SelectIO-Ultra
-- Device: Virtex-II Pro Family
---------------------------------------------------------------------
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity dci_test is
port (clk, reset, ce, control : in std_logic;
A, B : in std_logic_vector (3 downto 0);
Dout : out std_logic_vector (3 downto 0);
muxout : out std_logic_vector (3 downto 0));
end dci_test;

architecture dci_arch of dci_test is

--DCI output buffer component declaration
component OBUF_LVDCI_25 port (I : in std_logic; O : out std_logic);
end component;
attribute syn_black_box : boolean;
attribute black_box_pad_pin : string;
attribute syn_black_box of OBUF_LVDCI_25 : component is true;
attribute black_box_pad_pin of OBUF_LVDCI_25 : component is "O";

--HSTL Class I DCI output buffer component declaration
```
component OBUF_HSTL_I_DCI port (I : in std_logic; O: out std_logic);
end component;
attribute syn_black_box of OBUF_HSTL_I_DCI : component is true;
attribute black_box_pad_pin of OBUF_HSTL_I_DCI : component is "0";

signal muxout_int : std_logic_vector (3 downto 0);
signal dout_int : std_logic_vector (3 downto 0);

begin

process (clk, reset)
begin
  if (reset = '1') then
    dout_int<="0000";
  elsif (clk'event and clk='1') then
    dout_int<=dout_int+1;
  end if;
end process;

process (controls, A, B, DOUT_INT)
begin
  if (control='1') then
    muxout_int<=A and B;
  else
    muxout_int<=Dout_int;
  end if;
end process;

U0 : OBUF_LVDCI_25 port map(
  I=>dout_int(0),
  O=>dout(0));

U1 : OBUF_LVDCI_25 port map(
  I=>dout_int(1),
  O=>dout(1));

U2 : OBUF_LVDCI_25 port map(
  I=>dout_int(2),
  O=>dout(2));

U3 : OBUF_LVDCI_25 port map(
  I=>dout_int(3),
  O=>dout(3));

K0 : OBUF_HSTL_I_DCI port map(
  I=>muxout_int(0),
  O=>muxout(0));

K1 : OBUF_HSTL_I_DCI port map(
  I=>muxout_int(1),
  O=>muxout(1));

K2 : OBUF_HSTL_I_DCI port map(
  I=>muxout_int(2),
  O=>muxout(2));

K3 : OBUF_HSTL_I_DCI port map(
  I=>muxout_int(3),
  O=>muxout(3));

end dci_arch;
Digitally Controlled Impedance (DCI)

DCI in Virtex-II Pro/Virtex-II Pro X Hardware

DCI works with single-ended I/O standards and the 2.5V LVDS I/O standard. DCI supports the following Virtex-II Pro /Virtex-II Pro X standards:

LVDCI, LVDCI_DV2, GTL_DCI, GTLP_DCI, HSTL_I_DCI, HSTL_II_DCI, HSTL_III_DCI, HSTL_IV_DCI, HSTL_I_DCI_18, HSTL_II_DCI_18, HSTL_III_DCI_18, HSTL_IV_DCI_18, SSTL2_I_DCI, SSTL2_II_DCI, LVDS_25, and LVDSEXT_25.

To correctly use DCI in a Virtex-II Pro /Virtex-II Pro X device, users must follow the following rules:

1. V CCO pins must be connected to the appropriate V CCO voltage based on the IOSTANDARDs in that bank.
2. Correct DCI I/O buffers must be used in the software either by using IOSTANDARD attributes or instantiations in the HDL code.
3. External reference resistors must be connected to multipurpose pins (VRN and VRP) in the bank. These two multipurpose pins cannot be used as regular user I/Os. Refer to the Virtex-II Pro /Virtex-II Pro X pinouts for the specific pin locations. Pin VRN must be pulled up to V CCO by its reference resistor. Pin VRP must be pulled down to ground by its reference resistor.
4. The value of the external reference resistors should be selected to give the desired output impedance. If using GTL_DCI, HSTL_DCI, or SSTL_DCI I/O standards, then they should be 50Ω.
5. The values of the reference resistors must be within the supported range (20Ω – 100Ω).
6. Follow the DCI I/O banking rules:
   a. V REF must be compatible for all of the inputs in the same bank.
   b. V CCO must be compatible for all of the inputs and outputs in the same bank.
   c. No more than one DCI I/O standard using Single Termination type is allowed per bank.
   d. No more than one DCI I/O standard using Split Termination type is allowed per bank.
   e. Single Termination and Split Termination, Controlled Impedance Driver, and Controlled Impedance Driver with Half Impedance can co-exist in the same bank.
7. Avoid the simultaneous occurrence of these three conditions:
   a. The D0-D7, INIT, or DOUT pins are used as I/O after configuration
   b. One (or more) of those pins is a DCI input or output
   c. The programming file option "DCIUpdateMode" is set to "Quiet"

Otherwise, these I/Os will not function properly. In the case of an output, the I/O will be put into a 3-state condition, or will not have the correct termination. In the case of an input, the I/O will not have the correct termination.

This does not affect the RDWR_B and CS_B signals, which are also dual-purpose configuration pins.

8. "Quiet" mode set in programming file option "DCIUpdateMode" is not compatible with encrypted bitstream.

The behavior of DCI 3-state outputs is as follows:

If a LVDCI or LVDCI_DV2 driver is in 3-state, the driver is 3-stated. If a Driver with Single or Split Termination is in 3-state, the driver is 3-stated but the termination resistor remains.
The following section lists any special care actions that must be taken for each DCI I/O standard.

LVDCI_15, LVDCI_18, LVDCI_25, LVDCI_33

Using these buffers configures the outputs as controlled impedance drivers. The number extension at the end indicates the $V_{CCO}$ voltage that should be used. For example, 15 means $V_{CCO}=1.5V$, etc. There is no slew rate control or drive strength settings for LVDCI drivers.

LVDCI_DV2_15, LVDCI_DV2_18, LVDCI_DV2_25

Using these buffers configures the outputs as controlled drivers with half impedance. The number extension at the end indicates the $V_{CCO}$ voltage that should be used. For example, 15 means $V_{CCO}=1.5V$, etc. There is no slew rate control or drive strength settings for LVDCI_DV2 drivers.

GTL_DCI

GTL does not require a $V_{CCO}$ voltage. However, for GTL_DCI, $V_{CCO}$ must be connected to 1.2V. GTL_DCI provides single termination to $V_{CCO}$ for inputs or outputs.

GTLP_DCI

GTLP does not require a $V_{CCO}$ voltage. However, for GTLP_DCI, $V_{CCO}$ must be connected to 1.5V. GTLP_DCI provides single termination to $V_{CCO}$ for inputs or outputs.

HSTL_I_DCI, HSTL_III_DCI, HSTL_I_DCI_18, HSTL_III_DCI_18

HSTL_I_DCI provides split termination to $V_{CCO}/2$ for inputs. HSTL_III_DCI provides single termination to $V_{CCO}$ for inputs.

HSTL_II_DCI, HSTL_IV_DCI, HSTL_II_DCI_18, HSTL_IV_DCI_18

HSTL_II_DCI provides split termination to $V_{CCO}/2$ for inputs or outputs. HSTL_IV_DCI provides single termination to $V_{CCO}$ for inputs or outputs.

SSTL2_I_DCI, SSTL2_II_DCI

SSTL2_I_DCI and SSTL2_II_DCI provide split termination to $V_{CCO}/2$ for inputs. These I/O standards provide SSTL compatibility. Note that SSTL2_I_DCI and SSTL2_II_DCI have $V_{OH}$ and $V_{OL}$ levels that are different than non-DCI SSTL2_I and SSTL2_II levels. Answer Record 13012, available at support.xilinx.com, provides additional information on using DCI.

LVDS_25_DCI, LVDSEXT_25_DCI

LVDS_25_DCI and LVDSEXT_25_DCI provide split termination for the P and N inputs only. VRP and VRN should connect to 50Ω resistors. Equivalently, it provides 100Ω differential impedance between the LVDS inputs.

DCI Usage Examples

- Figure 3-105 provides examples illustrating the use of the HSTL_I_DCI, HSTL_II_DCI, HSTL_III_DCI, and HSTL_IV_DCI I/O standards.
- Figure 3-106 provides examples illustrating the use of the SSTL2_I_DCI and SSTL2_II_DCI I/O standards.
- Figure 3-107 provides examples illustrating the use of the LVDS_25_DCI and LVDSEXT_25_DCI I/O standards.
- Figure 3-108 provides examples illustrating the use of the GTL_DCI and GTLP_DCI I/O standards.

**Figure 3-105: HSTL DCI Usage Examples**
### Figure 3-106: SSTL DCI Usage Examples

<table>
<thead>
<tr>
<th>SSTL2_I or SSTL18_I</th>
<th>SSTL2_II or SSTL18_II</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Conventional</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>DCI Transmit</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Conventional</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Receive</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Conventional</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Transmit</strong></td>
<td></td>
</tr>
<tr>
<td><strong>DCI Receive</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Conventional</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Transmit</strong></td>
<td></td>
</tr>
<tr>
<td>**DCI **</td>
<td></td>
</tr>
<tr>
<td><strong>Receive</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Bidirectional</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Reference</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Resistor</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Recommended</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Z_0</strong>(1)**</td>
<td></td>
</tr>
<tr>
<td><strong>50Ω</strong></td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**
1. The SSTL-compatible 25Ω series resistor is accounted for in the DCI buffer, and it is not DCI controlled.
2. Z_0 is the recommended PCB trace impedance.
**Digitally Controlled Impedance (DCI)**

**Figure 3-107: LVDS DCI Usage Examples**

**NOTE:** Only LVDS25_DCI is supported (VCCO = 2.5V only)
### Figure 3-108: GTL/GTLP DCI Usage Examples

<table>
<thead>
<tr>
<th>DCI Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional</td>
<td>DCI Transmit, Conventional Receive</td>
</tr>
<tr>
<td>Conventional</td>
<td>DCI Transmit, Conventional Receive</td>
</tr>
<tr>
<td>DCI Transmit</td>
<td>DCI Receive, Conventional Transmit</td>
</tr>
<tr>
<td>DCI Transmit</td>
<td>DCI Receive, Conventional Transmit</td>
</tr>
<tr>
<td>Bidirectional</td>
<td></td>
</tr>
<tr>
<td>Reference Resistor</td>
<td>VRN = VRP = R = Z₀</td>
</tr>
<tr>
<td>Recommended Z₀</td>
<td>50Ω</td>
</tr>
</tbody>
</table>

**Notes:**
1. Z₀ is the recommended PCB trace impedance.
Double-Data-Rate (DDR) I/O

Introduction

Virtex-II Pro / Virtex-II Pro X devices have dedicated registers in a single IOB to implement input, output, and output with 3-state control Double-Data-Rate (DDR) registers. Input and output DDR is directly accomplished by the two registers on each path, clocked by the rising edges (or falling edges) from two different clock nets. The two clock signals are generated by the DCM and must be 180 degrees out of phase. Output DDR with 3-state requires the use of four registers in the IOB clocked in a similar fashion. Since the introduction of DLLs, Xilinx devices can generate low-skew clock signals that are 180 degrees out of phase, with a 50/50 duty cycle. These clocks reach the DDR registers in the IOB via dedicated routing resources.

Data Flow

Input DDR

Input DDR is accomplished via a single input signal driving two registers in the IOB. Both registers are clocked on the rising edge of their respective clocks. With proper clock forwarding, alternating bits from the input signal are clocked in on the rising edge of the two clocks, which are 180 degrees out of phase. Figure 3-109 depicts the input DDR registers and the signals involved.
CLK0 and CLK1 are 180 degrees out of phase. Both registers share the SET/PRE and RESET/CLR lines. As shown in Figure 3-110, alternating bits on the DATA line are clocked in via Q0 and Q1 while CE is High. The clocks are shifted out of phase by the DCM (CLK0 and CLK180 outputs) or by the inverter available on the CLK1 clock input.

Figure 3-109: Input DDR
Output DDR

Output DDR registers are used to clock output from the chip at twice the throughput of a single rising-edge clocking scheme. Clocking for output DDR is the same as input DDR. The clocks driving both registers are 180 degrees out of phase. The DDR MUX selects the register outputs. The output consists of alternating bits from DATA_1 and DATA_2. Figure 3-111 depicts the output DDR registers and the signals involved.
Both registers share the SET/PRE and RESET/CLR line. Both registers share the CE line which must be High for outputs to be seen on Q1 and Q2. Figure 3-112 shows the data flow for the output DDR registers.

![Output DDR Timing Diagram](image_url)

*Figure 3-112: Output DDR Timing Diagram*
Output DDR With 3-State Control

The 3-state control allows the output to have one of two values, either the output from the DDR MUX or high impedance. The Enable signal is driven by a second DDR MUX (Figure 3-113). This application requires the instantiation of two output DDR primitives.

![Diagram](image-url)
All four registers share the SET/PRESET and RESET/CLEAR lines. Two registers are required to accomplish the DDR task and two registers are required for the 3-state control. There are two Clock Enable signals, one for output DDRs performing the DDR function and another for the output DDRs performing the 3-state control function. Two 180 degree out of phase clocks are used. CLK1 clocks one of the DDR registers and a 3-state register. CLK2 clocks the other DDR register and the other 3-state register.

The DDR registers and 3-state registers are associated by the clock that is driving them. Therefore, the DDR register that is clocked by CLK1 is associated to the 3-state register being clocked by CLK1. The remaining two registers are associated by CLK2. If both 3-state registers are driving a logic High, the output sees a high impedance. If both 3-state registers are driving a logic Low, the output sees the values from the DDR MUX (see Figure 3-114).

![Timing Diagram for Output DDR With 3-State Control](image)

Figure 3-114: Timing Diagram for Output DDR With 3-State Control

When the 3-state registers are not driving the same logic value, the 3-state register being clocked by CLK1 is called TREG1. The other 3-state register TREG2 is clocked by CLK2. Similarly, the DDR register being clocked by CLK1 is called DREG1, and the other DDR register DREG2 is clocked by CLK2. If TREG1 is driving a logic High and TREG2 is driving a logic Low, the output sees a high impedance when CLK1 is High and the value out of DREG2 when CLK2 is High. If TREG2 is driving a logic High and TREG1 is driving a logic Low, the output sees a high impedance when CLK2 is High and the value out of DREG1 when CLK1 is High.
Characteristics

- All registers in an IOB share the same SET/PRE and RESET/CLR lines.
- The 3-State and Output DDR registers have common clocks (OTCLK1 & OTCLK2).
- All signals can be inverted (with no added delay) inside the IOB.
- DDR MUXing is handled automatically within the IOB. There is no manual control of the MUX-select. This control is generated from the clock.
- When several clocks are used, and when using DDR registers, the floorplan of a design should take into account that the input clock to an IOB is shared with a pair of IOBs.

Library Primitives

Input DDR registers are inferred, and dedicated output DDR registers have been provided as primitives for Virtex-II Pro / Virtex-II Pro X designs. Input DDR registers consist of two inferred registers that clock in a single data line on each edge. Generating 3-state output with DDR registers is as simple as instantiating a primitive.

![FDDRRSE Symbol: DDR Flip-Flop With Clock Enable and Synchronous Reset and Set](image1)

**Figure 3-115:** FDDRRSE Symbol: DDR Flip-Flop With Clock Enable and Synchronous Reset and Set

![FDDRCPE Symbol: DDR Flip-Flop With Clock Enable and Asynchronous PRESET and CLR](image2)

**Figure 3-116:** FDDRCPE Symbol: DDR Flip-Flop With Clock Enable and Asynchronous PRESET and CLR
VHDL and Verilog Instantiation

Examples are available in “VHDL and Verilog Templates,” page 250.

In VHDL, each template has a component declaration section and an architecture section. Each part of the template should be inserted within the VHDL design file. The port map of the architecture section should include the design signal names.

Constraints file syntax is provided where input registers need to be used. These settings force the input DDR registers into the IOB. The output registers should be instantiated and do not require any constraints file syntax to be pushed into the IOB.

Port Signals

FDDRRSE

Data Inputs - D0 and D1

D0 and D1 are the data inputs into the DDR flip-flop. Data on the D0 input is loaded into the flip-flop when R and S are Low and CE is High during a Low-to-High C0 clock transition. Data on the D1 input is loaded into the flip-flop when R and S are Low and CE is High during a Low-to-High C1 clock transition.

Clock Enable - CE

The enable pin affects the loading of data into the DDR flip-flop. When Low, new data is not loaded into the flip-flop. CE must be High to load new data into the flip-flop.

Clocks - C0 and C1

These two clocks are phase shifted 180 degrees (via the DLL) and allow selection of two separate data inputs (D0 and D1).

Synchronous Set - S
Synchronous Reset - R

The Reset (R) input, when High, overrides all other inputs and resets the output Low during any Low-to-High clock transition (C0 or C1). Reset has precedence over Set. When the Set (S) input is High and R is Low, the flip-flop is set, output High, during a Low-to-High clock transition (C0 or C1).

Data Output - Q

When power is applied, the flip-flop is asynchronously cleared and the output is Low. During normal operation, the value of Q is either D0 or D1. The Data Inputs description above states how the value of Q is chosen.

FDDRCPE

Data Inputs - D0 and D1

D0 and D1 are the data inputs into the DDR flip-flop. Data on the D0 input is loaded into the flip-flop when PRE and CLR are Low and CE is High during a Low-to-High C0 clock transition. Data on the D1 input is loaded into the flip-flop when PRE and CLR are Low and CE is High during a Low-to-High C1 clock transition.
Clock Enable - CE
The enable pin affects the loading of data into the DDR flip-flop. When Low, clock transitions are ignored and new data is not loaded into the flip-flop. CE must be High to load new data into the flip-flop.

Clocks - C0 and C1
These two clocks are phase shifted 180 degrees (via the DLL) and allow selection of two separate data inputs (D0 and D1).

Asynchronous Preset - PRE
Asynchronous Clear - CLR
The Preset (PRE) input, when High, sets the Q output High. When the Clear (CLR) input is High, the output is reset to Low.

Data Output - Q
When power is applied, the flip-flop is asynchronously cleared and the output is Low. During normal operation, The value of Q is either D0 or D1. The Data Inputs description above states how the value of Q is chosen.

Initialization in VHDL or Verilog
Output DDR primitives can be initialized in VHDL or Verilog code for both synthesis and simulation. For synthesis, the attributes are attached to the output DDR instantiation and are copied in the EDIF output file to be compiled by Xilinx tools. The VHDL code simulation uses a generic parameter to pass the attributes. The Verilog code simulation uses the defparam parameter to pass the attributes.

The DDR code examples (in VHDL ad Verilog) illustrate the following techniques.

Location Constraints
DDR instances can have LOC properties attached to them to constrain pin placement.

The LOC constraint uses the following form.
```
NET <net_name> LOC=A8;
```
where “A8” is a valid I/O pin location.

This device has a shared routing resource connecting the ICLK and OTCLK pins on pairs of IOBs. If two adjacent IOBs using DDR registers do not share the same clock signals on their clock pins (ICLK1, ICLK2, OTCLK1, and OTCLK2), or a DDR register locates next to a single data rate register that uses a different clock, one of the clock signals will be unroutable.

The IOB pairing is identical to the LVDS IOB pairs. Hence, the package pin-out table can also be used for pin assignment to avoid conflict.

Applications

DDR SDRAM
The DDR SDRAM is an enhancement to the Synchronous DRAM by effectively doubling the data throughput of the memory device. Commands are registered at every positive
clock edge. Input data is registered on both edges of the data strobe, and output data is referenced to both edges of the data strobe, as well as both edges of the clock.

Clock Forwarding

DDR can be used to forward a copy of the clock on the output. This can be useful for propagating a clock along with double-data-rate data that has an identical delay. It is also useful for multiple clock generation, where there is a unique clock driver for every clock load.

VHDL and Verilog Templates

VHDL and Verilog templates are available for output, output with 3-state enable, and input DDR registers.

Input DDR

To implement an Input DDR application, paste the following template in your code.

**DDR_input.vhd**

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity DDR_Input is
  Port (
    clk : in std_logic;
    d : in std_logic;
    rst : in std_logic;
    q1 : out std_logic;
    q2 : out std_logic
  );
end DDR_Input;

architecture behavioral of DDR_Input is
begin
  q1reg : process (clk, d, rst)
  begin
    if rst='1' then --asynchronous reset, active high
      q1 <= '0';
    elsif clk'event and clk='1' then  --Clock event - posedge
      q1 <= d;
    end if;
  end process;

  q2reg : process (clk, d, rst)
  begin
    if rst='1' then --asynchronous reset, active high
      q2 <= '0';
    end if;
  end process;
```

Double-Data-Rate (DDR) I/O

module DDR_Input (data_in, q1, q2, clk, rst);
input data_in, clk, rst;
output q1, q2;
reg q1, q2;

//Describe input DDR registers (behaviorally) to be inferred
always @ (posedge clk or posedge rst) //rising-edge DDR reg. and asynchronous reset
begin
  if (rst)
    q1 = 1'b0;
  else
    q1 = data_in;
end
always @ (negedge clk or posedge rst) //falling-edge DDR reg. and asynchronous reset
begin
  if (rst)
    q2 = 1'b0;
  else
    q2 = data_in;
end
endmodule

/* NOTE: You must include the following constraints in the .ucf file when running back-end tools, \ in order to ensure that IOB DDR registers are used:
INST "q2_reg" IOB=TRUE;
INST "q1_reg" IOB=TRUE;
Depending on the synthesis tools you use, it may be required to check the edif file for modifications to original net names...in this case, Synopsys changed the names: q1 and q2 to q1_reg and q2_reg
INST "q2_reg" IOB=TRUE;
INST "q1_reg" IOB=TRUE;

Depending on the synthesis tools you use, it may be required to check
the edif file for modifications to
original net names...in this case, Synopsys changed the names: q1 and q2
to q1_reg and q2_reg

*/

Output DDR

To implement an Output DDR application, paste the following template in your code.

**DDR_out.vhd**

```vhdl
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- pragma translate_off
LIBRARY UNISIM;
use UNISIM.VCOMPONENTS.ALL;
--pragma translate_on

entity DDR_Output is
  Port(
    clk : in std_logic; --clk and clk180 can be outputs from the DCM or
    clk180 : in std_logic;  --logical inverse of clk (the inverter is
    located in the IOB and will be inferred.
    d0 : in std_logic;  --data in to fddr
    d1 : in std_logic;  --data in to fddr
    ce : in std_logic;  --clock enable
    rst : in std_logic;  --reset
    set : in std_logic;  --set
    q : out std_logic  --DDR output
  );
end DDR_Output;

architecture behavioral of DDR_Output is

component FDDRRSE
  port(
    Q  : out std_logic;
    D0 : in std_logic;
    D1 : in std_logic;
    C0 : in std_logic;
    C1 : in std_logic;
    CE : in std_logic;
    R  : in std_logic;
    S : in std_logic
  );
end component;

begin

U0: FDDRRSE
  port map (  
    Q => q,
`
Double-Data-Rate (DDR) I/O

D0 => d0,
D1 => d1,
C0 => clk,
C1 => clk180,
CE => ce,
R => rst,
S => set
);

end behavioral;

DDR_out.v

module DDR_Output (d0, d1, q, clk, clk180, rst, set, ce);
input d0, d1, clk, clk180, rst, set, ce;
output q;

//Synchronous Output DDR primitive instantiation
FDDRRSE U1 ( .D0(d0),
               .D1(d1),
               .C0(clk),
               .C1(clk180),
               .CE(ce),
               .R(rst),
               .S(set),
               .Q(q)
               );
endmodule

Output DDR With 3-State Enable

To implement an Output DDR with 3-state Enable, paste the following template in your code:

DDR_3state.vhd

library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
-- pragma translate_off
LIBRARY UNISIM;
use UNISIM.VCOMPONENTS.ALL;
--pragma translate_on

entity DDR_3state is
Port(
  clk : in std_logic; --clk and clk180 can be outputs from the DCM or
clk180 : in std_logic; --logical inverse of clk (the inverter is
located in the IOB and will be inferred.
d0 : in std_logic; --data in to fddr
d1 : in std_logic; --data in to fddr
ce : in std_logic; --clock enable
set : in std_logic; --set
rst : in std_logic; --reset
);
en0 : in std_logic; --enable signal
en1 : in std_logic; --enable signal
data_out : out std_logic --data seen at pad
);
end DDR_3state;

architecture behavioral of DDR_3state is

signal ddr_out, tri : std_logic;

component FDDRRSE
  port (  
    Q : out std_logic;
    D0 : in std_logic;
    D1 : in std_logic;
    C0 : in std_logic;
    C1 : in std_logic;
    CE : in std_logic;
    R  : in std_logic;
    S  : in std_logic
  );
end component;

begin

  --Instantiate Ouput DDR registers
  U0: FDDRRSE port map(Q  => tri,
    D0 => en0,
    D1 => en1,
    C0 => clk,
    C1 => clk180,
    CE => ce,
    R  => rst,
    S  => set
  );

  --Instantiate three-state DDR registers
  U1: FDDRRSE port map( Q => ddr_out,
    D0 => d0,
    D1 => d1,
    C0 => clk,
    C1 => clk180,
    CE => ce,
    R  => rst,
    S  => set
  );

  --inferr the 3-State buffer
  process(tri, ddr_out)
  begin
    if tri = '1' then
      data_out <= 'Z';
    elsif tri = '0' then
      data_out <= ddr_out;
    end if;
  end process;
Double-Data-Rate (DDR) I/O

DDR_3state.v

module DDR_3state (d0, d1, data_out, en_0, en_1, clk, clk180, rst, set, ce);

input d0, d1, clk, clk180, rst, set, ce, en_0, en_1;

output data_out;
reg data_out;

wire q, q_tri;

//Synchronous Output DDR primitive instantiation
FDDRRSE U1 ( .D0(d0),
   .D1(d1),
   .C0(clk),
   .C1(clk180),
   .CE(ce),
   .R(rst),
   .S(set),
   .Q(q)
 );

//Synchronous 3-State DDR primitive instantiation
FDDRRSE U2 ( .D0(en_0),
   .D1(en_1),
   .C0(clk),
   .C1(clk180),
   .CE(ce),
   .R(rst),
   .S(set),
   .Q(q_tri)
 );

//3-State buffer description
always @(q_tri or q)
begin
   if (q_tri)
      data_out = 1'b1;
   else
      data_out = q;
end
endmodule
LVDS I/O

Introduction

Low Voltage Differential Signaling (LVDS) is a very popular and powerful high-speed interface in many system applications. Virtex-II Pro / Virtex-II Pro X I/Os are designed to comply with the EIA/TIA electrical specifications for LVDS to make system and board design easier. With the addition of an LVDS current-mode driver in the IOBs, which eliminates the need for external source termination in point-to-point applications, and with the choice of an extended mode, Virtex-II Pro / Virtex-II Pro X devices provide the most flexible solution for doing an LVDS design in an FPGA.

Table 3-58 lists all LVDS standards that are available for Virtex-II Pro / Virtex-II Pro X devices.

Table 3-58: Available LVDS Standards

<table>
<thead>
<tr>
<th>IOSTANDARD</th>
<th>Input (IBUFDS)</th>
<th>Output (OBUFDS)</th>
<th>3-State (OBUFTDS)</th>
<th>Clock (IBUFGDS)</th>
<th>Bi-Directional (IOBUF)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LVDS_25</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LVDSEXT_25(1)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LVDS_25_DCI</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>LVDSEXT_25_DCI(1)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>BLVDS_25</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDT_25</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVPECL_25</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ULVDS_25</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LDT_25_DT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVDSEXT_25_DT(1)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>LVDS_25_DT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ULVDS_25_DT</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes:
1. LVDSEXT = Extended mode LVDS buffer. This buffer provides a higher drive capability and voltage swing (350 - 750 mV), which makes it ideal for long-distance or cable LVDS links. The output AC characteristics of this LVDS Extended Mode driver are not within the EIA/TIA specifications. This LVDS Extended Mode driver is intended for situations that require higher drive capabilities in order to produce an LVDS signal that is within EIA/TIA specification at the receiver.

Creating an LVDS Input/Clock Buffer

Figure 3-117 illustrates the LVDS input and clock buffer primitives shown in Table 3-59. The pin names used are the same as those used in the HDL library primitives.

Figure 3-117: LVDS Input and Clock Primitives
To create an LVDS input, instantiate the desired mode (2.5V or Extended) LVDS input buffer. Notice that the P and N channels are included in the primitive (I = P, IB = N). Software automatically uses the appropriate pin from an adjacent IOB for the N channel. The same applies to LVDS clocks: Use IBUFGDS_LVDS*

### LVDS Input HDL Examples

#### VHDL Instantiation

```vhdl
U1: IBUFDS_LVDS_25
code
  port map (
    I => data_in_P,
    IB => data_in_N
    O => data_in
  );
```

#### Verilog Instantiation

```verilog
IBUFDS_LVDS_25 U1 (.I(data_in_P),
                   .IB(data_in_N),
                   .O(data_in)
  );
```

### Port Signals

- **I** = P-channel data input to the LVDS input buffer
- **IB** = N-channel data input to the LVDS input buffer
- **O** = Non-differential input data from LVDS input buffer

### Location Constraints

```vhdl
NET "data_in_P" LOC= "<pin_location>";
```
LVDS Receiver Termination

Virtex-II Pro / Virtex-II Pro X provides a true 100Ω differential termination (DT) across the input differential receiver terminals. The LVDS_25_DT, LVDSEXT_25_DT, LDT_25_DT, and ULVDS_25_DT standards support on-chip differential termination.

The on-chip input differential termination in Virtex-II Pro / Virtex-II Pro X provides major advantages over the external resistor or the DCI termination solution:

- Eliminates the stub at the receiver completely and therefore greatly improve signal integrity
- Consumes less power than DCI termination
- Supports LDT (not supported by DCI termination)
- Frees up VRP/VRN pins

Figure 3-118 is an example of a typical termination for an LVDS receiver on a board with 50Ω transmission lines.

Differential Termination LVDS Input HDL Examples

VHDL Instantiation

```vhdl
U1: IBUFDS_LVDS_25_DT
  port map (I => data_in_P, IB => data_in_N, O => data_in);
```

Verilog Instantiation

```verilog
IBUFDS_LVDS_25_DT U1 (I(data_in_P), IB(data_in_N), O(data_in));
```

Creating an LVDS Output Buffer

Figure 3-119 illustrates the LVDS output buffer primitives:

- **OBUFDS_LVDS_25**
- **OBUFDS_LVDSEXT_25**
The pin names used are the same as those used in the HDL library primitives.

![LVDS Output Buffer Primitives](UG002_C2_032_100200)

**Figure 3-119: LVDS Output Buffer Primitives**

To create an LVDS output, instantiate the desired mode (2.5 or Extended) LVDS output buffer. Notice that the P and N channels are included in the primitive (O = P, OB = N). Software automatically uses the appropriate pin from an adjacent IOB for the N channel.

**LVDS Output HDL Examples**

**VHDL Instantiation**

```vhdl
U1: OBUFDS_LVDS_25
   port map (
      I  => data_out,
      O  => data_out_P,
      OB => data_out_N
   );
```

**Verilog Instantiation**

```verilog
OBUFDS_LVDS_25 U1 ( .I(data_out),
                     .O(data_out_P),
                     .OB(data_out_N)
   );
```

**Port Signals**

- **I** = data input to the LVDS input buffer
- **O** = P-channel data output
- **OB** = N-channel data output

**Location Constraints**

```vhdl
NET "data_out_P" LOC= "<pin_location>";
```

**LVDS Transmitter Termination**

The Virtex-II Pro / Virtex-II Pro X LVDS transmitter does not require any termination. **Table 3-58, page 256** lists primitives that correspond to the LVDS current-mode drivers. LVDS current-mode drivers are a true current source and produce the proper (EIA/TIA compliant) LVDS signal. **Figure 3-120** illustrates an LVDS transmitter on a board with 50Ω transmission lines.
Creating an LVDS Output 3-State Buffer

Figure 3-121 illustrates the LVDS 3-state buffer primitives:

- OBUFTDS_LVDS_25
- OBUFTDS_LVDSEXT_25

The pin names used are the same as those used in the HDL library primitives.

LVDS 3-State HDL Example

VHDL Instantiation

```vhdl
U1: OBUFTDS_LVDS_25
    port map (  
        I => data_out,  
        T => tri,  
        O => data_out_P,  
        OB => data_out_N  
    );
```
Verilog Instantiation

```
OBUFTDS_LVDS_25 u1 (.I(data_out),
                       .T(tri),
                       .O(data_out_P),
                       .OB(data_out_N));
```

Port Signals

- **I** = data input to the 3-state output buffer
- **T** = 3-State control signal
- **O** = P-channel data output
- **OB** = N-channel data output

Location Constraints

```
NET "data_out_P" LOC = "<pin_location>";
```

LVDS 3-State Termination

The Virtex-II Pro / Virtex-II Pro X LVDS 3-state buffer does not require any termination. Table 3-58, page 256 lists primitives that correspond to LVDS current-mode drivers. These drivers are a true current source, and they produce the proper (EIA/TIA compliant) LVDS signal. Figure 3-122 illustrates a simple redundant point-to-point LVDS solution with two LVDS 3-state transmitters sharing a bus with one LVDS receiver and the required termination for the circuit.

![LVDS 3-State Termination Diagram](UG012_C2_030_121101)

Creating a Bidirectional LVDS Buffer

Since LVDS is intended for point-to-point applications, BLVDS (Bus-LVDS) is not an EIA/TIA standard implementation and requires careful adaptation of I/O and PCB layout design rules. The primitive supplied in the software library for bi-directional LVDS does not use the Virtex-II Pro / Virtex-II Pro X LVDS current-mode driver. Therefore, source termination is required. Refer to XAPP243 for examples of BLVDS termination. Refer to Table 3-36 for simultaneous-switching output guideline for BLVDS drivers.

The following are VHDL and Verilog instantiation examples of Virtex-II Pro / Virtex-II Pro X BLVDS primitives.
VHDL Instantiation

```vhdl
blvds_io: IOBUFDS_BLVDS_25
port map (
    I => data_out,
    O => data_in,
    T => tri,
    IO => data_IO_P,
    IOB => data_IO_N
);
```

Verilog Instantiation

```verilog
IOBUFDS_BLVDS_25  blvds_io (  .I(data_out),
                               .O(data_in),
                               .T(tri),
                               .IO(data_IO_P),
                               .IOB(data_IO_N)
);
```

Port Signals

- **I** = data output: internal logic to LVDS I/O buffer
- **T** = 3-State control to LVDS I/O buffer
- **IO** = P-channel data I/O to or from BLVDS pins
- **IOB** = N-channel data I/O to or from BLVDS pins
- **O** = Data input: off-chip data to LVDS I/O buffer

Location Constraints

Only the P or N channel must be constrained. Software automatically places the corresponding channel of the pair on the appropriate pin.

---

**LDT**

Lightning Data Transport (LDT) is a high speed interface and protocol introduced by Advanced Micro Devices. LDT is a differential signaling based interface that is very similar to LVDS. Virtex-II Pro / Virtex-II Pro X IOBs are equipped with LDT buffers. These buffers also have corresponding software primitives as follows:

- **IBUFDS_LDT_25**
- **IBUFGDS_LDT_25**
- **OBUFDS_LDT_25**
- **OBUFTDS_LDT_25**

**LDT Implementation**

LDT implementation is the same as LVDS with DDR, so follow all of the rules and guidelines set forth earlier in this chapter for LVDS-DDR, and replace the LVDS buffer with the corresponding LDT buffer. For more information on Virtex-II Pro / Virtex-II Pro X LDT electrical specification, refer to the [Data Sheet](#).
LVPECL I/O

Introduction

Low Voltage Positive Emitter-Coupled Logic (LVPECL) is a very popular and powerful high-speed interface in many system applications. Virtex-II Pro / Virtex-II Pro X I/Os are designed to comply with the EIA/TIA electrical specifications for 2.5V LVPECL to make system and board design easier.

Table 3-60 lists all LVPECL primitives that are available for Virtex-II Pro / Virtex-II Pro X devices.

Table 3-60: Available LVPECL Primitives

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>3-State</th>
<th>Clock</th>
<th>Bi-Directional</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBUFDS_LVPECL_25</td>
<td>OBUFDS_LVPECL_25</td>
<td>OBUFTDS_LVPECL_25</td>
<td>IBUFGDS_LVPECL_25</td>
<td></td>
</tr>
</tbody>
</table>

Creating an LVPECL Input/Clock Buffer

Figure 3-123 illustrates the LVPECL input and clock buffer primitives shown in Table 3-61. The pin names used are the same as those used in the HDL library primitives.

Table 3-61: LVPECL Input and Clock Buffer Primitives

<table>
<thead>
<tr>
<th>LVPECL Inputs</th>
<th>LVPECL Clocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>IBUFDS_LVPECL_25</td>
<td>IBUFGDS_LVPECL_25</td>
</tr>
</tbody>
</table>

IBUFDS_LVPECL*/IBUFGDS_LVPECL*

Figure 3-123: LVPECL Input and Clock Primitives

To create an LVPECL input, instantiate the 2.5V LVPECL input buffer. Notice that the P and N channels are included in the primitive (I = P, IB = N). Software automatically uses the appropriate pin from an adjacent IOB for the N channel. The same applies to LVPECL clocks: Use IBUFGDS_LVPECL*.

LVPECL Input HDL Examples

VHDL Instantiation

```vhdl
U1: IBUFDS_LVPECL_25
   port map (I => data_in_P,
             IB => data_in_N,
             O => data_in);
```

ug012_c2_067_082902
Verilog Instantiation

```verilog
IBUFDS_LVPECL_25 U1 (.I(data_in_P),
                .IB(data_in_N),
                .O(data_in)
);
```

Port Signals

- \( I \) = P-channel data input to the LVPECL input buffer
- \( IB \) = N-channel data input to the LVPECL input buffer
- \( O \) = Non-differential input data from LVPECL input buffer

Location Constraints

```verilog
NET "data_in_P" LOC= "<pin_location>";
```

LVPECL Receiver Termination

All LVPECL receivers require standard termination. Figure 3-124 is an example of a typical termination for an LVPECL receiver on a board with 50\( \Omega \) transmission lines.

![LVPECL Receiver Termination](image)

**Figure 3-124:** LVPECL Receiver Termination

Creating an LVPECL Output Buffer

Figure 3-125 illustrates the LVDS output buffer primitives:

- \( \text{OBUFDS_LVPECL}_25 \)
- \( \text{OBUFTDS_LVPECL}_25 \)

The pin names used are the same as those used in the HDL library primitives.

![LVPECL Output Buffer Primitives](image)

**Figure 3-125:** LVPECL Output Buffer Primitives

To create an LVPECL output, instantiate the desired mode LVPECL output buffer. Notice that the P and N channels are included in the primitive \( (O = P, OB = N) \). Software
automatically uses the appropriate pin from an adjacent IOB for the N channel. Refer to Table 3-36 for simultaneous switching output guideline for LVPECL drivers.

**LVPECL Output HDL Examples**

**VHDL Instantiation**

```vhdl
U1: OBUFDS_LVPECL_25
    port map ( 
        I => data_out, 
        O => data_out_P, 
        OB => data_out_N 
    );
```

**Verilog Instantiation**

```verilog
OBUFDS_LVPECL_25U1 ( .I(data_out),
                      .O(data_out_P),
                      .OB(data_out_N)
);
```

**Port Signals**

- **I** = data input to the LVPECL input buffer
- **O** = P-channel data output
- **OB** = N-channel data output

**Location Constraints**

```vhdl
NET "data_out_P" LOC= "<pin_location>";
```

**LVPECL Transmitter Termination**

The Virtex-II Pro/Virtex-II Pro X LVPECL transmitter requires the termination shown in Figure 3-126, which illustrates an LVPECL transmitter on a board with 50Ω transmission lines. Table 3-60 lists primitives that correspond to the LVPECL drivers.

![LVPECL Transmitter Termination Diagram](ug012_c2_66_082002)

*Figure 3-126: LVPECL Transmitter Termination*
Bitstream Encryption

Virtex-II Pro / Virtex-II Pro X devices have an on-chip decryptor that can be enabled to make the configuration bitstream (and thus the whole logic design) secure. The user can encrypt the bitstream in the Xilinx software, and the FPGA then performs the reverse operation, decrypting the incoming bitstream, and internally recreating the intended configuration. This method provides a very high degree of design security. Without knowledge of the encryption/decryption key or keys, potential pirates cannot use the externally intercepted bitstream to analyze, or even to clone the design. System manufacturers can be sure that their Virtex-II Pro / Virtex-II Pro X implemented designs cannot be copied and reverse engineered.

The Virtex-II Pro / Virtex-II Pro X devices store the internal decryption keys in a few hundred bits of dedicated RAM, backed up by a small externally connected battery. At <100 nA load, the endurance of the battery is only limited by its shelf life. Refer to Appendix C, “Choosing the Battery for VBATT” for suggestions on choosing a backup battery.

The method used to encrypt the data is Data Encryption Standard (DES). This is an official standard supported by the National Institute of Standards and Technology (NIST) and the U. S. Department of Commerce. DES is a symmetric encryption standard that utilizes a 56-bit key. Because of the increased sophistication and speed of today’s computing hardware, single DES is no longer considered to be secure. However, the Triple Data Encryption Algorithm (TDEA), otherwise known as triple DES, is authorized for use by U. S. federal organizations to protect sensitive data and is used by many financial institutions to protect their transactions. Triple DES has yet to be cracked. Both DES and triple DES are available in Virtex-II Pro / Virtex-II Pro X devices.

What DES Is

DES and triple DES are symmetric encryption algorithms. This means that the key to encrypt and the key to decrypt are the same. The security of the data is kept by keeping the key secret. This contrasts to a public key system, like RSA or PGP. One thing to note is that Virtex-II Pro / Virtex-II Pro X devices use DES in Cipher Block Chaining mode. This means that each block is combined with the previous encrypted block for added security. DES uses a single 56-bit key to encrypt 64-bit blocks one at a time.

How Triple DES is Different

Triple DES uses three keys (known as a key bundle or key set), and the encryption algorithm is repeated for each of those keys. If $E_K(I)$ and $D_K(I)$ denote the encryption and decryption of a data block $I$ using key $K$, the Triple DES encryption algorithm is as follows (known as E-D-E):

$$\text{Output}_{\text{encrypted}} = E_{K_3}(D_{K_2}(E_{K_1}(I)))$$

And the decryption algorithm is as follows (known as D-E-D):

$$\text{Output}_{\text{decrypted}} = D_{K_1}(E_{K_2}(D_{K_3}(I)))$$

$K_1 = K_2 = K_3$ gives the same result as single DES.

For a detailed description of the DES standard, refer to:

Bitstream Encryption

For a popular description of the origin and the basic concept of DES and many other older and newer encryption schemes, see the recent best-seller:


Classification and Export Considerations

Virtex-II Pro / Virtex-II Pro X FPGAs have been classified by the U. S. Department of Commerce as an FPLD (3A001.a.7), which is the same classification as current FPGAs. Only the decryptor is on-chip and can only be used to decrypt an incoming bitstream, so the classification has not changed and no new paperwork is required. The software has been classified under ECCN#:5D002 and can be exported globally under license exception ENC. No changes to current export practices are necessary.

Creating Keys

For Virtex-II Pro / Virtex-II Pro X, DES or triple DES (TDEA) can be used. DES uses a single 56-bit key, where triple-DES always uses three such keys. All of the keys can be chosen by the BitGen program at random, or can be explicitly specified by the user.

Virtex-II Pro / Virtex-II Pro X devices can have six separate keys programmed into the device. A particular device can store two sets of triple-DES keys and can thus accept alternate bitstreams from two competing IP vendors, without providing access to each other’s design. However, all of the keys must be programmed at once.

An encrypted bitstream is created by the BitGen program. Keys and key options can be chosen in two ways: by command-line arguments to BitGen, or by specifying a KeyFile (with the –g KeyFile command-line option). The BitGen options relevant to encryption are listed in Table 3-62.

Table 3-62: BitGen Encryption Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Values (default first where appropriate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encrypt</td>
<td>Whether to encrypt the bitstream</td>
<td>No, Yes</td>
</tr>
<tr>
<td>Key0</td>
<td>DES Key 0</td>
<td>pick, &lt;hex string&gt;</td>
</tr>
<tr>
<td>Key1</td>
<td>DES Key 1</td>
<td>pick, &lt;hex string&gt;</td>
</tr>
<tr>
<td>Key2</td>
<td>DES Key 2</td>
<td>pick, &lt;hex string&gt;</td>
</tr>
<tr>
<td>Key3</td>
<td>DES Key 3</td>
<td>pick, &lt;hex string&gt;</td>
</tr>
<tr>
<td>Key4</td>
<td>DES Key 4</td>
<td>pick, &lt;hex string&gt;</td>
</tr>
<tr>
<td>Key5</td>
<td>DES Key 5</td>
<td>pick, &lt;hex string&gt;</td>
</tr>
<tr>
<td>KeyFile</td>
<td>Location of separate key definition file</td>
<td>&lt;string&gt;</td>
</tr>
<tr>
<td>Keyseq0</td>
<td>Set the key sequence for key 0 (S = single, F = first, M = middle, L = last)</td>
<td>S,F,M,L</td>
</tr>
<tr>
<td>Keyseq1</td>
<td>Set the key sequence for key 1</td>
<td>S,F,M,L</td>
</tr>
<tr>
<td>Keyseq2</td>
<td>Set the key sequence for key 2</td>
<td>S,F,M,L</td>
</tr>
<tr>
<td>Keyseq3</td>
<td>Set the key sequence for key 3</td>
<td>S,F,M,L</td>
</tr>
<tr>
<td>Keyseq4</td>
<td>Set the key sequence for key 4</td>
<td>S,F,M,L</td>
</tr>
</tbody>
</table>
The key sequence (Keyseq) is set to S for single key encryption, F for first key in multi-key encryption, M for middle key in multi-key encryption, and L for last key in multi-key encryption. When the KeyFile option is specified, BitGen looks in that file for all other DES key options listed above. An example for the input KeyFile using triple DES is:

```
# Comment for key file
Key 0 0x9ac28ebeb2d83b;
Key 1 pick;
Key 2 string for my key;
Key 3 0x0000000000000000;
Key 4 8774eb3ebb4f84;
Keyseq 0 F;
Keyseq 1 M;
Keyseq 2 L;
Keyseq 3 F;
Keyseq 4 M;
Keyseq 5 L;
Key StartCBC 503f2f655b1b2f82;
StartKey 0;
```

Every key is given in the output key file, with unused key locations set to "0x0000000000000000." The proper key sequence prefix is added for all used keys. The prefix is preserved for unused keys, if the user specified a value. The output key file has the same base file name as the .bit file, but with a .nky file extension.

The command line equivalent of the input key file above is as follows:

```
bitgen -g security:level1 -g Encrypt:Yes -g Key0: 0x9ac28ebeb2d83b -g Key1:pick -g Key2: string for my key -g Key3:0x0000000000000000 -g Key4:8774eb3ebb4f84 -g Keyseq0:F, -g Keyseq1:M, -g Keyseq2:L -g Keyseq3:F -g Keyseq4:M -g Keyseq5:L -g StartCBC:503f2f655b1b2f82 -g StartKey:0
```

If the key file is used, the command line is as follows:

```
bitgen -g security:level1 -g Encrypt:Yes -g KeyFile:mykeyfile myinput.ncd
```

The output key file from either of the above inputs looks something like this:

```
Device 2VP4FG456;
Key 0 0x9ac28ebeb2d83b;
Key 1 0xdb1adb5f08b972;
Key 2 0x5452032773c286;
Key 3 0x0000000000000000;
Key 4 0x8774eb3ebb4f84;
Key 5 0x0000000000000000;
Keyseq 0 F;
Keyseq 1 M;
Keyseq 2 L;
Keyseq 3 F;
Keyseq 4 M;
```

---

**Table 3-62: BitGen Encryption Options**

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
<th>Values (default first where appropriate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Keyseq5</td>
<td>Set the key sequence for key 5</td>
<td>S,F,M,L</td>
</tr>
<tr>
<td>StartKey</td>
<td>Key number to start decryption</td>
<td>0,3</td>
</tr>
<tr>
<td>StartCBC</td>
<td>Constant Block Chaining start value</td>
<td>pick, &lt;string&gt;</td>
</tr>
</tbody>
</table>
Bitstream Encryption

Keyseq 5 L;
Key StartCBC 0x503f2f655b1b2f82;
StartKey 0;

In the case of the string for Key2, if the keyvalue is a character string, BitGen encodes the string into a 56-bit hex string. The same character string gives the same 56-bit hex string every time. This enables passwords or phrases to be used instead of hex strings.

The above keys are all specified as 64 bits each. The first 8 bits are used by Xilinx as header information and the following 56 bits as the key. BitGen accepts 64 bit keys, but automatically overrides the header, if necessary.

Because of security issues, the –g Compress option cannot be used with bitstream encryption. Also, partial reconfiguration is not allowed.

In order to guarantee security of a bitstream, the Constant Block Chaining (CBC) initial value should be different for each design. The actual value of the CBC is not important, only that it be a different value for every different bitstream that is encrypted with the same key or every re-encryption of the same bitstream with a different key.

Loading Keys

DES keys can only be loaded through JTAG. The iMPACT™ tools have the capability to take a .nky file and program the device with the keys. In order to program the keys, a “key-access mode” is entered. When this mode is entered, all of the FPGA memory, including the keys and configuration data, is cleared. Once the keys are programmed, they cannot be reprogrammed without clearing the entire device. This “key access mode” is completely transparent to most users.

Keys are programmed using the ISC_PROGRAM instruction, as detailed in the JTAG 1532 specification. SVF generation is also supported, if keys are to be programmed using a different method, such as a microprocessor or JTAG test software.

Loading Encrypted Bitstreams

Once the device has been programmed with the correct keys, the device can be configured with an encrypted bitstream. Non-encrypted bitstreams may also be used to configure the device, and the stored keys are ignored. The method of configuration is not at all affected by encryption. Any of the modes may be used, and the signaling does not change (see Chapter 4, “Configuration”). However, all bitstreams must configure the entire device, since partial reconfiguration is not permitted.

Once the device has been configured with an encrypted bitstream, it cannot be reconfigured without toggling the PROG pin, cycling power, or performing the JTAG JPROG_B instruction. All of these events fully clear the configuration memory, but none of these events reset the keys as long as V_BATT or V_CCAUX are maintained.

V_BATT

V_BATT is a separate battery voltage to allow the keys to remain programmed in the Virtex-II Pro / Virtex-II Pro X device. V_BATT draws very little current (on the order of nA) to keep the keys programmed. A small watch battery is suitable. Refer to V_BATT DC Characteristics in the Data Sheet and the battery’s specifications to estimate its lifetime. For a detailed discussion of battery types suitable for use as a V_BATT source, see Appendix C, “Choosing the Battery for V_BATT”.
While the auxiliary voltage ($V_{CCAUX}$) is applied, $V_{BATT}$ does not draw any current, and the battery can be removed or exchanged.

**Temperature-Sensing Diode (DXP/DXN)**

The Virtex-II Pro FPGA *temperature-sensing diode* can be accessed through the DXP (Anode) and DXN (Cathode) pins. The DXP and DXN pins are wired internally to a diode-connected transistor, which creates a remote temperature sensor.

*Note:* These additional pins are not available on all packages.

DXP and DXN are dedicated pins attached to the substrate/die itself. They cannot be accessed through the software tools. They are always available, and no special actions in the design need be taken in order to use them. If the temperature-sensing feature is not used, the DXP and DXN pins can be left unconnected.

The temperature-sensing diode is one part of a two-part system. To complete the system, a *temperature sensor interface device* is also required. Most temperature sensor interface devices provide corresponding pins that can connect directly to the DXP and DXN pins on the Virtex-II Pro FPGA. Once the upper and lower temperature limits have been set up and the device is operated in conjunction with the sensing diode in the FPGA, the temperature sensor interface device creates an output signal when these bounds are exceeded. This output can be used as an interrupt to turn off the clock, turn on a fan, or perform some other operation intended to reduce heat.

The accuracy of the temperature measurement achieved by this two-part system does not depend on the temperature-sensing diode itself (DXN/DXP pins), because the voltage-versus-temperature curve is determined by the physical nature of the diode. Numerical readout accuracy relies instead upon the temperature sensor interface device, which translates the IV-versus-temperature curves into an actual temperature reading. The data sheets of the temperature sensors listed below contain the accuracy specifications.

**Examples of Temperature Sensors**

1. **Maxim Remote/Local Temperature Sensors.** For general information on these devices, go to the Maxim website at [http://www.maxim-ic.com](http://www.maxim-ic.com).

   Specifically, the data sheets for these devices can be found at:

   Refer to the "PC Board Layout" section of these data sheets for important design considerations.

2. **National Semiconductor LM83 or LM86 Triple-Diode Input and Local Digital Temperature Sensors with Two-Wire Interface.** For general information on these devices, go to the National Semiconductor website at [http://www.national.com](http://www.national.com).

   Specifically, the data sheets for these devices can be found at:

   Refer to the "Application Hints" section of these data sheets for important design considerations.
CORE Generator System

Introduction

This section on the Xilinx CORE Generator™ System and the Xilinx Intellectual Property (IP) Core offerings is provided as an overview of products that facilitate the Virtex-II Pro / Virtex-II Pro X design process. For more detailed and complete information, consult the CORE Generator Guide, which can be accessed online in the Xilinx software installation, as well as at the http://toolbox.xilinx.com/docsan/xilinx4/manuals.htm site under the “Design Entry Tools” heading.

The CORE Generator System

The Xilinx CORE Generator System is the cataloging, customization, and delivery vehicle for IP cores targeted to Xilinx FPGAs. This tool is included with all Xilinx ISE BaseX, ISE Foundation, and ISE Alliance Series software packages. The CORE Generator provides centralized access to a catalog of ready-made IP functions ranging in complexity from simple arithmetic operators, such as adders, accumulators, and multipliers, to system-level building blocks, such as filters, transforms, and memories. Cores can be displayed alphabetically, by function, by vendor, or by type. Each core comes with its own data sheet, which documents the core’s functionality in detail.

The CORE Generator User Interface (see Figure 3-127) has direct links to key Xilinx web support pages, such as the Xilinx IP Center website (www.xilinx.com/ipcenter) and Xilinx Technical Support, making it very easy to access the latest Virtex-II Pro / Virtex-II Pro X IP releases and get helpful, up-to-date specifications and information on technical issues. Links to partner IP providers are also built into the informational GUIs for the various partner-supplied AllianceCORE products described under “AllianceCORE Program,” page 276.

The use of CORE Generator IP cores in Virtex-II Pro / Virtex-II Pro X designs enables designers to shorten design time, and it also helps them realize high levels of performance and area efficiency without any special knowledge of the Virtex-II Pro / Virtex-II Pro X architecture. The IP cores achieve these high levels of performance and logic density by using Xilinx Smart-IP™ technology.
Smart-IP Technology

Smart-IP technology leverages Xilinx FPGA architectural features, such as look-up tables (LUTs), distributed RAM, segmented routing and floorplanning information, as well as relative location constraints and expert logic mapping to optimize the performance of every core instance in a given Xilinx FPGA design. In the context of Virtex-II Pro / Virtex-II Pro X cores, Smart-IP technology includes the use of the special high-performance Virtex-II Pro / Virtex-II Pro X architectural features, such as embedded 18x18 multipliers, block memory, shift register look-up tables (SRL16’s), and special wide mux elements.

Smart-IP technology delivers:

- Physical layouts optimized for high performance
- Predictable high performance and efficient resource utilization
- Reduced power requirements through compact design and interconnect minimization
- Performance independent of device size
- Ability to use multiple cores without deterioration of performance
- Reduced compile time over competing architectures
CORE Generator Design Flow

A block diagram of the CORE Generator design flow is shown in Figure 3-128.

Notes:
1. The outputs produced by the CORE Generator consist of an implementation Netlist and optional schematic symbol, HDL template files, and HDL simulation model wrapper files.

Core Types

Parameterized Cores

The CORE Generator System supplies a wide assortment of parameterized IP cores that can be customized to meet specific Virtex-II Pro / Virtex-II Pro X design needs and size constraints. See Figure 3-129. For each parameterized core, the CORE Generator System supplies:

- A customized EDIF implementation netlist (.EDN)
- A parameterized Verilog or VHDL behavioral simulation model (.V, .VHD) and corresponding wrapper file (also .V, .VHD)
- Verilog or VHDL templates (.VEO, .VHO)
- An ISE Foundation or Viewlogic® schematic symbol

The EDIF implementation netlist is used by the Xilinx tools to implement the core. The other design files generated depend on the Design Entry settings specified (target CAE vendor, and design flow type -- schematic or HDL). Schematic symbol files are generated when a schematic design flow is specified for the project.

Parameterized HDL simulation models are provided in two separate HDL simulation libraries, one for Verilog functional simulation support, and the other for VHDL functional simulation support. The libraries, which are included as part of the Xilinx installation, are in the following locations:
If using a compiled simulator, these libraries must be precompiled before performing a functional simulation of the cores. An analyze_order file describing the required compile order of these models is included with each XilinxCoreLib library, one for Verilog (verilog_analyze_order) and one for VHDL (vhdl_analyze_order).

For an HDL design flow, Verilog and VHDL templates (.VEO and .VHO files) are also provided to facilitate the integration of the core into the design for the purposes of functional simulation, synthesis, and implementation. The Verilog (.V) and VHDL (.VHD) wrapper files are also generated. The wrapper files for a particular core are compiled like normal simulation models. They convey custom parameter values to the corresponding generic, parameterized behavioral model for that core in the XilinxCoreLib library. The custom parameter values are used to tailor the behavior of the customized core.

The following is a sample VHO template:

```vhdl
component adder8
  port (  
a: IN std_logic_VECTOR(7 downto 0);
b: IN std_logic_VECTOR(7 downto 0);
c: IN std_logic;
ce: IN std_logic;
   ci: IN std_logic;
   clr: IN std_logic;
s: OUT std_logic_VECTOR(8 downto 0));
end component;
```

**Figure 3-129: Core Customization Window for a Parameterized Core**
-- Synplicity black box declaration
attribute black_box : boolean;
attribute black_box of test : component is true;

-- COMP_TAG_END ------ End COMPONENT Declaration ----------

-- The following code must appear in the VHDL architecture body. Substitute your own instance name and net names.

------------------ Begin Cut here for INSTANTIATION Template ---- INST_TAG
your_instance_name : adder8
  port map (
    a => a,
    b => b,
    c => c,
    ce => ce,
    ci => ci,
    clr => clr,
    s => s);

------------------ INST_TAG_END ------ End INSTANTIATION Template -----------

-- You must compile the wrapper file test.vhd when simulating the core, test. When compiling the wrapper file, be sure to reference the XilinxCoreLib VHDL simulation library. For detailed instructions, please refer to the "Core Generator Guide".

Fixed Netlist Cores

The other type of Virtex-II Pro/Virtex-II Pro X core provided by the CORE Generator is the fixed netlist core. These are preset, non-parameterized designs that are shipped with the following:

- A fixed EDIF implementation netlist (as opposed to one that is customized on the fly)
- .VEO and .VHO templates
- Non-parameterized .V and .VHD behavioral simulation models
- Schematic symbol support

Examples include the fixed netlist Xilinx FFTs and most AllianceCORE products.

Since the HDL behavioral models for fixed netlist cores are not parameterized, the corresponding .VEO and .VHO template files are correspondingly simple. They do not need to pass customizing parameter values to a library behavioral model.

Xilinx IP Solutions and the IP Center

The CORE Generator works in conjunction with the Xilinx IP Center on the world wide web to provide the latest IP and software upgrades. To make the most of this resource, Xilinx highly recommends that whenever starting a design, first do a quick search of the Xilinx IP Center (www.xilinx.com/ipcenter) to see whether a ready-made core solution is already available.

A complete catalog of Xilinx cores and IP tools resides on the IP Center, including:

- LogiCORE Products
- AllianceCORE Products
- Reference Designs
• XPERTS Partner Consultants
• Design Reuse Tools

When installing the CORE Generator software, the designer gains immediate access to dozens of cores supplied by the LogiCORE Program. In addition, data sheets are available for all AllianceCORE products, and additional, separately licensed, advanced function LogiCORE products are also available. New and updated Virtex-II Pro / Virtex-II Pro X IP for the CORE Generator can be downloaded from the IP Center and added to the CORE Generator catalog.

LogiCORE Program

LogiCORE products are designed, sold, licensed, and supported by Xilinx. LogiCORE products include a wide selection of generic, parameterized functions, such as muxes, adders, multipliers, and memory cores which are bundled with the Xilinx CORE Generator software at no additional cost to licensed software customers. System-level cores, such as PCI, Reed-Solomon, ADPCM, HDLC, POS-PHY, and Color Space Converters are also available as optional, separately licensed products. Probably, the most common application of the CORE Generator is to use it to quickly generate Virtex-II Pro / Virtex-II Pro X block and distributed memories. A more detailed listing of available LogiCORE products is available in Table 3-63 and on the Xilinx IP Center website (www.xilinx.com/ipcenter).

Types of IP currently offered by the Xilinx LogiCORE program include:
• Basic Elements: logic gates, registers, multiplexers, adders, multipliers
• Communications and Networking: ADPCM modules, HDLC controllers, ATM building blocks, forward error correction modules, and POS-PHY Interfaces
• DSP and Video Image Processing: cores ranging from small building blocks (e.g., Time Skew Buffers) to larger system-level functions (e.g., FIR Filters and FFTs)
• System Logic: accumulators, adders, subtracters, complementers, multipliers, integrators, pipelined delay elements, single and dual-port distributed and block RAM, ROM, and synchronous and asynchronous FIFOs
• Standard Bus Interfaces: PCI 64/66 (64-bit, 66 MHz), 64/33 (64-bit, 33 MHz), and 32/33 (32-bit, 3 3MHz) Interfaces

AllianceCORE Program

The AllianceCORE program is a cooperative effort between Xilinx and third-party IP developers to provide additional system-level IP cores optimized for Xilinx FPGAs. To ensure a high level of quality, AllianceCORE products are implemented and verified in a Xilinx device as part of the certification process.

Xilinx develops relationships with AllianceCORE partners who can complement the Xilinx LogiCORE product offering. Where Xilinx does not offer a LogiCORE for a particular function, Xilinx partners with an AllianceCORE partner to offer that function. A large percentage of Xilinx AllianceCORE partners focus on data and telecommunication applications, as well as processor and processor peripheral designs.

Together, Xilinx and the AllianceCORE partners are able to provide an extensive library of cores to accelerate the design process. AllianceCORE products include customizable cores which can be configured to exact needs, as well as fixed netlist cores targeted toward specific applications. In many cases, partners can provide cores customized to meet the specific design needs if the primary offerings do not fit the requirements. Additionally,
source code versions of the cores are often available from the partners at additional cost for those who need maximum flexibility.

The library of Xilinx and AllianceCORE IP cores allows designers to leverage the expertise of experienced designers who are well-versed in optimizing designs for Virtex-II Pro/Virtex-II Pro X and other Xilinx architectures. This enables designers to obtain high performance and density in the target device with a faster time to market.

Reference Designs

Xilinx offers two types of reference designs; application notes (XAPPs) developed by Xilinx, and reference designs developed through the Xilinx Reference Design Alliance Program. Both types are extremely valuable to customers looking for guidance when designing systems. Reference designs can often be used as starting points for implementing a broad spectrum of functions in Xilinx programmable logic.

Application notes developed by Xilinx usually include supporting design files. They are supplied free of charge, without technical support or warranty. To see currently available reference designs, visit the Xilinx website.

Reference designs developed through the Xilinx Reference Design Alliance Program are developed, owned, and controlled by the partners in the program. The goal of the program is to form strategic engineering and marketing partnerships with other semiconductor manufacturers and design houses so as to assist in the development of high quality, multicomponent reference designs that incorporate Xilinx devices and demonstrate how they can operate at the system level with other specialized and general purpose semiconductors.

The reference designs in the Xilinx Reference Design Alliance Program are fully functional and applicable to a wide variety of digital electronic systems, including those used for networking, communications, video imaging, and DSP applications. Visit the Xilinx website to see a list of designs currently available through this program.

XPERTS Program

Xilinx established the XPERTS Program to provide customers with access to a worldwide network of certified design consultants proficient with Xilinx Platform FPGAs, software, and IP core integration. All XPERT members are certified and have extensive expertise and experience with Xilinx technology in various vertical applications, such as communications and networking, DSP, video and image processing, system I/O interfaces, and home networking.

XPERTS partners are an integral part of Xilinx strategy to provide customers with cost-efficient design solutions, while accelerating time to market. For more information on Xilinx XPERTS Program, visit the Xilinx website.

Design Reuse Tools

To facilitate the archiving and sharing of IP created by different individuals and workgroups within a company, Xilinx offers the IP Capture Tool. The IP Capture Tool helps to package design modules created by individual engineers in a standardized format so that they can be cataloged and distributed using the Xilinx CORE Generator. A core can take the form of synthesizable VHDL or Verilog code, or a fixed function netlist. Once it is packaged by the IP Capture Tool and installed into the CORE Generator, the “captured” core can be shared with other designers within a company through an internal network. The IP Capture Tool is supplied as a separate utility through the Xilinx IP Center. For more information, see the Xilinx website.
CORE Generator Summary

The CORE Generator delivers a complete catalog of IP including behavioral models, synthesis templates, and netlists with performance guaranteed by Xilinx Smart-IP technology. It is a repository for LogiCORE products from Xilinx, AllianceCORE products from Xilinx partners, and it supports Design Reuse for internally developed IP. In addition, LogiCORE products are continuously updated to add support for new Xilinx architectures, such as Virtex-II Pro / Virtex-II Pro X. The most current IP updates are available from the Xilinx IP Center.

Utilizing the CORE Generator library of parameterizable cores, designed by Xilinx for Xilinx FPGAs, the designer can enjoy the advantages of design reuse, including faster time to market and lower cost solutions. For more information, visit the Xilinx IP Center www.xilinx.com/ipcenter website.

Virtex-II Pro/Virtex-II Pro X IP Cores Support

Table 3-63 provides a partial listing of cores available for Virtex-II Pro / Virtex-II Pro X designs. For a complete catalog of IP, visit the Xilinx IP Center www.xilinx.com/ipcenter website.

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Software-Only IP:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Memory Tests utility</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Used to test Memory Interfaces</td>
<td></td>
</tr>
<tr>
<td>VxWorks Integration / RTOS</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Delivered with each peripheral with the Device Driver</td>
<td></td>
</tr>
<tr>
<td>Adaptation Layer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Board Support Package (BSP)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>BSP for the Xilinx development board including PPC405 boot code</td>
<td></td>
</tr>
<tr>
<td>Chip Support Package (CSP)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Configurable BSP – Delivered through Platform Generator</td>
<td></td>
</tr>
<tr>
<td>Bus Infrastructure:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>OPB Arbiter</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>PLB Arbiter</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>PLB-OPB Bridge</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>OPB-PLB Bridge</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>OPB IPIF Modules:</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPIF-Slave Attachment</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>IPIF-Master Attachment</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>IPIF-Address Decode</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>IPIF-Interrupt Control</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
</tbody>
</table>
Table 3-63: Virtex-II Pro/Virtex-II Pro X IP Cores Support (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPIF-Read Packet FIFOs</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>IPIF-Write Packet FIFOs</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>IPIF-DMA</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>IPIF-Scatter Gather</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
</tbody>
</table>

### Memory Interfaces:

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLB External Memory Controller</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>SRAM &amp; FLASH interface including Flash device driver</td>
<td></td>
</tr>
<tr>
<td>OPB External Memory Controller</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>SRAM &amp; FLASH interface including Flash device driver</td>
<td></td>
</tr>
</tbody>
</table>

### Memory Interfaces (continued):

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLB DDR Controller</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>DDR Memory Controller</td>
<td></td>
</tr>
<tr>
<td>OPB BRAM Controller</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>OPB BRAM Controller</td>
<td></td>
</tr>
<tr>
<td>OPB ZBT Controller</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>ZBT Memory Controller</td>
<td></td>
</tr>
</tbody>
</table>

### Peripherals:

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interrupt Controller</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>UART-16550</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>UART-16450</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>IIC Master &amp; Slave</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>SPI Master &amp; Slave</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>Ethernet 10/100 MAC</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>ATM Utopia Level 2 Slave</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>TimeBase/Watch Dog Timer</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>Timer/Counter</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>UART - Lite</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
<tr>
<td>GPIO</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Device driver included</td>
<td></td>
</tr>
</tbody>
</table>

### Basic Elements:

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUFE-based Multiplexer Slice</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256 bits wide</td>
<td></td>
</tr>
<tr>
<td>BUFT-based Multiplexer Slice</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256 bits wide</td>
<td></td>
</tr>
<tr>
<td>Binary Counter</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>2-256 bits output width</td>
<td></td>
</tr>
<tr>
<td>Binary Decoder</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>2-256 bits output width</td>
<td></td>
</tr>
<tr>
<td>Bit Bus Gate</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256 bits wide</td>
<td></td>
</tr>
<tr>
<td>Bit Gate</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256 bits wide</td>
<td></td>
</tr>
<tr>
<td>Bit Multiplexer</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256 bits wide</td>
<td></td>
</tr>
<tr>
<td>Bus Gate</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256 bits wide</td>
<td></td>
</tr>
</tbody>
</table>
### Table 3-63: Virtex-II Pro/Virtex-II Pro X IP Cores Support (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bus Multiplexer</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1O widths up to 256 bits</td>
<td></td>
</tr>
<tr>
<td>Comparator</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256 bits wide</td>
<td></td>
</tr>
<tr>
<td>FD-based Parallel Register</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256 bits wide</td>
<td></td>
</tr>
<tr>
<td>FD-based Shift Register</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-64 bits wide</td>
<td></td>
</tr>
<tr>
<td>LD-based Parallel Latch</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256 bits wide</td>
<td></td>
</tr>
<tr>
<td>RAM-based Shift Register</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256 bits wide, 1024 words deep</td>
<td></td>
</tr>
</tbody>
</table>

#### Communication & Networking:

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>3G FEC Package</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Viterbi Decoder, Turbo Codec, Convolutional Enc</td>
<td>3G Wireless Infrastructure</td>
</tr>
<tr>
<td>3GPP Compliant Turbo Convolutional Decoder</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>3GPP specs, 2 Mb/s, BER=10-6 for 1.5dB SNR</td>
<td>3G Wireless Infrastructure</td>
</tr>
<tr>
<td>3GPP Compliant Turbo Convolutional Encoder</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Compliant w/ 3GPP, puncturing</td>
<td>3G Wireless Infrastructure</td>
</tr>
<tr>
<td>3GPP Turbo Decoder</td>
<td>SysOn Chip</td>
<td>AllianceCORE</td>
<td>3GPP/UMTS compliant, IMT-2000, 2Mb/s data</td>
<td>Error correction, wireless</td>
</tr>
<tr>
<td>8b/10b Decoder</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Industry std 8b/10b en/decode for serial data transmission</td>
<td>Physical layer of Fiber Channel</td>
</tr>
<tr>
<td>8b/10b Encoder</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Industry std 8b/10b en/decode for serial data transmission</td>
<td>Physical layer of Fiber Channel</td>
</tr>
<tr>
<td>ADPCM 1024 Channel</td>
<td>Amphi on</td>
<td>AllianceCORE</td>
<td>G.721, 723, 726, 726a, 727, 727a, u-law, a-law</td>
<td>DECT, VOIP, cordless telephony</td>
</tr>
<tr>
<td>ADPCM 256 Channel</td>
<td>Amphi on</td>
<td>AllianceCORE</td>
<td>G.721, 723, 726, 726a, 727, 727a, u-law, a-law</td>
<td>DECT, VOIP, cordless telephony</td>
</tr>
<tr>
<td>ADPCM 512 Channel</td>
<td>Amphi on</td>
<td>AllianceCORE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ADPCM 768 Channel</td>
<td>Amphi on</td>
<td>AllianceCORE</td>
<td>G.721, 723, 726, 726a, 727, 727a, u-law, a-law</td>
<td>DECT, VOIP, cordless telephony</td>
</tr>
<tr>
<td>ADPCM Speech Codec, 32 Channel (DO-DI-ADPCM32)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>G.726, G.727, 32 duplex channels</td>
<td>DECT, VOIP, Wireless local loop, DSLAM, PBX</td>
</tr>
<tr>
<td>ADPCM Speech Codec, 64 Channel (DO-DI-ADPCM64)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>G.726, G.727, 64 duplex channels</td>
<td>DECT, VOIP, wireless local loop, DSLAM, PBX</td>
</tr>
<tr>
<td>BOOST LITE Bluetooth Baseband Processor</td>
<td>NewLogic</td>
<td>AllianceCORE</td>
<td>Compliant to Bluetooth v1.1, BQB qualified software for L2CAP, LHP, HCI, voice support</td>
<td>Bluetooth applications</td>
</tr>
<tr>
<td>BOOST Lite Bluetooth Baseband Processor</td>
<td>NewLogic</td>
<td>AllianceCORE</td>
<td>Compliant to Bluetooth v1.1, BQB qualified software for L2CAP, LHP, HCI, voice support</td>
<td>Bluetooth applications</td>
</tr>
</tbody>
</table>
### Table 3-63: Virtex-II Pro/Virtex-II Pro X IP Cores Support (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Convolutional Encoder</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>k from 3 to 9, puncturing from 2/3 to 12/13</td>
<td>3G base stations, broadcast, wireless LAN, cable modem, xDSL, satellite com, uwave</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DVB-RCS Turbo Decoder</td>
<td>iCODI NG</td>
<td>AllianceCORE</td>
<td>DVB-RCS compliant, 9 Mb/s data rate, switchable code rates and frame sizes</td>
<td>Error correction, wireless, DVB, Satellite data link</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flexbus 4 Interface Core, 16-Channel (DO-DI-FLX4C16)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td></td>
<td>Line card: terabit routers &amp; optical switches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flexbus 4 Interface Core, 4-Channel (DO-DI-FLX4C4)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td></td>
<td>Line card: terabit routers &amp; optical switches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flexbus 4 Interface Core, 1-Channel (DO-DI-FLX4C1)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td></td>
<td>Line card: terabit routers &amp; optical switches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HDLC Controller Core, 32 Channels</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>32 full duplex, CRC-16/32, 8/16-bit address insertion/deletion</td>
<td>X.25, POS, cable modems, frame relay switches, video conferencing over ISDN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HDLC Controller Core, Single Channel</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>16/32-bit frame seq, 8/16-bit addr insert/delete, flag/zerop insert/detect</td>
<td>X.25, POS, cable modems, frame relay switches, video conf. over ISDN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interleaver/De-interleaver</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Convolutional, width up to 256 bits, 256 branches</td>
<td>Broadcast, wireless LAN, cable modem, xDSL, satellite com, uwave nets, digital TV</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PE-MACMII Dual Speed 10/100 Mb/s Ethernet MAC</td>
<td>Alcatel</td>
<td>AllianceCORE</td>
<td>802.3 compliant, Supports single &amp; multimode fiber optic devices, MII interfaces, RMON and Etherstate statistics</td>
<td>Networking, Broadband, NIC, SOHO, Home networking, storage, routers, switches, printers</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS-PHY Level 3 Link Layer Interface Core, 48 Channel (DO-DI-POSL3LINK48A)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td></td>
<td>Line card: terabit routers &amp; optical switches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS-PHY L3 Link Layer Interface, 16-Ch (DO-DI-POSL3LINK16)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td></td>
<td>Line card: terabit routers &amp; optical switches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS-PHY L3 Link Layer Interface, 4-Ch (DO-DI-POSL3LINK4)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td></td>
<td>Line card: terabit routers &amp; optical switches</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>POS-PHY L3 Link Layer Interface, 2-Ch (DO-DI-POSL3LINK2)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td></td>
<td>Line card: terabit routers &amp; optical switches</td>
</tr>
</tbody>
</table>
### Table 3-63: Virtex-II Pro/Virtex-II Pro X IP Cores Support (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>POS-PHY L3 Link Layer Interface, Single Channel</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Std or custom coding, 3-12 bit symbol width, up to 4095 symbols</td>
<td>Broadcast, wireless LAN, digital TV, cable modem, xDSL, satellite com, uwave nets</td>
</tr>
<tr>
<td>POS-PHY L4 Multi-Channel Interface (DO-DI-POS4MC)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>parameterizable, RTL available</td>
<td>Error correction, wireless, DSL</td>
</tr>
<tr>
<td>Reed-Solomon Decoder</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Std or custom coding, 3-12 bit width, up to 4095 symbols with 256 check symb.</td>
<td>Broadcast, wireless LAN, digital TV, cable modem, xDSL, satellite com, uwave nets</td>
</tr>
<tr>
<td>Reed-Solomon Decoder</td>
<td>TILAB</td>
<td>AllianceCORE</td>
<td>parameterizable, RTL available</td>
<td>Error correction, wireless, DSL</td>
</tr>
<tr>
<td>SDLC Controller</td>
<td>CAST</td>
<td>AllianceCORE</td>
<td>Like Intel 8XC152 Global Serial Channel, Serial Comm., HDLC apps, telecom</td>
<td>Embedded systems, professional audio, video</td>
</tr>
<tr>
<td>SPEEDROUTER Network Processor</td>
<td>IP</td>
<td>AllianceCORE</td>
<td>Solution requires SPEEDAnalyzer ASIC, 2.5 Gb/s fdx wire speed; net processor (NPV)</td>
<td>Networking, edge and access, Switches and routers</td>
</tr>
<tr>
<td>Turbo Decoder - 3GPP</td>
<td>SysOn Chip</td>
<td>AllianceCORE</td>
<td>3GPP/UMTS compliant, 2 Mb/s data rate</td>
<td>Error correction, wireless</td>
</tr>
<tr>
<td>Turbo Encoder</td>
<td>TILAB</td>
<td>AllianceCORE</td>
<td>3GPP/UMTS compliant, up to 4 interleaver laws</td>
<td>Error correction, wireless</td>
</tr>
<tr>
<td>TURBO_DEC Turbo Decoder</td>
<td>TILAB</td>
<td>AllianceCORE</td>
<td>3GPP/UMTS compliant, &gt;2 Mb/s data rate</td>
<td>Error correction, wireless</td>
</tr>
<tr>
<td>Viterbi Decoder</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Puncturing, serial &amp; parallel architecture,</td>
<td>3G base stations, broadcast, wireless LAN, cable modem, xDSL, satellite com, uwave</td>
</tr>
<tr>
<td>Viterbi Decoder, IEEE 802-compatible</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Constraint length(k)=7, G0=171, G1=133</td>
<td>L/MMDS, cable modem, broadcast equip, wireless LAN, xDSL, sat com, uwave nets</td>
</tr>
</tbody>
</table>
Table 3-63: Virtex-II Pro/Virtex-II Pro X IP Cores Support (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Digital Signal Processing:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024-Point Complex FFT IFFT for Virtex-II</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>16 bit complex data, 2’s comp, forward and inverse transform</td>
<td></td>
</tr>
<tr>
<td>16-Point Complex FFT IFFT for Virtex-II</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>16 bit complex data, 2’s comp, forward and inverse transform</td>
<td></td>
</tr>
<tr>
<td>256-Point Complex FFT IFFT for Virtex-II</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>16 bit complex data, 2’s comp, forward and inverse transform</td>
<td></td>
</tr>
<tr>
<td>32 Point Complex FFT/IFFT</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>64-Point Complex FFT IFFT for Virtex-II</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>16 bit complex data, 2’s comp, forward and inverse transform</td>
<td></td>
</tr>
<tr>
<td>Bit Correlator</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>4096 taps, serial/parallel input, 4096 bits width</td>
<td></td>
</tr>
<tr>
<td>Cascaded Integrator Comb (CIC)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>32 bits data width, rate change from 8 to 16384</td>
<td></td>
</tr>
<tr>
<td>Direct Digital Synthesizer</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>8-65K samples, 32-bits output precision, phase dithering/offset</td>
<td></td>
</tr>
<tr>
<td>Distributed Arithmetic FIR Filter</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>32-bit input/coeff width, 1024 taps, 1-8 chan, polyphase, online coeff reload</td>
<td></td>
</tr>
<tr>
<td>GVA-300 Virtex-II DSP Hardware Accelerator</td>
<td>GV</td>
<td>AllianceCORE</td>
<td>2 Virtex-II, Spartan-II FPGAs, 1 CPLD, Matlab I/F</td>
<td>DSP prototyping</td>
</tr>
<tr>
<td>LFSR, Linear Feedback Shift Register</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>168 input widths, SRL16/register implementation</td>
<td></td>
</tr>
<tr>
<td><strong>Math Functions:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Accumulator</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256s bit wide</td>
<td></td>
</tr>
<tr>
<td>Adder Subtractor</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256s bit wide</td>
<td></td>
</tr>
<tr>
<td>DFP2INT Floating Point to Integer Converter</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>Full IEEE-754 compliance, 4 pipelines, Single precision real format support</td>
<td>DSP, Math, Arithmetic apps</td>
</tr>
<tr>
<td>DFPADD Floating Point Adder</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>Full IEEE-754 compliance, 4 pipelines, Single precision real format support</td>
<td>DSP, Math, Arithmetic apps</td>
</tr>
<tr>
<td>DFPCOMP Floating Point Comparator</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>Full IEEE-754 compliance, 4 pipelines, Single precision real format support</td>
<td>DSP, Math, Arithmetic apps</td>
</tr>
<tr>
<td>DFPDIV Floating Point Divider</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>Full IEEE-754 compliance, 15 pipelines, Single precision real format support</td>
<td>DSP, Math, Arithmetic apps</td>
</tr>
</tbody>
</table>
### Table 3-63: Virtex-II Pro/Virtex-II Pro X IP Cores Support (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>DFPmul Floating Point Multiplier</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>Full IEEE-754 compliance, 7 pipelines, 32x32 mult, Single precision real format support</td>
<td>DSP, Math, Arithmetic apps</td>
</tr>
<tr>
<td>DFPSQRT Floating Point Square Root</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>Full IEEE-754 compliance, 4 pipelines, Single precision real format support</td>
<td>DSP, Math, Arithmetic apps</td>
</tr>
<tr>
<td>DINT2FP Integer to Floating Point Converter</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>Full IEEE-754 compliance, double word input, 2 pipelines, Single precision real output</td>
<td>DSP, Math, Arithmetic apps</td>
</tr>
<tr>
<td>Multiply Accumulator (MAC)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Input width up to 32 bits, 65-bit accumulator, truncation rounding</td>
<td></td>
</tr>
<tr>
<td>Multiply Generator</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>64-bit input data width, constant, reloadable or variable inputs, parallel/sequential implementation</td>
<td></td>
</tr>
<tr>
<td>Pipelined Divider</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>32-bit input data width, multiple clock per output</td>
<td></td>
</tr>
<tr>
<td>Sine Cosine Look Up Table</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>3-10 bit in, 4-32 bit out, distributed/block ROM</td>
<td></td>
</tr>
<tr>
<td>Twos Complementer</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Input width up to 256 bits</td>
<td></td>
</tr>
</tbody>
</table>

**Memories & Storage Elements:**

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Asynchronous FIFO</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256 bits, 15-65535 words, DRAM or BRAM, independent 1/O clock domains</td>
<td></td>
</tr>
<tr>
<td>Content Addressable Memory (CAM)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-512 bits, 2-10K words, SRL16</td>
<td></td>
</tr>
<tr>
<td>Distributed Memory</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-1024 bit, 16-65536 word, RAM/ROM/SRL16, opt output regs and pipelining</td>
<td></td>
</tr>
<tr>
<td>Dual-Port Block Memory</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256 bits, 2-13K words</td>
<td></td>
</tr>
<tr>
<td>Single-Port Block Memory</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256 bits, 2-128K words</td>
<td></td>
</tr>
<tr>
<td>Synchronous FIFO</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>1-256 bits, 16-256 words, distributed/block RAM</td>
<td></td>
</tr>
</tbody>
</table>
### Table 3-63: Virtex-II Pro/Virtex-II Pro X IP Cores Support (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microprocessors, Controllers &amp; Peripherals:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10/100 Ethernet MAC</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Interfaces through OPB to MicroBlaze™</td>
<td>Networking, comm., processor applications</td>
</tr>
<tr>
<td>AX1610 16-bit RISC Processor</td>
<td>Loarant</td>
<td>AllianceCORE</td>
<td>44 opcode, 64-K word data, program, Harvard arch.</td>
<td>Control functions, State mach, Coprocessor</td>
</tr>
<tr>
<td>C165X MicroController</td>
<td>CAST</td>
<td>AllianceCORE</td>
<td>Microchip 16C5X PIC like</td>
<td>Embedded systems, telecom</td>
</tr>
<tr>
<td>C68000 Microprocessor</td>
<td>CAST</td>
<td>AllianceCORE</td>
<td>MC68000 Compatible</td>
<td>Embedded systems, pro audio, video</td>
</tr>
<tr>
<td>CPU FPGA (Virtex-II) MicroEngine Cards</td>
<td>NMI</td>
<td>AllianceCORE</td>
<td>Hitachi SH-3 CPU</td>
<td>Embedded systems</td>
</tr>
<tr>
<td>CZ80CPU Microprocessor</td>
<td>CAST</td>
<td>AllianceCORE</td>
<td>Zilog Z80 compatible, 8-bit processor</td>
<td>Embedded systems, Communications</td>
</tr>
<tr>
<td>DDR SDRAM Controller Core</td>
<td>Memec-Core</td>
<td>AllianceCORE</td>
<td>DDR SDRAM burst length support for 2,4,8 per access, supports data 16,32, 64, 72.</td>
<td>Digital video, embedded computing, networking</td>
</tr>
<tr>
<td>DFPIC125X Fast RISC MicroController</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>PIC 12c4x like, 2X faster, 12-bit wide instruction set, 33 instructions</td>
<td>Embedded systems, telecom, audio and video</td>
</tr>
<tr>
<td>DFPIC1655X Fast RISC MicroController</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>S/W compatible with PIC16C55X, 14-bit instruction set, 35 instructions</td>
<td>Embedded systems, telecom, audio and video</td>
</tr>
<tr>
<td>DFPIC165X Fast RISC MicroController</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>PIC 12c4x like, 2X faster, 12-bit wide instruction set, 33 instructions</td>
<td>Embedded systems, telecom, audio and video</td>
</tr>
<tr>
<td>D12CM I2C Bus Controller Master</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>I2C-like, multi master, fast/std. modes</td>
<td>Embedded systems</td>
</tr>
<tr>
<td>D12CM I2C Bus Controller Slave</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>I2C-like, Slave</td>
<td>Embedded</td>
</tr>
<tr>
<td>D12CSB I2C Bus Controller Slave Base</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>I2C-like, Slave</td>
<td>Embedded Systems</td>
</tr>
<tr>
<td>DR8051 RISC MicroController</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>80C31 instruction set, RISC architecture 6.7X faster than standard 8051</td>
<td>Embedded systems, telecom, video</td>
</tr>
<tr>
<td>DR8051BASE RISC MicroController</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>80C31 instruction set, high speed multiplier, RISC architecture 6.7X faster than standard 8051</td>
<td>Embedded systems, telecom, video</td>
</tr>
<tr>
<td>DR8052EX RISC MicroController</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>80C31 instruction set, high speed multi/div .RISC 6.7X faster than standard 8051</td>
<td>Embedded systems, telecom, video</td>
</tr>
</tbody>
</table>
Table 3-63: Virtex-II Pro/Virtex-II Pro X IP Cores Support (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>e8254 Programmable Interval Timer/Counter</td>
<td>einfochips</td>
<td>AllianceCORE</td>
<td>Three 8-bit parallel ports, 24 programmable IO lines, 8-bit bidi data bus</td>
<td>Processor, I/O interface</td>
</tr>
<tr>
<td>e8255 Peripheral Interface</td>
<td>einfochips</td>
<td>AllianceCORE</td>
<td>Three 8-bit parallel ports, 24 programmable IO lines, 8-bit bidi data bus</td>
<td>Processor, I/O interface</td>
</tr>
<tr>
<td>Flip805x-PS Microprocessor</td>
<td>Dolphin</td>
<td>AllianceCORE</td>
<td>Avg 8X faster &amp; code compatible v. legacy 8051, verification bus monitor,</td>
<td>DSP, Telecom, industrial, high speed</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>SFR IF, DSP focused</td>
<td>control</td>
</tr>
<tr>
<td>IIC Master and Slave</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Interfaces through OPB to MicroBlaze™</td>
<td>Networking, com, processor applic</td>
</tr>
<tr>
<td>Lavacore Configurable Java Processor Core</td>
<td>derivation</td>
<td>AllianceCORE</td>
<td>32b data/address optional DES</td>
<td>Internet appliance, industrial control</td>
</tr>
<tr>
<td>Lavacore Configurable Java Processor Core</td>
<td>derivation</td>
<td>AllianceCORE</td>
<td>32b data/address optional DES</td>
<td>Internet appliance, industrial control</td>
</tr>
<tr>
<td>Lightfoot 32-bit Java Processor Core</td>
<td>Digital</td>
<td>AllianceCORE</td>
<td>32bit data, 24 bit address, 3 Stage pipeline, Java/C dev. tools</td>
<td>Internet appliance, industrial control, HAVi multimedia, set top boxes</td>
</tr>
<tr>
<td>Microblaze™ Soft RISC Processor</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Soft RISC Processor, small footprint</td>
<td>Networking, communications</td>
</tr>
<tr>
<td>OPB Arbiter</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Bundled in the MicroBlaze Development Kit</td>
<td>Processor applications</td>
</tr>
<tr>
<td>OPB GPIO</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Bundled in the MicroBlaze Development Kit</td>
<td>Processor applications</td>
</tr>
<tr>
<td>OPB Interrupt Controller</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Bundled in the MicroBlaze Development Kit</td>
<td>Processor applications</td>
</tr>
<tr>
<td>OPB Memory Interface (Flash, SRAM)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Bundled in the MicroBlaze Development Kit</td>
<td>Processor applications</td>
</tr>
<tr>
<td>OPB Timer/Counter</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Bundled in the MicroBlaze Development Kit</td>
<td>Processor applications</td>
</tr>
<tr>
<td>OPB UART (16450, 16550)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Interfaces through OPB to MicroBlaze</td>
<td>Processor applications</td>
</tr>
<tr>
<td>OPB UART Lite</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Bundled in the MicroBlaze Development Kit</td>
<td>Processor applications</td>
</tr>
<tr>
<td>OPB WDT</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Bundled in the MicroBlaze Development Kit</td>
<td>Processor applications</td>
</tr>
<tr>
<td>PF3100 PC/104-Plus Reconfigurable Module</td>
<td>derivation</td>
<td>AllianceCORE</td>
<td>PC/104 &amp; PC/104+ development board</td>
<td>Internet appliance, industrial control</td>
</tr>
<tr>
<td>SPI</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Interfaces through OPB to MicroBlaze</td>
<td>Networking, communications, processor</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>applications</td>
<td>applications</td>
</tr>
<tr>
<td>XF-UART Asynchronous Communications Core</td>
<td>Memecore</td>
<td>AllianceCORE</td>
<td>UART and baud rate generator</td>
<td>Serial data communication</td>
</tr>
</tbody>
</table>
Table 3-63: Virtex-II Pro/Virtex-II Pro X IP Cores Support (Continued)

<table>
<thead>
<tr>
<th>Function</th>
<th>Vendor Name</th>
<th>IP Type</th>
<th>Key Features</th>
<th>Application Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard Bus Interfaces:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>PCI32 Virtex Interface Design Kit (DO-DI-PCI32-DKT)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>Includes PCI32 board, drive development kit, and customer education 3-day training class</td>
<td>PC add-in boards, CPCI, Embedded</td>
</tr>
<tr>
<td>PCI32 Virtex Interface, IP Only (DO-DI-PCI32-IP)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>v2.2 comp, assured PCI timing, 3.3/5-V, 0-waitstate, CPCI hot swap friendly</td>
<td>PC boards, CPCI, Embedded, hiperf video, gb ethernet</td>
</tr>
<tr>
<td>PCI64 &amp; PCI32, IP Only (DO-DI-PCI-AL)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>v2.2 comp, assured PCI timing, 3.3/5-V, 0-waitstate, CPCI hot swap friendly</td>
<td>PC boards, CPCI, Embedded, hiperf video, gb ethernet</td>
</tr>
<tr>
<td>PCI64 Virtex Interface Design Kit (DO-DI-PCI64-DKT)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>v2.2 comp, assured PCI timing, 3.3/5-V, 0-waitstate, CPCI hot swap friendly</td>
<td>PC boards, CPCI, Embedded, hiperf video, gb ethernet</td>
</tr>
<tr>
<td>PCI64 Virtex Interface, IP Only (DO-DI-PCI64-IP)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>v2.2 comp, assured PCI timing, 3.3/5-V, 0-waitstate, CPCI hot swap friendly</td>
<td>PC boards, CPCI, Embedded, hiperf video, gb ethernet</td>
</tr>
<tr>
<td>RapidIO 8-bit port LP-LVDS Phy Layer (DO-DI-RIO8-PHY)</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>RapidIO Interconnect v1.1 compliant, verified with Motorola’s RapidIO bus functional model v1.4</td>
<td>Routers, switches, backplane, control plane, data path, embedded sys, high speed interface to memory and encryption engines, high end video</td>
</tr>
<tr>
<td><strong>Video &amp; Image Processing:</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>USB 1.1 Device Controller</td>
<td>Memec-Core</td>
<td>AllianceCORE</td>
<td>Compliant with USB1.1 spec., Supports VCI bus, Performs CRC, Supports 1.5 Mb/s &amp; 12 Mb/s</td>
<td>Scanners, Printers, Handhelds, Mass Storage</td>
</tr>
<tr>
<td>1-D Discrete Cosine Transform</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td>8-24 bits for coeff &amp; input, 8-64 pts</td>
<td>image, video phone, color laser printers</td>
</tr>
<tr>
<td>2-D DCT/IDCT Forward/Inverse Discrete Cosine Transform</td>
<td>Xilinx</td>
<td>LogiCORE</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FASTJPEG_BW Decoder</td>
<td>BARCO-SILEX</td>
<td>AllianceCORE</td>
<td>Conforms to ISO/IEC Baseline 10918-1, Gray-Scale</td>
<td>Video editing, digital camera, scanners</td>
</tr>
<tr>
<td>FASTJPEG_C Decoder</td>
<td>BARCO-SILEX</td>
<td>AllianceCORE</td>
<td>Conforms to ISO/IEC Baseline 10918-1, color, multi-scan, Gray-Scale</td>
<td>Video editing, digital camera, scanners</td>
</tr>
</tbody>
</table>
Summary

This chapter covers the following topics:

- Introduction
- Configuration Solutions
- Software Support and Data Files
- Serial Programming Modes
- SelectMAP Programming Modes
- JTAG / Boundary-Scan Programming Modes
- Configuration Details
- Readback

Introduction

Virtex-II Pro devices are configured by loading application-specific configuration data into internal memory. Configuration is carried out using a subset of the device pins, some of which are dedicated, while others can be reused as general-purpose inputs and outputs after configuration is complete.

Depending on the system design, several configuration modes are selectable via mode pins. The mode pins M2, M1, and M0 are dedicated pins. An additional pin, HSWAP_EN, is used in conjunction with the mode pins to select whether user I/O pins have pull-up resistors during configuration. By default, HSWAP_EN is tied High (internal pull-up resistor), which shuts off pull-up resistors on the user I/O pins during configuration. When HSWAP_EN is tied Low, the pull-up resistors are on and therefore, the user I/Os have pull-up resistors during configuration.

Other dedicated pins are:

- CCLK - the configuration clock pin
- DONE - configuration status pin
- TDI, TDO, TMS, TCK - boundary-scan pins
- PROG_B - configuration reset pin

Depending on the configuration mode selected, CCLK can be an output generated by the Virtex-II Pro FPGA or an input accepting externally generated clock data. For correct operation, these pins require a \( V_{CC AUX} \) of 2.5V to permit LVCMOS operations.
All dual-function configuration pins are contained in banks 4 and 5. Bank 4 contains pins used in serial configuration modes, banks 4 and 5 contain pins used for SelectMAP modes.

A persist option is available, which can be used to force pins to retain their configuration function even after device configuration is complete. If the persist option is not selected, then the configuration pins with the exception of CCLK, PROG_B, and DONE can be used for user I/O in normal operation. The persist option does not apply to boundary-scan related pins. The persist feature is valuable in applications that employ partial reconfiguration, dynamic reconfiguration, or readback.

The M2, M1, and M0 mode pins are intended to be set at a constant DC voltage level, either through pull-up or pull-down resistors, or tied directly to ground or $V_{CCAUX}$. The mode pins are not to be toggled (changed) during and after configuration.

### Configuration Modes

Virtex-II Pro supports the following configuration modes:

- Master-Serial
- Slave-Serial (default)
- Master SelectMAP
- Slave SelectMAP
- Boundary-Scan (IEEE 1532 and IEEE 1149)

Table 4-1 shows Virtex-II Pro configuration mode pin settings.

<table>
<thead>
<tr>
<th>Configuration Mode</th>
<th>M2</th>
<th>M1</th>
<th>M0</th>
<th>CCLK Direction</th>
<th>Data Width</th>
<th>Serial Dout</th>
</tr>
</thead>
<tbody>
<tr>
<td>Master Serial</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Out</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>Slave Serial</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>In</td>
<td>1</td>
<td>Yes</td>
</tr>
<tr>
<td>Master SelectMAP</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>Out</td>
<td>8</td>
<td>No</td>
</tr>
<tr>
<td>Slave SelectMAP</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>In</td>
<td>8</td>
<td>No</td>
</tr>
<tr>
<td>Boundary Scan</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>N/A</td>
<td>1</td>
<td>No</td>
</tr>
</tbody>
</table>

**Notes:**
1. The HSWAP_EN pin controls the pull-ups. Setting M2, M1, and M0 selects the configuration mode, while the HSWAP_EN pin controls whether or not the pull-ups are used.
2. Daisy chaining is possible only in modes where Serial Dout is used. For example, in SelectMAP modes, the first device does NOT support daisy chaining of downstream devices.

Table 4-2 lists the total number of bits required to configure each device.

<table>
<thead>
<tr>
<th>Device</th>
<th>Default Bitstream Size</th>
<th>Device</th>
<th>Default Bitstream Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC2VP2</td>
<td>1,305,376</td>
<td>XC2VP40</td>
<td>15,868,192</td>
</tr>
<tr>
<td>XC2VP4</td>
<td>3,006,496</td>
<td>XC2VP50</td>
<td>19,021,344</td>
</tr>
<tr>
<td>XC2VP7</td>
<td>4,485,408</td>
<td>XC2VP70</td>
<td>26,098,976</td>
</tr>
<tr>
<td>XC2VP20</td>
<td>8,214,560</td>
<td>XC2VPX70</td>
<td>26,098,976</td>
</tr>
<tr>
<td>XC2VPX20</td>
<td>8,214,560</td>
<td>XC2VP100</td>
<td>34,292,768</td>
</tr>
<tr>
<td>XC2VP30</td>
<td>11,589,920</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Configuration Process and Flow

The configuration process involves loading the configuration bitstream into the FPGA using the selected mode. There are four major phases in the configuration process:

- Clearing Configuration Memory
- Initialization
- Loading Configuration Data
- Device Startup

Figure 4-1 illustrates the configuration process flow.

*Note: If power-up is at room temperature, $V_{CCO-4} \geq 1.5V$ is acceptable.
Power Up

The $V_{CCINT}$ power pins must be supplied with a 1.5V source. (Refer to the Virtex-II Pro and Virtex-II Pro X Platform FPGA Data Sheet for DC characteristics.) The IOB voltage input for Bank 4 ($V_{CCO_4}$) and the auxiliary voltage input ($V_{CCAUX}$) are also used as a logic input to the Power-On-Reset (POR) circuitry. Even if this bank is not being used, $V_{CCO_4}$ must be connected to a 1.5V or greater source.

Clearing Configuration Memory

In the memory clear phase, non-configuration I/O pins are 3-stated with optional pull-up resistors. The INIT_B and DONE pins are driven Low by the FPGA, and the memory is cleared. After PROG_B transitions High, memory is cleared twice and initialization can begin.

The INIT_B pin transitions High when the clearing of configuration memory is complete. A logic Low on the PROG_B input resets the configuration logic and holds the FPGA in the clear configuration memory state. When PROG_B is released, the FPGA continues to hold INIT_B Low until it has completed clearing all of the configuration memory. The minimum Low pulse time for PROG_B is defined by the $T_{PROGRAM}$ timing parameter. There is no maximum value. The power-up timing of configuration signals is shown in Figure 4-2. Corresponding power-up timing characteristics are detailed in Module 3 of the Data Sheet.

![Figure 4-2: Power-Up Timing Configuration Signals](image)

Initialization

For the initialization phase, the INIT_B pin is released, the mode pins are sampled, the appropriate pins become active, and the configuration process begins. It is possible to delay configuration by externally holding INIT_B Low.

Delaying Configuration

The INIT_B pin can also be held Low externally to delay configuration of the FPGA. The FPGA samples its mode pins on the rising edge of INIT_B. After INIT_B transitions to High, configuration can begin. No additional time-out or waiting periods are required, but configuration does not need to commence immediately after the transition of INIT_B. The configuration logic does not begin processing data until the synchronization word from the bitstream is loaded.
Loading Configuration Data

Cyclic Redundancy Checking (CRC) is automatically performed ("AutoCRC" — see page 343) at the end of any write to the Frame Data Input Register (FDRI), and again prior to startup. If the CRC checks do not fail, the device start-up phase can begin.

If the CRC check fails, the INIT_B pin is asserted (logic Low) to indicate that a CRC error has occurred. The startup sequence is aborted and configuration fails with DONE=0 and INIT=0.

To reconfigure the device, the PROG_B pin should be asserted to reset the configuration logic. Cycling power on the device also resets the FPGA for configuration. For more information on CRC calculation, see “CRC Register (CRC),” page 343.

The details of loading configuration data in each of the five modes are discussed in the following sections:

• “Serial Programming Modes,” page 302
• “SelectMAP Programming Modes,” page 306
• “JTAG / Boundary-Scan Programming Modes,” page 319

After the last data frame is loaded, the global set/reset (GSR) signal is pulsed (see the "GRESTORE" description in Table 4-25, page 345). The GSR signal initializes all registers in the device prior to entering the startup sequence, which is the next configuration phase.

Device Startup

Device startup brings the FPGA out of the configuration process and into normal device operation. The Startup Sequence is controlled by an eight-phase state machine that:

• Deasserts GTS, activating I/Os
• Asserts GWE, allowing RAMs and flip-flops to toggle
• Releases the DONE pin
• Asserts the End-Of-Startup signal (EOS) during the final phase. The EOS signal is an internal flag that can be read through the Status Register (see “Status Register (STAT),” page 347).

Figure 4-3 depicts the default start-up sequence.

The order of the start-up events is user-programmable according to the BitGen startup options (see Appendix A, “BitGen and PROMGen Switches and Options”). Delays can be inserted in the start-up sequence to allow DCMs to lock and DCI to match.

The startup sequencer always waits for the DONE pin on the FPGA to go High during the Done cycle. The DONE pin does not necessarily go High when it is "released" by the device, since the pin may be held Low externally by other devices. The DONE pin can be configured as an open-drain output, in which case the DONE pin is "released" to a high-impedance state. Alternatively, it can be configured as an active driver, in which case the DONE pin is "released" to an active High level. The BitGen DRIVE DONE (bitgen -g drivedone:yes) setting determines whether the active driver is enabled on the DONE pin. This feature allows several FPGAs with connected DONE pins become active simultaneously.
The DONE pin may be held low for many CCLK cycles after being released by the device, but when the transition from logic 0 to logic 1 begins, it must finish within one CCLK cycle. If many devices are connected to the same DONE signal, the last device may not be able to drive DONE high within one CCLK cycle. In this case an external pull-up resistor may be needed on the DONE signal. If the risetime on the DONE signal cannot be reduced to one CCLK period, set the DONE PIPE option (bitgen -g donepipe:yes) to allow an additional CCLK cycle.

**Configuration Pins**

Dedicated configuration pins (CCLK, PROG_B, DONE, M2, M1, and M0) are powered by \( V_{CCAUX} \) at 2.5V. They should not be exposed to 3.3V signals. However, in cases where they must be connected to 3.3V, it is highly recommended that they connect through 100\( \Omega \) series resistors to 3.3V, or that no more than 3 mA of current be allowed to pass through these pins at any time.

Dual-function pins (D0/DIN, D1:D7, CS_B, RDWR_B, BUSY/DOUT and INIT_B) powered by \( V_{CCO} \) are designated for configuration and other user functions after configuration. Table 4-3 is the list of all dedicated, dual-function, and user pins.

Before and during configuration, all configuration I/O pins are set for the LVCMOS25 (12mA, fast slew rate) standard when \( V_{CCO} \) is 2.5V. It becomes LVCMOS33 (16mA, fast slew rate) when \( V_{CCO} \) is 3.3V.

The configuration clock pin CCLK is set for the LVCMOS25 (12mA, slow slew rate).
### Table 4-3: Virtex-II Pro Pins and Voltages

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Direction</th>
<th>Supply Voltage (V)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCLK</td>
<td>Input/Output</td>
<td>2.5</td>
<td>Pin is supplied by VCCAUX. Input should not be driven by a 3.3V signal unless a 100Ω series resistor is used. Output is LVCMOS25 and compatible with LVTTL.</td>
</tr>
<tr>
<td>PROG_B</td>
<td>Input</td>
<td>2.5</td>
<td>Pin is supplied by VCCAUX (must be 2.5V). It should not be pulled up to 3.3V unless a 100Ω series resistor is used.</td>
</tr>
<tr>
<td>DONE</td>
<td>Open-drain</td>
<td>2.5</td>
<td>Pin is supplied by VCCAUX (must be 2.5V). It should not be pulled up to 3.3V unless a 470Ω series resistor is used.</td>
</tr>
<tr>
<td>M2, M1, M0</td>
<td>Input</td>
<td>2.5</td>
<td>Pins are supplied by VCCAUX (must be 2.5V). They should not be pulled up to 3.3V unless 100Ω series resistors are used.</td>
</tr>
<tr>
<td>HSWAP_EN</td>
<td>Input</td>
<td>2.5</td>
<td>Pin is supplied by VCCAUX (must be 2.5V). It should not be pulled up to 3.3V unless a 100Ω series resistor is used.</td>
</tr>
<tr>
<td>TDI</td>
<td>Input</td>
<td>2.5/3.3</td>
<td>Pin is supplied by VCCAUX (3.3V tolerant).</td>
</tr>
<tr>
<td>TMS</td>
<td>Input</td>
<td>2.5/3.3</td>
<td>Pin is supplied by VCCAUX (3.3V tolerant).</td>
</tr>
<tr>
<td>TCK</td>
<td>Input</td>
<td>2.5/3.3</td>
<td>Pin is supplied by VCCAUX (3.3V tolerant).</td>
</tr>
<tr>
<td>TDO</td>
<td>Open-drain</td>
<td>2.5/3.3</td>
<td>Pin is open-drain and can be pulled up to 3.3V. The external pull-up resistor should be stronger than 200Ω. There is no internal pull-up.</td>
</tr>
<tr>
<td>PWRDWN_B</td>
<td>Input</td>
<td>2.5</td>
<td>Pin is supplied by VCCAUX (2.5V). It has an internal pull-up. Do not pull up externally to 3.3V unless a 100Ω series resistor is used.</td>
</tr>
<tr>
<td>D0/DIN, D1-D7</td>
<td>Input/Output</td>
<td>2.5/3.3</td>
<td>Pins are supplied by VCCO.</td>
</tr>
<tr>
<td>CS_B</td>
<td>Input</td>
<td>2.5/3.3</td>
<td>Pin is supplied by VCCO.</td>
</tr>
<tr>
<td>RDWR_B</td>
<td>Input</td>
<td>2.5/3.3</td>
<td>Pin is supplied by VCCO.</td>
</tr>
<tr>
<td>BUSY/DOUT</td>
<td>Output</td>
<td>2.5/3.3</td>
<td>Pin is supplied by VCCO.</td>
</tr>
<tr>
<td>INIT_B</td>
<td>Input/Output</td>
<td>2.5/3.3</td>
<td>Pin is supplied by VCCO.</td>
</tr>
<tr>
<td>VRP</td>
<td>Input</td>
<td>N/A</td>
<td>Reference resistor pulled down to GND.</td>
</tr>
<tr>
<td>VRRN</td>
<td>Input</td>
<td>2.5/3.3</td>
<td>Reference resistor pulled up to VCCO.</td>
</tr>
<tr>
<td>VREF</td>
<td>Input</td>
<td>2.5/3.3</td>
<td>Depends on the I/O standard used.</td>
</tr>
<tr>
<td>DXN, DXP</td>
<td>N/A</td>
<td>N/A</td>
<td>These pins are the cathode and anode of the temperature diode.</td>
</tr>
<tr>
<td>VBATT</td>
<td>Input</td>
<td>3.0</td>
<td>Battery supply for the encryption keys. Connect this pin to either VCCAUX or GND when bitstream encryption is not used.</td>
</tr>
<tr>
<td>User I/O</td>
<td>Input/Output</td>
<td>Various</td>
<td>All regular user I/Os support the following 3.3V signaling standards: LVTTL, LVCMOS, PCI33, PCI66, PCIX, LVDCI33</td>
</tr>
</tbody>
</table>
Mixed Voltage Environments

Virtex-II Pro devices have multiple power inputs.

- $V_{CCINT}$ (1.5V) powers the internal circuitry.
- $V_{CC AUX}$ (2.5V) powers the input buffers and auxiliary circuitry.
- $V_{CCO}$ (1.5V, 1.8V, 2.5V, or 3.3V) powers the IOB circuitry.

SelectIO-Ultra is separated into eight banks of I/O groups. Each bank can be configured with one of several I/O standards. Refer to Chapter 3, “Design Considerations” for I/O banking rules and available I/O standards. Before and during configuration, all I/O banks are set for the LVCMOS25 (12mA) standard.

All dedicated configuration pins are powered by $V_{CC AUX}$. All dual-function configuration pins are located within banks 4 and 5. As described under “Configuration Process and Flow”, the $V_{CCO}$ input voltage is used as a logic input to the power-on-reset (POR) circuitry if the $V_{CCO}$ voltage level is monitored by the power-on-reset (POR) circuitry. If it drops below the minimum allowed voltage, the POR circuit will reset the entire device, clearing configuration memory and causing DONE to go Low.

Figure 4-4 summarizes the configuration $V_{CCO}$ requirements discussed in the following sections.

### Table 4-4: Configuration Modes and $V_{CCO}$ Voltages

<table>
<thead>
<tr>
<th>Configuration Mode</th>
<th>Pins Used</th>
<th>$V_{CCO_4}$</th>
<th>$V_{CCO_5}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>JTAG</td>
<td>Dedicated pins</td>
<td>1.5V min$^{(1)}$</td>
<td>not a concern</td>
</tr>
<tr>
<td>Serial</td>
<td>Dedicated pins plus DOUT, DIN, and INIT</td>
<td>2.5V/3.3V</td>
<td>not a concern</td>
</tr>
<tr>
<td>SelectMAP</td>
<td>Dedicated pins plus dual-function pins</td>
<td>2.5V/3.3V</td>
<td>2.5V/3.3V</td>
</tr>
</tbody>
</table>

**Notes:**

1. $V_{CCO\_4}$ is monitored by the POR circuit. See section “Power Up,” page 292.

### JTAG Configuration Mode

For JTAG configuration mode, JTAG inputs are independent of $V_{CCO}$ and work between 2.5V and 3.3V TTL levels ($V_{IL\_max} = 0.8V$, $V_{IH\_min} = 2.0V$). The JTAG input pins are 3.3V tolerant. The JTAG output (TDO) is an open-drain output and must be pulled up to the appropriate voltage level (typically 3.3V) through an external resistor. The value of the external pull-up resistor depends on the capacitive loading on the TDO pin and the operating frequency. It is recommended that a resistor of 200Ω or stronger be used. The optimal TDO pull-up value can be determined through IBIS simulation.

### Serial Configuration Mode

For Serial configuration mode, all of the configuration pins are in bank 4. $V_{CCO}$ pins of bank 4 can either be at 2.5V or 3.3V.

Xilinx recommends using the Platform Flash series of In-System Programmable Configuration PROMs. This family provides 3.3V and 2.5V output capability. In Master serial mode, the $V_{CCO}$ of the Platform Flash PROM and the $V_{CCO}$ of the Virtex-II Pro FPGA should connect to the same voltage level (2.5V/3.3V).
The DO pin of the PROM is at LVCMOS25 levels, and the DIN pin of the Virtex-II Pro is at LVTTL levels. Therefore, they are compatible. Figure 4-4 shows the Virtex-II Pro FPGA and the Platform Flash PROM in master serial configuration.

![Virtex-II Pro and Platform Flash PROM in Master Serial Configuration](image)

**Figure 4-4:** Virtex-II Pro and Platform Flash PROM in Master Serial Configuration

In Slave serial mode, the VCCO of the FPGA can be either 2.5V or 3.3V. However, make sure the CCLK pin of the FPGA is not exposed to more than 2.5V, since this is a 2.5V pin powered by VCCAUX.

**SelectMAP Configuration Mode**

For SelectMAP configuration mode, all of the configuration pins are in banks 4 and 5. VCCO pins for these banks can either be 2.5V or 3.3V.

**Special VCCO Requirements during Configuration and Readback**

Serial and SelectMAP configuration modes require 2.5V or 3.3V on the VCCO_4 and VCCO_5 inputs, per Table 4-4. If the I/Os in these banks are programmed for a voltage standard requiring some other voltage level, the VCCO level will need to be switched. For example, if the I/Os in bank 4 are programmed for 1.5V HSTL and the device is configured through the Master Serial mode, VCCO_4 will need to be held at 2.5V during configuration and switched to 1.5V after the DONE pin goes High.

**Configuration Solutions**

Several configuration solutions are available to support Virtex-II Pro devices, each targeted to specific application requirements. Guidance and support (application notes, reference designs, and so forth) is also available for designers looking to develop and implement their own configuration solution for Virtex FPGAs.

**System ACE CF (CompactFlash) Solution**

The System Advanced Configuration Environment (System ACE™) solution combines a standard CompactFlash™ Association (CFA) Type I or Type II memory module (CompactFlash or 1” disk drive) with a Xilinx-designed ACE Controller™ configuration control chip. See Figure 4-5.
The CompactFlash card stores an unlimited number of bitstreams, and supports densities up to 2 GB for FAT16 file formats. This card is capable of storing one large bitstream or several smaller bitstreams. If several bitstreams are used, the system can be set up so that individual bitstreams are callable as needed, allowing for dynamic reconfiguration of the Virtex-II Pro device and other Xilinx FPGAs in the JTAG chain.

The ACE Controller drives bits through the FPGA JTAG chain and has three other ports:

- A port for interfacing with a microprocessor or a network
- A port for interfacing with the CompactFlash card
- A port that provides access to the FPGA JTAG chain for FPGA testing or configuration via automatic test equipment or via desktop or third-party programmers

For further information on any System ACE product, visit the www.xilinx.com/systemace website.

Configuration PROMs

Using Platform Flash PROMs

The Platform Flash family of in-system programmable (ISP) configuration PROMs provides the flexibility of reprogrammability and the efficiency of small-footprint packages, combined with both serial and SelectMAP (parallel) FPGA configurability. The family offers single devices ranging in density from 1 Mb to 32 Mb, and PROMs can be cascaded to support even larger bitstreams. Data throughput rates of up to 264 Mb/s can be attained, and FPGA reconfiguration can be triggered via a JTAG command.

These PROMs are programmable using Xilinx iMPACT software via cable, HW-130, or standard third-party programmers.

Platform Flash PROMs are available in VO20, VO48, and FS48 packages. See Table 4-5, page 301 to determine which PROMs go with which Virtex-II FPGAs. Package drawings are available in Appendix B, “Platform Flash Family PROMs.”
Using XC17V00 PROMs

The XC17V00 family of one-time programmable (OTP) PROMs provides a proven, low-cost, compact, and pre-engineered configuration solution. Ranging from 1 Mb to 16 Mb, this family can also be daisy-chained to support larger bitstreams. This family supports serial configuration of Virtex-II Pro FPGAs; in addition, the XC17V08 and XC17V16 support SelectMAP configuration modes.

The XC17V00 family can be used for stabilized designs that are in a high-volume production flow and/or for designs requiring a low-cost solution. XC17V00 PROMs can be programmed either by using the HW-130 or by using a variety of third-party programmers. The XC17V00 PROMs are available in VO8, SO20, PC20, VQ44, and PC44 packages. Data sheets for PROMs are available at www.xilinx.com. See Table 4-5, page 301 to determine which PROMs go with which Virtex-II Pro FPGAs.

Flash PROMs With a CPLD Configuration Controller

Some designers prefer to leverage existing Flash memory in their system to store the configuration bitstreams. A small CPLD-based configuration controller can provide the mechanism to access the bitstreams in the FLASH and deliver them quickly to Virtex-II Pro devices. The following application notes describe the details for a serial or SelectMAP configuration architecture using FLASH memories and CPLDs:

- **XAPP079**, Configuring Xilinx FPGAs Using an XC9500 CPLD and Parallel PROM (www.xilinx.com/bvdocs/appnotes/xapp079.pdf) describes an architecture that configures a chain of Virtex-II Pro devices using Master-Serial mode. See Figure 4-6 for an example of FPGA configuration using a CPLD and a parallel PROM.

- **XAPP137**, Configuring Virtex FPGAs From Parallel EPROMs With a CPLD (www.xilinx.com/bvdocs/appnotes/xapp137.pdf) describes an architecture that configures one or more Virtex-II Pro devices using the Slave SelectMAP mode. See
Figure 4-7 for an example of FPGA configuration using a CPLD and a parallel EPROM.

Embedded Solutions

Using an Embedded Microcontroller

XAPP058, Xilinx In-System Programming Using an Embedded Microcontroller (www.xilinx.com/bvdocs/appnotes/xapp058.pdf) describes a compact and robust process that (re)configures Virtex-II Pro devices directly from a microprocessor through the JTAG test port of the Virtex-II Pro device. The process additionally supports (re)configuration of XC18V00 ISP PROMs and CPLDs that reside on the JTAG scan chain. Portable, reference C-code is provided with the application note for rapid implementation.

Using IEEE Standard 1532

Systems that implement an IEEE Standard 1532 player can configure Virtex-II Pro devices. Users require a 1532 BSDL file and a 1532 configuration data file. 1532 BSDL files for Xilinx devices and information on the Xilinx JDRIVE 1532 configuration engine are available here:


Choose the JDrive Engine entry from the selection menu on the left of the page.
PROM and System ACE CF Selection Guide

Use Table 4-5 to determine which PROM or System ACE CF device goes with which Virtex-II Pro FPGA.

**Table 4-5: PROM and System ACE Solutions for Virtex-II Pro Devices**

<table>
<thead>
<tr>
<th>Virtex-II Pro Device</th>
<th>Default Bitstream Size (bits)</th>
<th>PROM Solutions</th>
<th>System ACE CompactFlash (CF) Solution(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>One-Time Programmable (OTP)</td>
<td>In-System Programmable (ISP)</td>
</tr>
<tr>
<td>XC2VP2</td>
<td>1,305,376</td>
<td>XC17V01</td>
<td>XCF02P</td>
</tr>
<tr>
<td>XC2VP4</td>
<td>3,006,496</td>
<td>XC17V04</td>
<td>XCF04P</td>
</tr>
<tr>
<td>XC2VP7</td>
<td>4,485,408</td>
<td>XC17V08</td>
<td>XCF08P</td>
</tr>
<tr>
<td>XC2VP20</td>
<td>8,214,560</td>
<td>XC17V08</td>
<td>XCF08P</td>
</tr>
<tr>
<td>XC2VPX20</td>
<td>8,214,560</td>
<td>XC17V08</td>
<td>XCF08P</td>
</tr>
<tr>
<td>XC2VP30</td>
<td>11,589,920</td>
<td>XC17V08 + XC17V04</td>
<td>XCF16P</td>
</tr>
<tr>
<td>XC2VP40</td>
<td>15,868,192</td>
<td>XC17V16</td>
<td>XCF16P</td>
</tr>
<tr>
<td>XC2VP50</td>
<td>19,021,344</td>
<td>XC17V16 + XC17V04</td>
<td>XCF32P</td>
</tr>
<tr>
<td>XC2VP70</td>
<td>26,098,976</td>
<td>XC17V16 + XC17V16</td>
<td>XCF32P</td>
</tr>
<tr>
<td>XC2VPX70</td>
<td>26,098,976</td>
<td>XC17V16 + XC17V16</td>
<td>XCF32P</td>
</tr>
<tr>
<td>XC2VP100(2)</td>
<td>34,292,768</td>
<td>2 x XC17V16 + XC17V01</td>
<td>XCF32P(2)</td>
</tr>
</tbody>
</table>

**Notes:**

1. Requires XCCACE-TQ144 CompactFlash Controller
2. Assumes compression is used. Without compression, Xilinx recommends the System ACE configuration solution for this larger-density device. Storing configuration data by cascading multiple independent PROMs is an option, but requires careful signal integrity and power supply planning.
Software Support and Data Files

This section provides information on Xilinx device programming software and configuration-related data files.

iMPACT

To program Virtex-II Pro and other Xilinx devices with a personal computer, Xilinx provides iMPACT as a part of the ISE software package. A free version of iMPACT is also available through the WebPACK software suite. More information on WebPACK is available at:


Programming Cables

iMPACT supports several Xilinx programming cables that are compatible with Virtex-II Pro devices, including the Parallel-Cable III, Parallel-Cable IV, and Platform Cable USB. For more information on these cables, or to order programming cables online, please visit:


Boundary Scan Interconnect Testing for Virtex-II Pro Devices

Virtex-II Pro supports the EXTEST, INTEST, and SAMPLE/PRELOAD instructions that are required for Boundary Scan interconnect tests. Xilinx does not provide direct support for Boundary Scan software, although several third party suppliers offer Boundary Scan test equipment that is compatible with Virtex-II Pro. A list of third-parties offering Boundary Scan test equipment is available at:

http://www.support.xilinx.com/xlnx/xil_prodcat_landingpage.jsp?title=Configuration+Solutions

Choose Third-Party Tools from the selection menu on the left of the page.

In-System Programming Data Files

Many third party JTAG configuration solutions and in-system configuration solutions require an SVF (Serial Vector Format) or STAPL (Standard Test and Programming Language) file. The SVF and STAPL file formats are used to convey Boundary Scan instructions in a generic format. Customers requiring an SVF or STAPL file can use iMPACT software to generate these files. For more information on SVF, STAPL, and In-System Programming (ISP), see the following resources online:

- The iMPACT Software Manual is included on the software manuals web page.
- In-System Programming details are contained in Xilinx Application Note XAPP058.

Serial Programming Modes

In the serial programming modes, the FPGA is configured by loading one bit per CCLK cycle. In Master Serial mode, the FPGA drives the CCLK pin. In Slave Serial mode, the FPGAs CCLK pin is driven by an external source. In both serial configuration modes, the MSB of each data byte is always written to the DIN pin first.
Master Serial Mode

The Master Serial mode is designed so the FPGA can be configured from a Serial PROM, Figure 4-8. The speed of the CCLK is selectable by BitGen options, see “Appendix A, “BitGen and PROMGen Switches and Options.” Be sure to select a CCLK speed supported by the PROM.

Figure 4-8 shows a Master Serial FPGA configuring from a PROM. Figure 4-9 shows the timing relationships.

Notes:
1. If the Virtex-II Pro device has not selected the DriveDONE option, then an external pull-up resistor of 330Ω should be added to the DONE pin. This pull-up resistor is not needed if DriveDONE = Yes.

Notes:
1. For Master configurations, the CCLK does not transition until after initialization as indicated by arrow.
2. DOUT transitions on the falling edge of CCLK.
Slave Serial Mode

In serial configuration mode, the FPGA is configured by loading one bit per CCLK cycle. In Slave Serial mode, the FPGA’s CCLK pin is driven by an external source. In both serial configuration modes, the MSB of each data byte is always written to the DIN pin first. The Slave Serial configuration mode allows for FPGAs to be configured from other logic devices (e.g., microprocessors) or in daisy-chain fashion. Figure 4-10 shows a Master Serial FPGA configuring from a PROM with a Slave Serial FPGA in a daisy chain with the Master.

Daisy-Chain Configuration

Virtex-II Pro devices can be daisy-chained with the Virtex, Spartan-II, Virtex-E, Spartan-IIE, Virtex-II, Spartan-3, and Virtex-4 families. There are three important design considerations when designing a mixed-serial daisy chain:

1. Many older devices cannot accept as fast a CCLK frequency as (for example) a Virtex-4 device can generate. Select a configuration CCLK speed supported by all devices in the chain.
2. Newer devices should be grouped at the beginning of the serial daisy chain, with older devices located at the end of the chain.
3. There is a limit to the number of configuration bits a device can pass through its DOUT pin. As shown in Table 4-6, this limit varies for different families. The sum of the bitstream lengths for all downstream devices must not exceed the number in this table for each family.

Notes:
1. If none of the devices have been selected to DriveDONE, then an external pull-up resistor of 330Ω should be added to the common DONE line. This pull-up resistor is not needed if DriveDONE = Yes. If used, DriveDONE should be enabled only for the last device in the configuration chain.
Serial Programming Modes

The separate bitstreams for the FPGAs in a daisy chain must be combined into a single PROM file, by using either iMPACT or the PROMGen utility (see Appendix A, “BitGen and PROMGen Switches and Options”). Separate .bit files cannot be simply concatenated together to form a daisy-chain bitstream.

The first device in the chain is the first to be configured. No data is passed onto the DOUT pin until all the data frames, start-up command, and CRC check have been loaded. CRC checks only include the data for the current device, not for any others in the chain. After finishing the first stream, data for the next device is loaded. The data for the downstream device appears on DOUT typically about 80 CCLK cycles after being loaded into DIN. This is due to internal packet processing. Each daisy-chained bitstream carries its own synchronization word. Nothing of the first bitstream is passed to the next device in the chain other than the daisy-chained configuration data.

The DONE_cycle must be set before GTS, or during the same cycle to guarantee each Virtex-II Pro device to move to the operation state when all the DONE pins have been released. When daisy-chaining multiple devices, either set the last device in the chain to DriveDONE, or add external pull-up resistors to counteract the combined capacitive loading on DONE. If non-Virtex devices are included in the daisy chain, it is important to set their bitstreams to SyncToDONE with BitGen options. For more information on Virtex BitGen options, see Appendix A, “BitGen and PROMGen Switches and Options.”

### Table 4-6: Maximum Number of Configuration Bits, Various Device Families

<table>
<thead>
<tr>
<th>Architecture</th>
<th>Maximum DOUT Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Virtex-4</td>
<td>$32 \times (2^{27} - 1) = 4,294,967,264$</td>
</tr>
<tr>
<td>Virtex-II Pro, Virtex-II</td>
<td>$32 \times (2^{27} - 1) = 4,294,967,264$</td>
</tr>
<tr>
<td>Spartan-3</td>
<td>$32 \times (2^{27} - 1) = 4,294,967,264$</td>
</tr>
<tr>
<td>Virtex, Virtex-E, Spartan-II, Spartan-IIIE</td>
<td>$32 \times (2^{20} - 1) = 33,554,216$</td>
</tr>
</tbody>
</table>

The DONE_cycle must be set before GTS, or during the same cycle to guarantee each Virtex-II Pro device to move to the operation state when all the DONE pins have been released. When daisy-chaining multiple devices, either set the last device in the chain to DriveDONE, or add external pull-up resistors to counteract the combined capacitive loading on DONE. If non-Virtex devices are included in the daisy chain, it is important to set their bitstreams to SyncToDONE with BitGen options. For more information on Virtex BitGen options, see Appendix A, “BitGen and PROMGen Switches and Options.”

### Notes:

1. For Slave configurations, a free running CCLK can be used, as shown in Figure 4-11.
2. DOUT transitions on the falling edge of CCLK.

### Figure 4-11: Serial Configuration Clocking Sequence

[Diagram of Serial Configuration Clocking Sequence]
SelectMAP Programming Modes

The SelectMAP™ interface provides an 8-bit bidirectional data bus interface to the Virtex-II Pro configuration logic. In Master SelectMAP mode the CCLK signal is an output from the FPGA. In Slave SelectMAP mode the CCLK signal is an input. Slave SelectMAP mode allows for both configuration and readback, while only configuration is possible in Master SelectMAP mode.

Table 4-7 describes the SelectMAP configuration interface pins.

Table 4-7: SelectMAP Configuration Interface Pins

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Type</th>
<th>Dedicated or Dual-Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>D[0:7]</td>
<td>Bidirectional</td>
<td>Dual-Function</td>
<td>The D[0:7] pins function as a bi-directional data bus for the SelectMAP interface, with D0 serving as the MSB. The D[0:7] bus direction is controlled by the RDWR_B signal. When the SelectMAP interface is enabled (CS_B=0) and the port is set for write control (RDWR_B=0), the D[0:7] signals are registered by rising CCLK edges. When the SelectMAP interface is disabled (CS_B=1), the D[0:7] pins are placed in a high-Z state. After configuration, the D[0:7] pins become user I/O unless the bitgen -g persist:yes setting is used.</td>
</tr>
<tr>
<td>RDWR_B</td>
<td>Input</td>
<td>Dual-Function</td>
<td>The RDWR_B signal determines whether the SelectMAP interface is set for read or write control. 0 = write control, D[0:7] are inputs 1 = read control, D[0:7] are outputs The RDWR_B pin is registered on rising CCLK edges. If a new RDWR_B value is registered by the device while the SelectMAP interface is enabled (CS_B=0), an ABORT will occur (see “SelectMAP ABORT Sequence and ABORT Recovery,” page 314). After configuration, the RDWR_B pin becomes user I/O unless the bitgen -g persist:yes setting is used.</td>
</tr>
<tr>
<td>CS_B</td>
<td>Input</td>
<td>Dual-Function</td>
<td>Active-Low chip select to enable the SelectMAP data bus. 0 = SelectMAP data bus enabled 1 = SelectMAP data bus disabled. When the CS_B signal is deasserted, the SelectMAP interface is asynchronously disabled and all SelectMAP outputs are placed in a high-Z state. After configuration, the CS_B pin becomes user I/O unless the bitgen -g persist:yes setting is used.</td>
</tr>
<tr>
<td>BUSY</td>
<td>Three-State Output</td>
<td>Dual-Function</td>
<td>The BUSY pin is an active-High three-state output. 0 = BUSY deasserted 1 = BUSY asserted The BUSY pin has two functions: 1. During configuration, the BUSY pin indicates that data are being loaded too quickly. Values on the D[0:7] bus are ignored while BUSY is active. 2. During readback, BUSY indicates when readback data is valid. When the SelectMAP interface is disabled (CS_B=1), the BUSY pin is placed in a high-Z state. After configuration, the BUSY pin becomes user I/O unless the bitgen -g persist:yes setting is used.</td>
</tr>
<tr>
<td>M[2:0]</td>
<td>Input</td>
<td>Dedicated</td>
<td>MODE pins. Specify configuration mode.</td>
</tr>
</tbody>
</table>
Daisy-chaining is not possible for SelectMAP mode, however multiple devices can be connected on the same SelectMAP bus as shown in Figure 4-14, page 310. To connect devices on a SelectMAP bus, the DATA pins (D0:D7), CCLK, RDWR_B, BUSY, PROG_B, DONE, and INIT_B are connected in common between all devices. The CS_B (Chip Select) inputs are kept separate so that each device can be accessed individually. External control logic is required to arbitrate between devices by asserting and deasserting the CS_B signals as necessary.

Alternatively, it is possible to simultaneously configure multiple devices using the same bitstream, provided that readback is not required. In this arrangement the CS_B pins are connected together, along with the remaining configuration pins, so that the SelectMAP ports on all devices are enabled simultaneously.

Table 4-7: SelectMAP Configuration Interface Pins (Continued)

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Type</th>
<th>Dedicated or Dual-Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCLK</td>
<td>Input or Output</td>
<td>Dedicated</td>
<td>Configuration clock for all modes other than JTAG. Input for Slave SelectMAP mode; output for Master SelectMAP mode.</td>
</tr>
<tr>
<td>PROG_B</td>
<td>Input</td>
<td>Dedicated</td>
<td>Active-Low asynchronous full-chip reset. Has permanent weak pull-up.</td>
</tr>
</tbody>
</table>
| INIT_B   | Bi-directional        | Dual-Function             | Before MODE pins are sampled, INIT_B is an input that can be held Low to delay configuration. After MODE pins are sampled, INIT_B is an open-drain, active Low output indicating whether a CRC error occurred during configuration: 
0 = CRC error 
1 = No CRC error 
After configuration, the INIT_B pin becomes user I/O unless the bitgen -g persist:yes setting is used. |
| DONE     | Input or Output       | Dedicated                 | Active High signal indicating configuration is complete: 
0 = FPGA not configured 
1 = FPGA configured 
Pin has optional internal pull-up. |
Master SelectMAP Mode

The Master SelectMAP mode is used to connect a single Virtex-II FPGA to one or more configuration PROMs as shown in Figure 4-12.

**Notes:**
1. The DONE pin is by default an open-drain output requiring an external pull-up resistor. In this arrangement, the active DONE driver can be enabled, eliminating the need for an external pull-up resistor.
2. The INIT_B pin is a bidirectional, open-drain pin. An external pull-up is required.
3. The BitGen startup clock setting must be set for CCLK for SelectMAP configuration.
4. The PROM in this diagram represents one or more Xilinx serial PROMs. Multiple serial PROMs can be cascaded to increase the overall configuration storage capacity.
5. The .bit file must be reformatted into a PROM file before it can be stored on the serial PROM. Refer to Appendix A, “BitGen and PROMGen Switches and Options.”
6. On XC17V00 devices, the reset polarity is programmable. RESET should be set for active Low when using an XC17V00 device in this setup.
7. The Xilinx PROM must be set for parallel mode. (This mode is not available for all devices.)
8. When configuring a Virtex-II Pro device in SelectMAP mode from a Xilinx configuration PROM, the RDWR_B and CS_B signals can be tied Low (see section “Master SelectMAP Data Loading”).
9. The BUSY signal does not need to be monitored for this setup and can be left unconnected (see section “Master SelectMAP Data Loading”).

The following pins are involved in Master SelectMAP configuration mode:

**DATA Pins (D[0:7])**

The D0 through D7 pins function as a bidirectional data bus in the SelectMAP mode. Configuration data is written to the bus, and readback data is read from the bus. The bus direction is controlled by the RDWR_B signal. See “Configuration Details,” page 337. The D0 pin is considered the MSB of each byte.

**RDWR_B**

When asserted Low, the RDWR_B signal indicates that data is being written to the data bus. When High, the RDWR_B signal indicates that data is being read from the data bus.
CS_B

The Chip Select input (CS_B) enables the SelectMAP data bus. To write or read data onto or from the bus, the CS_B signal must be asserted Low. When CS_B is High, Virtex-II Pro devices do not drive onto or read from the bus.

CCLK

The CCLK pin is a clock output in the Master SelectMAP interface. It synchronizes all loading and reading of the data bus for configuration and readback. The CCLK pin is driven by the FPGA.

Master SelectMAP Data Loading

To load data in the Master SelectMAP mode, a data byte is loaded on every rising CCLK edge as shown in Figure 4-13. If BUSY is being monitored during configuration (for example, when configuring with an encrypted bitstream) and it is High, the current byte must be reloaded when BUSY is Low.

The first byte can be loaded on the first rising CCLK edge that INIT_B is High, and when both CS_B and RDWR_B are asserted Low. CS_B and RDWR_B can be asserted anytime before or after INIT_B has gone High. However, the SelectMAP interface is not active until after INIT_B has gone High. If RDWR_B toggles while CS_B is asserted, an ABORT occurs.

Extra processing time is needed when the bitstream is encrypted. Because of this, BUSY handshaking is required for encrypted bitstreams at a lower speed than for non-encrypted bitstreams. The maximum speed at which a Virtex-II Pro/Virtex-II Pro X device can be configured via SelectMAP without BUSY handshaking is 50 MHz.

Slave SelectMAP Mode

One or more Virtex-II Pro devices in Slave SelectMAP mode can be connected on a common SelectMAP bus (Figure 4-14). In a SelectMAP bus, the DATA pins (D[0:7]), CCLK, RDWR_B, BUSY, PROG_B, DONE, and INIT_B share a common connection between all of the devices. To allow each device to be accessed individually, the CS_B (Chip Select) inputs...
must not be tied together. External control of the CS_B signal is required and is usually provided by a microprocessor or CPLD.

Figure 4-14: Slave SelectMAP Mode Circuit Diagram

Notes:
1. If none of the Virtex-II Pro devices have been selected to DriveDONE, add an external 330Ω pull-up resistor to the common DONE line. This pull-up resistor is not needed if DriveDONE = Yes. If used, DriveDONE should be selected only for the last device in the configuration chain.

If Readback is going to be performed on the device after configuration, the RDWR_B and BUSY signals must be handled appropriately. (For details, refer to “Readback,” page 365.)

Figure 4-14 does not show a control module for the SelectMAP interface. The SelectMAP interface is typically driven by a processor, microcontroller, FPGA or CPLD.

Controlling SelectMAP Data Flow

The rules for sending configuration data to a Virtex-II Pro device via the SelectMAP interface follow:

- All SelectMAP inputs are ignored until the INIT_B pin goes high after power-up. The RDWR_B and CS_B signals can be asserted prior to or during power-up without triggering an ABORT.
- After INIT_B has gone high, the RDWR_B signal should be set before asserting CS_B. Registering a new value on RDWR_B while CS_B is enabled will trigger an ABORT. If necessary, the RDWR_B signal can be toggled without triggering an ABORT if:
  - CS_B is deasserted, or
  - if CS_B is asserted BUT no positive CCLK edge arrives.

Refer to “Toggling between Read and Write Control” below for details.
SelectMAP Programming Modes

- When the SelectMAP port is set for write control (RDWR_B=0), the device samples the D0:7 inputs on each rising CCLK edge.
- The BUSY signal must be monitored during readback and when bitstream encryption is used during configuration.

There are several ways to control configuration data loading using Slave SelectMAP mode:

1. **Continuous Data Loading (Free-Running CCLK).**

   After the INIT_B signal goes High, the device registers the D[0:7] inputs on every rising CCLK edge while the SelectMAP interface is enabled (CS_B=0) and set for write control (RDWR_B=0). Configuration data are loaded continuously, without pauses. This method is depicted in Figure 4-15.

![Figure 4-15: Continuous Data Loading for SelectMAP Modes](ug071_25_121903)

**Notes:**

1. CS_B signal can be tied Low if there is only one device on the SelectMAP bus. If CS_B is not tied Low, it can be asserted at any time.
2. RDWR_B can be tied Low if readback is not needed. RDWR_B should not be toggled after CS_B has been asserted, as this triggers an ABORT. (See “SelectMAP ABORT Sequence and ABORT Recovery.”)
3. If CS_B is tied Low, BUSY is driven Low before INIT_B toggles High.
4. The MODE pins are sampled when INIT_B goes High.
5. RDWR_B should be asserted before CS_B to avoid causing an ABORT.
6. CS_B is asserted, enabling the SelectMAP interface.
7. BUSY remains in high-Z state until CS_B is asserted.
8. The first byte is loaded on the first rising CCLK edge after CS_B is asserted.
9. The configuration bitstream is loaded one byte per rising CCLK edge.
10. After the last byte is loaded the device enters the startup sequence.
11. The startup sequence lasts a minimum of eight CCLK cycles.
12. The DONE pin goes High during the startup sequence. Additional CCLKs could be required to complete the startup sequence.
13. After configuration has finished, the CS_B signal can be deasserted.
14. After the CS_B signal is deasserted, RDWR_B can be deasserted.
2. **SelectMAP Data Loading Controlled by the CS_B signal (Non-Contiguous Data Strobe).**

In some cases configuration will need to be paused while the next byte of configuration data is presented on the D[0:7] pins. One way of pausing SelectMAP configuration is to disable the SelectMAP interface (CS_B=0). When the SelectMAP interface is disabled, the D[0:7] and BUSY pins are placed in a high-Z state, and all inputs are ignored. The D[0:7] signals are sampled on the next rising CCLK edge after the SelectMAP interface is enabled. See Figure 4-16.

![Figure 4-16: SelectMAP Data Loading Controlled by the CS_B Signal](image)

**Notes:**

1. RDWR_B is driven Low by the user, setting the D[0:7] pins as inputs for configuration. RDWR_B can be tied Low if readback is not needed. RDWR_B should not be toggled after CS_B has been asserted, as this triggers an ABORT. (See “SelectMAP ABORT Sequence and ABORT Recovery.”)
2. Device is ready for configuration after INIT_B toggles High.
3. The user asserts CS_B Low, enabling SelectMAP data bus. CS_B signal can be tied Low if there is only one device on the SelectMAP bus. If CS_B is not tied Low, it can be asserted at any time.
4. BUSY goes Low shortly after CS_B is asserted. If CS_B is tied Low, BUSY is driven Low before INIT_B toggles High.
5. Byte loaded on rising CCLK edge.
6. Byte loaded on rising CCLK edge.
7. The user deasserts CS_B; byte ignored.
8. The user deasserts CS_B; byte ignored.
9. Byte loaded on rising CCLK edge.
10. Byte loaded on rising CCLK edge.
11. The user deasserts CS_B; byte ignored.
12. Byte loaded on rising CCLK edge.
13. Byte loaded on rising CCLK edge.
14. Byte loaded on rising CCLK edge.
3. **Data Loading Controlled by Gating CCLK (Controlled CCLK).**

An alternate way to control SelectMAP data loading is to gate (halt) the CCLK signal. If needed, the RDWR_B signal can be toggled during this time, as long as it is returned to its prior value before the next rising CCLK edge. If the RDWR_B signal is driven high (set for read control) while the SelectMAP interface is enabled (CS_B=0), the D[0:7] pins will become outputs, and will be driven actively by the device. The D[0:7] signals will retain their state while CCLK is paused. See **Figure 4-17**.

![Data Loading Controlled by gating CCLK](ug071_27_080204)

**Figure 4-17:** *Data Loading Controlled by gating CCLK*

**Notes:**
1. D[0:7] pins are in High-Z state while CS_B is deasserted.
2. RDWR_B has no effect on the device while CS_B is deasserted.
3. CS_B is asserted by the user. Device begins loading configuration data on rising CCLK edges.
4. Byte loaded on rising CCLK edge.
5. Byte loaded on rising CCLK edge.
6. Byte loaded on rising CCLK edge.

**Toggling between Read and Write Control**

This section explains how to change the SelectMAP interface from write control to read control and vise-versa. This is only needed for readback operations, during which the SelectMAP port must be set for write control to send commands to the device, then toggled to read control to retrieve configuration data from the device (see “Readback,” page 365). For designs that do not require readback, the RDWR_B signal can be tied Low.

Write and read control for the SelectMAP interface is determined by the RDWR_B input: the SelectMAP data pins (D0:7) are inputs when the interface is set for Write control (RDWR_B = 0); they are outputs when the interface is set for Read control (RDWR_B = 1).

The procedure for changing the SelectMAP interface from Write to Read Control, or vice-versa, is:

1. **Deassert CS_B (set CS_B = 1).**
   - After deasserting CS_B, the device reacts by asynchronously disabling the SelectMAP interface, placing the D[0:7] pins and the BUSY pin in a high-Z state.

2. **Toggle RDWR_B**
   - RDWR_B = 0 : Write control
   - RDWR_B = 1 : Read control

3. **Assert CS_B (set CS_B = 0)**
   - After asserting CS_B, the BUSY signal and the D[0:7] pins will become active on the next rising CCLK edge, with the BUSY signal asserted. After a few CCLK cycles (the exact number is variable), the device will deassert the BUSY signal.
When switching to write control (writing data to the FPGA), the device begins sampling data on the D[0:7] pins after it deasserts the BUSY signal. Similarly, when switching to read control, the device presents valid data on the D[0:7] pins after it deasserts the BUSY signal.

This procedure is illustrated in Figure 4-18.

![Figure 4-18: Changing the SelectMAP Port from Write to Read Control](image)

**SelectMAP ABORT Sequence and ABORT Recovery**

An ABORT is an interruption in the SelectMAP configuration or readback sequence that occurs when the user improperly toggles the RDWR_B signal. In some cases, this might be done intentionally, because the device will provide a status word that can be useful in debugging configuration problems.

**Triggering an ABORT**

An ABORT occurs when the device registers a new RDWR_B state while CS_B remains asserted and a rising CCLK edge occurs.

**ABORT During Configuration**

An ABORT is triggered during configuration as follows:

1. Configuration sequence begins normally.
2. User pulls the RDWR_B pin High while CS_B is still asserted.
3. The device sees a rising edge on CCLK.
4. If CS_B remains asserted and RDWR_B remains High (SelectMAP interface is set for read control), the device drives BUSY High and it drives the ABORT Status word on the D[0:7] outputs (see “ABORT Status Word,” page 316).
5. The ABORT ends after four CCLK cycles, regardless of whether or not CS_B is asserted.

This sequence is illustrated by the timing diagram in Figure 4-19.
**ABORT During Readback**

An ABORT is triggered during readback as follows:

1. Readback sequence begins normally.
2. User pulls the RDWR_B pin Low while CS_B is still asserted.
3. The device sees a rising edge on CCLK.
4. If CS_B remains asserted and RDWR_B remains Low (SelectMAP interface is set for write control), the device places BUSY and the D[0:7] pins in a high-Z state.
5. The ABORT ends after four CCLK cycles, regardless of whether or not CS_B is asserted. If the RDWR_B signal is toggled to read control (RDWR_B=0) before or during the ABORT, the current status byte will appear on the D[0:7] pins.

This sequence is illustrated by the timing diagram in Figure 4-20.

Note that ABORTs during readback are not followed by a status word if the RDWR_B signal remains set for write control.
ABORT Status Word

During the configuration ABORT sequence, the device drives a status word onto the D[0:7] pins over four CCLK cycles, provided that the SelectMAP interface is set for read control during the ABORT. Each byte gives the status of the same four signals. The key for each ABORT byte is given in Table 4-8.

During the ABORT sequence, the device drives a status word onto the D[0:7] pins. The key for that status word is as follows:

<table>
<thead>
<tr>
<th>Bit Number</th>
<th>Status Bit Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>D7</td>
<td>CFGERR_B</td>
<td>Configuration error (active Low)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = A configuration error has occurred.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = No configuration error.</td>
</tr>
<tr>
<td>D6</td>
<td>DALIGN</td>
<td>Sync word received (active High)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = No sync word received.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = Sync word received by interface logic.</td>
</tr>
<tr>
<td>D5</td>
<td>RIP</td>
<td>Readback in progress (active High)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = No readback in progress.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = A readback is in progress.</td>
</tr>
<tr>
<td>D4</td>
<td>IN_ABORT_B</td>
<td>ABORT in progress (active Low)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0 = ABORT is in progress.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1 = No ABORT in progress.</td>
</tr>
<tr>
<td>D3-D0</td>
<td>1111</td>
<td></td>
</tr>
</tbody>
</table>

The ABORT sequence lasts for four CCLK cycles. The status byte reflects data alignment, ABORT status, and whether readback is in progress. An example sequence is given in Table 4-9.

<table>
<thead>
<tr>
<th>Byte Number</th>
<th>D[0:7]</th>
<th>DALIGN</th>
<th>IN_ABORT_B</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1111011</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1110011</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1111001</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1111101</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

After the last cycle, the synchronization word can be reloaded to establish data alignment.

ABORT Recovery

There are two procedures for recovering from an ABORT—one simple, the other more complex. These procedures apply to both read- and write-ABORT recovery.

ABORT Recovery Procedure 1

Pulse the PROG_B pin at any time during or after the ABORT sequence. Configuration must start over completely.
ABORT Recovery Procedure 2

This procedure is more complicated, but it does not involve clearing configuration memory. This allows configuration or readback to resume from where it was interrupted by the ABORT sequence. To take advantage of this capability, however, users must have fine control over, and a detailed understanding of, the configuration process. Use of this procedure will require a thorough knowledge of the material in the “Configuration Details” and “Readback” sections.

1. Wait until the ABORT sequence has finished
2. Toggle the SelectMAP interface to Write control
3. Write the Synchronization Word
4. Write a value equal to the (number of words per frame – 1) to the FLR
5. Write the RCFG command
6. Set the FAR to address $FFFFFF (out of bounds)
7. Send the FDRO read header to the device. The read length is a number of words equal to the frame length.
8. Write four NO-OP commands
9. Toggle the SelectMAP interface to Read control
10. Read 1 frame from the device
11. Toggle the SelectMAP interface to Write control

Steps to Resume Readback:

12. Send RCFG command
13. Set the FAR to the desired address
14. Send the FDRO read header
15. etc…

Steps to Resume Configuration:

12. Send the WCFG command
13. Set the FAR to the desired address
14. Send the FDRI read header
15. etc…

Internal Configuration Access Port (ICAP)

Virtex-II Pro devices feature an Internal Configuration Access Port (ICAP) that provides a configuration interface to the FPGA fabric. This allows the user design to control configuration and readback operations. The ICAP interface uses the same protocol as the SelectMAP interface in Slave mode, although it features separate data input and output ports, whereas the D[0:7] pins on the SelectMAP interface are bidirectional (see Table 4-7, page 306). To use ICAP, the ICAP_VIRTEX2 primitive must be instantiated in the user design. The ICAP_VIRTEX2 primitive is shown in Figure 4-21.
The ports on the ICAP primitive correspond to ports on the SelectMAP interface as shown in Table 4-10. The ICAP ports behave exactly as do the equivalent SelectMAP ports.

**Table 4-10: ICAP Ports and Equivalent SelectMAP Ports**

<table>
<thead>
<tr>
<th>ICAP Port</th>
<th>Equivalent SelectMAP Port</th>
</tr>
</thead>
<tbody>
<tr>
<td>BUSY</td>
<td>BUSY</td>
</tr>
<tr>
<td>O[0:7]</td>
<td>D[0:7] when SMAP port is set for read control</td>
</tr>
<tr>
<td>CE</td>
<td>CS_B</td>
</tr>
<tr>
<td>CLK</td>
<td>CCLK</td>
</tr>
<tr>
<td>I[0:7]</td>
<td>D[0:7] when SMAP port is set for write control</td>
</tr>
<tr>
<td>WRITE</td>
<td>RDWR_B</td>
</tr>
</tbody>
</table>

The following restrictions govern the use of ICAP:

- ICAP and SelectMAP cannot be used together after configuration.
  - After the DONE pin goes High, the ICAP port will be disabled if the PERSIST bit is set.
  - A design that instantiates the ICAP port can be configured via SelectMAP, provided that the PERSIST bit is not set.

- Care must be taken when using the JTAG interface on a design that also accesses the ICAP port.
  - The JTAG CFG_IN, CFG_OUT, JSTART, and JSHUTDOWN instructions must not be sent to the device while configuration or readback operations are being performed through the ICAP port.
  - After finishing a configuration or readback operation on the JTAG interface, the configuration logic should be desynchronized before performing any operations using the ICAP port. Conversely, after finishing an operation on the ICAP port, the configuration logic should be desynchronized before performing any operations using the JTAG interface.
JTAG / Boundary-Scan Programming Modes

Introduction

Virtex-II Pro devices support the new IEEE 1532 standard for In-System Configuration (ISC), based on the IEEE 1149.1 standard. The IEEE 1149.1 Test Access Port and Boundary-Scan Architecture is commonly referred to as JTAG. JTAG is an acronym for the Joint Test Action Group, the technical subcommittee initially responsible for developing the standard. This standard provides a means to assure the integrity of individual components and the interconnections between them at the board level. With increasingly dense multi-layer PC boards, and more sophisticated surface mounting techniques, boundary-scan testing is becoming widely used as an important debugging standard.

Devices containing boundary-scan logic can send data out on I/O pins in order to test connections between devices at the board level. The circuitry can also be used to send signals internally to test the device specific behavior. These tests are commonly used to detect opens and shorts at both the board and device level.

In addition to testing, boundary-scan offers the flexibility for a device to have its own set of user-defined instructions. The added common vendor specific instructions, such as configure and verify, have increased the popularity of boundary-scan testing and functionality.

Boundary-Scan for Virtex-II Pro Devices Using IEEE Standard 1149.1

The Virtex-II Pro family is fully compliant with the IEEE Standard 1149.1 Test Access Port and Boundary-Scan Architecture. The architecture includes all mandatory elements defined in the IEEE 1149.1 Standard. These elements include the Test Access Port (TAP), the TAP controller, the instruction register, the instruction decoder, the boundary-scan register, and the bypass register. The Virtex-II Pro family also supports some optional instructions; the 32-bit identification register, and a configuration register in full compliance with the standard. Outlined in the following sections are the details of the JTAG architecture for Virtex-II Pro devices.

Test Access Port

The Virtex-II Pro TAP contains four mandatory dedicated pins as specified by the protocol.

Table 4-11: Virtex-II Pro TAP Controller Pins

<table>
<thead>
<tr>
<th>Pin</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDI</td>
<td>Test Data In</td>
</tr>
<tr>
<td>TDO</td>
<td>Test Data Out</td>
</tr>
<tr>
<td>TMS</td>
<td>Test Mode Select</td>
</tr>
<tr>
<td>TCK</td>
<td>Test Clock</td>
</tr>
</tbody>
</table>

There are three input pins and one output pin to control the 1149.1 boundary-scan TAP controller. There are optional control pins, such as TRST (Test Reset) and enable pins, which might be found on devices from other manufacturers. It is important to be aware of these optional signals when interfacing Xilinx devices with parts from different vendors, because they might need to be driven.

The TAP controller is a 16-state state machine shown in Figure 4-22. The four mandatory TAP pins are outlined below.
• TMS - This pin determines the sequence of states through the TAP controller on the rising edge of TCK. TMS has an internal resistive pull-up to provide a logic High if the pin is not driven.

• TCK - This pin is the JTAG test clock. It sequences the TAP controller and the JTAG registers in the Virtex-II Pro devices.

• TDI - This pin is the serial input to all JTAG instruction and data registers. The state of the TAP controller and the current instruction held in the instruction register determine which register is fed by the TDI pin for a specific operation. TDI has an internal resistive pull-up to provide a logic High to the system if the pin is not driven. TDI is applied into the JTAG registers on the rising edge of TCK.

• TDO - This pin is the serial output for all JTAG instruction and data registers. The state of the TAP controller and the current instruction held in the instruction register determine which register (instruction or data) feeds TDO for a specific operation. TDO changes state on the falling edge of TCK and is only active during the shifting of instructions or data through the device. This pin is an open-drain output and should be pulled up by an external resistor to the appropriate level (typically 3.3V).

  **Note:** As specified by the IEEE Standard, the TMS and TDI pins all have internal pull-up resistors by default. These internal pull-up resistors of 50-150 kΩ are active, regardless of the mode selected. User can select a pull-up, pull-down, or float option in the implementation tools.

For JTAG configuration mode, JTAG inputs are independent of $V_{CCO}$ and work between 2.5V and 3.3V TTL levels ($V_{IL\ max} = .8V$, $V_{IH\ min} = 2.0V$). The JTAG input pins are 3.3V tolerant. The JTAG output (TDO) is an open-drain output and must be pulled up to the appropriate voltage level (typically 3.3V) through an external resistor. The value of the external pull-up resistor depends on the capacitive loading on the TDO pin and the operating frequency, but it should not be less than 200Ω. The optimal TDO pull-up value can be determined through IBIS simulation.

### TAP Controller

Figure 4-22, page 321 diagrams a 16-state finite state machine. The four TAP pins control how data is scanned into the various registers. The state of the TMS pin at the rising edge of TCK determines the sequence of state transitions. There are two main sequences, one for shifting data into the data register and the other for shifting an instruction into the instruction register.

### Boundary-Scan Instruction Set

To determine the operation to be invoked, an instruction is loaded into the Instruction Register (IR). The length of the IR is device size specific. The IR is 10 bits wide on the XC2VP2, XC2VP4, and XC2VP7, and 14 bits wide on the XC2VP20, XC2VP30, XC2VP40, XC2VP50, XC2VP70, and XC2VP100. The bottom six bits of the instruction codes are the same for all devices sizes, to support the new IEEE Standard 1532 for In-System Configurable (ISC) devices. The additional IR bits for each instruction are 1s. Table 4-12 lists the available instructions for Virtex-II Pro devices.
### JTAG / Boundary-Scan Programming Modes

**Figure 4-22:** State Diagram for the TAP Controller

**Table 4-12:** Virtex-II Pro Boundary Scan Instructions for XC2VP2 through XC2VP7

<table>
<thead>
<tr>
<th>Boundary Scan Command</th>
<th>Binary Code (9:0)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EXTEST</td>
<td>1111000000</td>
<td>Enables boundary-scan EXTEST operation</td>
</tr>
<tr>
<td>SAMPLE</td>
<td>1111000001</td>
<td>Enables boundary-scan SAMPLE operation</td>
</tr>
<tr>
<td>USER1</td>
<td>1111000010</td>
<td>Access user-defined register 1</td>
</tr>
<tr>
<td>USER2</td>
<td>1111000011</td>
<td>Access user-defined register 2</td>
</tr>
<tr>
<td>CFG_OUT</td>
<td>1111000100</td>
<td>Access the configuration bus for readback</td>
</tr>
<tr>
<td>CFG_IN</td>
<td>1111000101</td>
<td>Access the configuration bus for configuration</td>
</tr>
<tr>
<td>INTEST</td>
<td>1111000111</td>
<td>Enables boundary-scan INTEST operation</td>
</tr>
<tr>
<td>USERCODE</td>
<td>1111001000</td>
<td>Enables shifting out user code</td>
</tr>
<tr>
<td>IDCODE</td>
<td>1111001001</td>
<td>Enables shifting out of ID code</td>
</tr>
<tr>
<td>HIGHZ</td>
<td>1111001010</td>
<td>3-states output pins while enabling the bypass register</td>
</tr>
<tr>
<td>JSTART</td>
<td>1111001100</td>
<td>Clocks the start-up sequence when StartClk is TCK</td>
</tr>
<tr>
<td>JSHUTDOWN</td>
<td>1111001101</td>
<td>Clocks the shutdown sequence</td>
</tr>
<tr>
<td>BYPASS</td>
<td>1111111111</td>
<td>Enables BYPASS</td>
</tr>
<tr>
<td>JPROG_B</td>
<td>1111001011</td>
<td>Equivalent to and has the same affect as PROG_B</td>
</tr>
<tr>
<td>RESERVED</td>
<td>All other codes</td>
<td>Xilinx reserved instructions</td>
</tr>
</tbody>
</table>
The mandatory IEEE 1149.1 commands are supported in Virtex-II Pro devices, as well as several Xilinx vendor-specific commands. Virtex-II Pro devices have a powerful command set. The EXTEST, INTEST, SAMPLE/PRELOAD, BYPASS, IDCODE, USERCODE, and HIGHZ instructions are all included. The TAP also supports two internal user-defined registers (USER1 and USER2) and configuration/readback of the device. The Virtex-II Pro boundary-scan operations are independent of mode selection. The boundary-scan mode in Virtex-II Pro devices overrides other mode selections. For this reason, boundary-scan instructions using the boundary-scan register (SAMPLE/PRELOAD, INTEST, EXTEST) must not be performed during configuration. All instructions except USER1 and USER2 are available before a Virtex-II Pro device is configured. After configuration, all instructions are available.

JSTART and JSHUTDOWN are instructions specific to the Virtex-II Pro architecture and configuration flow. As described in Table 4-12, the JSTART and JSHUTDOWN instructions clock the startup sequence when the appropriate bitgen option is selected. The instruction does not work correctly without the correct bitgen option selected.

```
bitgen -g startupclk:jtagclk designName.ncd
```

For details on the standard boundary-scan instructions EXTEST, INTEST, and BYPASS, refer to the IEEE Standard. The user-defined registers (USER1/USER2) are described in “USER1, USER2 Registers,” page 325.

Boundary-Scan Architecture

Virtex-II Pro device registers include all registers required by the IEEE 1149.1 Standard. In addition to the standard registers, the family contains optional registers for simplified testing and verification (Table 4-13).

Table 4-13: Virtex-II Pro JTAG Registers

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Register Length</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instruction register</td>
<td>Device specific</td>
<td>Holds current instruction OPCODE and captures internal device status.</td>
</tr>
<tr>
<td>Boundary scan register</td>
<td>3 bits per I/O</td>
<td>Controls and observes input, output, and output enable.</td>
</tr>
<tr>
<td>Bypass register</td>
<td>1 bit</td>
<td>Device bypass.</td>
</tr>
<tr>
<td>Identification register</td>
<td>32 bits</td>
<td>Captures device ID.</td>
</tr>
<tr>
<td>JTAG configuration</td>
<td>64 bits</td>
<td>Allows access to the configuration bus when using the CFG_IN or CFG_OUT instructions.</td>
</tr>
<tr>
<td>USERCODE register</td>
<td>32 bits</td>
<td>Captures user-programmable code.</td>
</tr>
</tbody>
</table>

Boundary-Scan Register

The test primary data register is the boundary-scan register. Boundary-scan operation is independent of individual IOB configurations. Each IOB, bonded or un-bonded, starts as bidirectional with 3-state control. Later, it can be configured to be an input, output, or 3-state only. Therefore, three data register bits are provided per IOB (Figure 4-23).

When conducting a data register (DR) operation, the DR captures data in a parallel fashion during the CAPTURE-DR state. The data is then shifted out and replaced by new data during the SHIFT-DR state. For each bit of the DR, an update latch is used to hold the input
data stable during the next SHIFT-DR state. The data is then latched during the UPDATE-DR state when TCK is Low.

The update latch is opened each time the TAP Controller enters the UPDATE-DR state. Care is necessary when exercising an INTEST or EXTEST to ensure that the proper data has been latched before exercising the command. This is typically accomplished by using the SAMPLE/PRELOAD instruction.

Consider internal pull-up and pull-down resistors when developing test vectors for testing opens and shorts. The boundary-scan mode determines if the IOB has a pull-up resistor. Figure 4-23 is a representation of Virtex-II Pro Boundary-Scan Architecture.

![Virtex Series Boundary Scan Logic](image)

**Figure 4-23: Virtex Series Boundary Scan Logic**

### Bit Sequence

The order in each non-TAP IOB is described in this section. The input is first, then the output, and finally the 3-state IOB control. The 3-state IOB control is closest to the TDO. The input-only pins contribute only the input bit to the boundary-scan I/O data register. The bit sequence of the device is obtainable from the “Boundary-Scan Description Language Files” (BSDL files) for the Virtex family. These files can be obtained from the Xilinx software download area. The bit sequence is independent of the design. It always has the same bit order and the same number of bits.

### Bypass Register

The other standard data register is the single flip-flop BYPASS register. It passes data serially from the TDI pin to the TDO pin during a bypass instruction. This register is initialized to zero when the TAP controller is in the CAPTURE-DR state.
Instruction Register

The instruction register loads the OPCODE necessary for the Virtex-II Pro boundary-scan instruction set. This register loads the current OPCODE and captures internal device status. The width of this register is device size specific.

Configuration Register (Boundary-Scan)

The configuration register is a 64-bit register. This register allows access to the configuration bus and readback operations.

Identification Register

Virtex devices have a 32-bit identification register, commonly referred to as the IDCODE register. This register is based upon IEEE Standard 1149.1 and allows easy identification of the part being tested or programmed via boundary scan.

Virtex-II Pro Identification Register

The Virtex-II Pro JTAG ID Code register has the following format.

```
3322 2222222 211111111 110000000000
1098 7654321 098765432 109876543210
vvvv:fffffff:aaaaaaaaa:ccccccccccc1
```

where

- \( v \) is the revision code
- \( f \) is the 7-bit family code, 0001001 for Virtex-II Pro
- \( a \) is the number of array rows plus columns in the part, expressed in 9 bits:
  - XC2VP2 rows + columns = 16 + 22 = 38 = 0x026
  - XC2VP4 rows + columns = 40 + 22 = 62 = 0x03E
  - XC2VP7 rows + columns = 40 + 34 = 74 = 0x04A
  - XC2VP20 rows + columns = 56 + 46 = 102 = 0x066
  - XC2VP30 rows + columns = 80 + 46 = 126 = 0x07E
  - XC2VP40 rows + columns = 88 + 58 = 146 = 0x092
  - XC2VP50 rows + columns = 88 + 70 = 158 = 0x09E
  - XC2VP70 rows + columns = 104 + 82 = 186 = 0x0BA
  - XC2VP100 rows + columns = 120 + 94 = 214 = 0x0D6

- \( c \) is the company code = 00001001001. Since the last bit of the JTAG IDCDE is always one, the last three hex digits appear as 0x093.

Examples of how the binary digits translate into hex codes appear in Table 4-14.
ID Codes assigned to Virtex-II Pro FPGAs are shown in Table 4-15.

### Table 4-14: Example JTAG IDCODE Concatenation

<table>
<thead>
<tr>
<th>FPGA</th>
<th>IDCODE(1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC2VP7</td>
<td>(&lt;v\rangle1226093&lt;\v&gt;)</td>
</tr>
<tr>
<td>XC2VP40</td>
<td>(&lt;v\rangle1292093&lt;\v&gt;)</td>
</tr>
<tr>
<td>XC2VP7</td>
<td>(&lt;v\rangle123E093&lt;\v&gt;)</td>
</tr>
<tr>
<td>XC2VP50</td>
<td>(&lt;v\rangle129E093&lt;\v&gt;)</td>
</tr>
<tr>
<td>XC2VP7</td>
<td>(&lt;v\rangle124A093&lt;\v&gt;)</td>
</tr>
<tr>
<td>XC2VP70</td>
<td>(&lt;v\rangle12BA093&lt;\v&gt;)</td>
</tr>
<tr>
<td>XC2VP20</td>
<td>(&lt;v\rangle1266093&lt;\v&gt;)</td>
</tr>
<tr>
<td>XC2VP30</td>
<td>(&lt;v\rangle127E093&lt;\v&gt;)</td>
</tr>
</tbody>
</table>

#### Notes:
1. The \(<v\rangle\) in the IDCODE stands for the device’s revision code (in hex).

### USERCODE Register

USERCODE is supported in the Virtex family as well. This register allows a user to specify a design-specific identification code. The USERCODE can be programmed into the device and read back for verification at a later time. The USERCODE is embedded into the bitstream during bitstream generation (bitgen -g UserID option) and is valid only after configuration.

### USER1, USER2 Registers

The USER1 and USER2 registers are only valid after configuration. These two registers must be defined by the user within the design. These registers can be accessed after they are defined by the TAP pins.

The BSCAN_VIRTEX2 library macro is required when creating these registers. This symbol is only required for driving internal scan chains (USER1 and USER2). The BSCAN_VIRTEX2 macro provides two user pins (SEL1 and SEL2) for determining usage of USER1 or USER2 instructions respectively. For these instructions, two corresponding pins (TDO1 and TDO2) allow user scan data to be shifted out of TDO. In addition, there are individual clock pins (DRCK1 and DRCK2) for each user register. There is a common input pin (TDI) and shared output pins that represent the state of the TAP controller (RESET, SHIFT, and UPDATE). Unlike earlier FPGA families that required the BSCAN macro to dedicate TAP pins for boundary scan, Virtex-II Pro TAP pins are dedicated and do not require the BSCAN_VIRTEX2 macro for normal boundary-scan instructions or operations.
Note that these are user-defined registers. The example (Figure 4-24) is one of many implementations. For HDL, the BSCAN_VIRTEX2 macro needs to be instantiated in the design.

Figure 4-24: BSCAN_VIRTEX2 (Example Usage)
Using Boundary Scan in Virtex-II Pro Devices

Some of the most commonly requested timing parameters are illustrated in Figure 4-25. For corresponding specifications, see the Virtex-II Pro / Virtex-II Pro X Data Sheet.

![Virtex-II Pro Boundary Scan Port Timing Waveforms](image)

Figure 4-25: Virtex-II Pro Boundary Scan Port Timing Waveforms

For further information on the Startup sequence, bitstream, and internal configuration registers referenced here, refer to “Readback,” page 365.

Configuring Through Boundary-Scan

One of the most common boundary-scan vendor-specific instructions is the configure instruction. An individual Virtex-II Pro device is configured via JTAG on power-up using TAP. If the Virtex-II Pro device is configured on power-up, it is advisable to tie the mode pins to the boundary-scan configuration mode settings; 101 (M2 = 1, M1 = 0, M0 = 1).

Configuration flow for Virtex-II Pro device configuration with JTAG is shown in Figure 4-26. The sections that follow describe how the Virtex-II Pro device can be configured as a single device via boundary-scan or as part of a multiple-device scan chain.

A configured device can be reconfigured by toggling the TAP and entering a CFG_IN instruction after pulsing the PROG_B pin or issuing the shut-down sequence. (Refer to “Power Up,” page 292). For additional details on power-up or the start-up sequence in Virtex-II Pro devices, see “Device Startup,” page 293.

Customers who wish to implement the Virtex-II Pro JTAG Configuration algorithm are encouraged to use the SVF-based flow provided in Xilinx Application Note XAPP058.

Single Device Configuration

Configure a Virtex-II Pro part as a single device via boundary-scan operations as follows. Ensure that the bitstream is generated with the JTAG clock option:

```
bitgen -g startupclk:jtagclk <designName>.ncd
```

Also, when using iMPACT software, verify that the most current version is being used.

Table 4-16 describes the TAP controller commands required to configure a Virtex-II Pro device. Refer to Figure 4-22, page 321 for TAP controller states. These TAP controller commands are issued automatically if configuring the part with the iMPACT software.
Chapter 4: Configuration

Figure 4-26:  Device Configuration Flow Diagram

(1) The JTAG interface becomes available after configuration memory is cleared. It is possible to shift in JTAG instructions for boundary scan operation, or configure the device through JTAG, while holding INIT_B Low to prevent the sampling of the Mode pins.

Power Up

VCCI > 1.0 V

Yes

No

Keep Clearing Configuration Memory

Yes

PROG_B Low?

Clear Configuration Memory Once More

JTAG Available(1)

INIT_B = High?

Yes

No

Sample Mode Pins

Load CFG_IN Instruction

Load Bitstream

CRC Correct?

No

Abort Startup

Yes

Synchronous TAP Reset

(Clock five 1's on TMS)

Load JSTART Instruction

Start-up Sequence

Operational

Reconfigure?

Yes

No

Shutdown Sequence

Load JShutdown Instruction
### Table 4-16: Single Device Configuration Sequence

<table>
<thead>
<tr>
<th>TAP Controller Step and Description</th>
<th>Set &amp; Hold</th>
<th># of Clocks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td><strong>TDI</strong></td>
<td><strong>TMS</strong></td>
</tr>
<tr>
<td>1. On power-up, place a logic “one” on the TMS and clock the TCK five times. This ensures starting</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>in the TLR (Test-Logic-Reset) state.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2. Move into the RTI state.</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>3. Move into the SELECT-IR state.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>4. Enter the SHIFT-IR state.</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>5. Start loading the CFG_IN instruction, LSB first:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XC2VP2 $\rightarrow$ XC2VP7:</td>
<td>111_000101</td>
<td>0</td>
</tr>
<tr>
<td>XC2VP20 $\rightarrow$ XC2VP100:</td>
<td>111_1111_000101</td>
<td>0</td>
</tr>
<tr>
<td>6. Load the MSB of CFG_IN instruction when exiting SHIFT-IR, as defined in the IEEE standard.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>7. Enter the SELECT-DR state.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>8. Enter the SHIFT-DR state.</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>9. Shift in the Virtex-II Pro bitstream. Bit$_n$ (MSB) is the first bit in the bitstream$^{(1)}$.</td>
<td>bit$_1$...bit$_n$</td>
<td>0</td>
</tr>
<tr>
<td>10. Shift in the last bit of the bitstream. Bit$_0$ (LSB) shifts on the transition to EXIT1-DR.</td>
<td>bit$_0$</td>
<td>1</td>
</tr>
<tr>
<td>11. Enter UPDATE-DR state.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>12. Reset TAP by clocking five 1’s on TMS</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>13. Enter the SELECT-IR state.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>14. Move to the SHIFT-IR state.</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>15. Start loading the JSTART instruction. The JSTART instruction initializes the startup sequence.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>XC2VP2 $\rightarrow$ XC2VP7:</td>
<td>111_001100</td>
<td>0</td>
</tr>
<tr>
<td>XC2VP20 $\rightarrow$ XC2VP100:</td>
<td>111_1111_001100</td>
<td>0</td>
</tr>
<tr>
<td>16. Load the last bit of the JSTART instruction.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>17. Move to the UPDATE-IR state.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>18. Move to RTI and clock the STARTUP sequence by applying a minimum of 12 clock cycles to the TCK.</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>19. Move to the TLR state. The device is now functional.</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

**Notes:**

1. In the Configuration Register, data is shifted in from the right (TDI) to the left (TDO).
Multiple Device Configuration

It is possible to configure multiple Virtex-II Pro devices in a chain (Figure 4-27). The devices in the JTAG chain are configured one at a time. The multiple device configuration steps can be applied to any size chain. Ensure the bitstream is generated with the JTAG clock option:

```
bitgen -g startupclk:jtagclk <designName>.ncd
```

Refer to the State Diagram in Figure 4-22 for the following TAP controller steps.

1. On power-up, place a logic 1 on the TMS and clock the TCK 5 times. This ensures starting in the TLR (Test-Logic-Reset) state.
2. Load the CFG_IN instruction into the target device (and BYPASS in all other devices). Go through RTI (RUN-TEST/IDLE).
3. Load in the configuration bitstream per step 7 through step 11 in Table 4-16.
4. Repeat step 2 and step 3 for each device.
5. Reset all TAPs by clocking five 1’s on TMS.
6. Load the JSTART command into all devices.
7. Go to RTI and clock TCK 12 times.
   All devices are active at this point.

```
Figure 4-27:  Boundary Scan Chain of Devices
```

**Notes:**

1. PROG_B pin should be deasserted during JTAG operation.
Reconfiguring Through Boundary Scan

The ability of Virtex-II Pro devices to perform partial reconfiguration is the reason that the configuration memory is not cleared when reconfiguring the device. When reconfiguring a chain of devices, refer to step 3 in Table 4-16. There are two methods to reconfigure Virtex-II Pro devices without possible internal contention. The first method is to pulse the PROG_B pin which resets the internal configuration memory. The alternate method is to perform a shutdown sequence, placing the device in a safe state. The following shutdown sequence includes using internal registers. (For details on internal registers, refer to “Readback,” page 365.)

1. Load the CFG_IN instruction.
2. In SHIFT-DR state, load the synchronization word followed by the Reset CRC Register (RCRC) command.
   1111 1111 1111 1111 1111 1111 1111 1111 → Pad word
   1010 1010 1001 1001 0101 0101 0110 0110 → Synchronization word
   0011 0000 0000 0000 1000 0000 0000 0001 → Header: Write to CMD register
   0000 0000 0000 0000 0000 0000 0000 0000 0111 → RCRC command
   0000 0000 0000 0000 0000 0000 0000 0000 → flush pipe
   0000 0000 0000 0000 0000 0000 0000 0000 → flush pipe
3. Load JSHUTDOWN.
4. Go to RTI and clock TCK at least 12 times to clock the shutdown sequence.
5. Proceed to SHIFT-IR state and load the CFG_IN instruction again.
6. Go to SHIFT-DR state and load the configuration bits. Make sure the configuration bits contain AGHIGH command, which asserts the global signal GHIGH_B. This prevents contention while writing configuration data.
   0011 0000 0000 0000 1000 0000 0000 0001 → Header: Write to CMD
   0000 0000 0000 0000 0000 0000 0000 1000 → AGHIGH command asserts GHIGH_B
3. Load JSHUTDOWN.
7. When all configuration bits have been loaded, reset the TAP by clocking five 1’s on TMS.
8. Go to SHIFT-IR state and load the JSTART instruction.
9. Go to RTI and clock TCK at least 12 times to clock the startup sequence.
10. Go to TLR state to complete the reconfiguration process.

Debugging Configuration

To verify successful configuration, there are several options. Some of the most helpful verification steps include using TAP pins and the readback command. Using the Virtex-II Pro TAP controller and status pins is discussed first.

When using TAP controller pins, TDO is driven only in the SHIFT-DR and SHIFT-IR state. If the output of the TDO can be changed via an external pull-up resistor, the TAP is not in SHIFT-IR or SHIFT-DR. If the TAP can be controlled precisely, use this to test the application.

In JTAG configuration, the status pin (DONE) functions the same as in the other configuration modes. The DONE pin can be monitored to determine if a bitstream has been completely loaded into the device. If DONE is Low, the entire bitstream has not been sent or the start-up sequence is not finished. If DONE is High, the entire bitstream has been received correctly. The INIT_B pin functions similar to a normal INIT_B but does not indicate a configuration error in boundary-scan configuration.

In addition to external pin monitoring, an internal test can be conducted. The second method includes the following steps to capture the internal device status register contents:
Chapter 4: Configuration

1. Move the TAP to TLR state.

2. Go to SHIFT-IR state and load in the CFG_IN instruction.

3. Go to SHIFT-DR state and shift in the following 64-bit pattern with the MSB (left-most bit), shifted in first.

   \[1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ 1111 \ → \text{Pad word}\]
   \[1010 \ 1010 \ 1001 \ 1001 \ 0101 \ 0101 \ 0110 \ 0110 \ → \text{Synchronization word}\]
   \[0010 \ 1000 \ 0000 \ 0000 \ 1110 \ 0000 \ 0000 \ 0010 \ → \text{Read STATus Register}^{(1)}\]
   \[0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ → \text{flush pipe}\]
   \[0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ → \text{flush pipe}\]
   \[0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ → \text{flush pipe}\]

**Notes:**

1. Since an odd number of words (1) is sent to the device for this JTAG read operation, it must end with exactly three pad words. This is because all JTAG read command sequences must consist of an even number of words (see “Packet Processor,” page 341 for details).

4. After shifting in the pattern, load the CFG_OUT instruction in the SHIFT-IR state.

5. Move to SHIFT-DR state and clock TCK 32 times while reading TDO. The data seen on TDO is the content of the status register. The last bit out is a one if a CRC error occurred. If successful, it should read as follows.

   \[0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 1\text{MMM} \ 1110 \ 11101^{(1,2)}\]

**Notes:**

1. \(\text{MMM}\) is the mode pins value.
2. Assuming that the device is in normal operation mode.
3. Refer to “Status Register (STAT),” page 347 for a detailed explanation of the bits in the Status Register.

6. Go to SHIFT-IR state and load the CFG_IN instruction again.

7. Move to SHIFT-DR state and shift in the following bit pattern:

   \[0011 \ 0000 \ 0000 \ 0000 \ 1000 \ 0000 \ 0000 \ 0000 \ 0001 \ → \text{Header: Write to CMD register}\]
   \[0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0111 \ → \text{RCRC command}\]
   \[0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ → \text{flush pipe}\]
   \[0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ 0000 \ → \text{flush pipe}\]

8. Put the TAP in TLR state when finished.

The device status register also gives the status of the DONE and INIT_B signals. For information on the status register, refer to Figure 4-38, page 347.
Once the device is powered up, it goes to an Unprogrammed state. The I/Os are all either 3-stated or pulled up. When ISC_ENABLE is successfully executed, the ISC_Enabled signal is asserted, and the device moves to ISC_Accessed state. When the device moves to ISC_Accessed state from Operational state, the shutdown sequence is executed. The I/Os are all either 3-stated or pulled up.

The StartUp sequence is executed when in the ISC_Accessed state. At the end of the StartUp Sequence, ISC_Enabled is cleared and the device moves to ISC_Complete. The minimum clock cycle requirement is the number of clock cycles required to complete the StartUp sequence. At the completion of the minimum required clock cycles, ISC_Enabled is deasserted.

Whether the StartUp sequence is successful or not is determined by CRC or configuration error status from the configuration processor. If the startup is completed, ISC_Done is asserted; otherwise, ISC_Done stays Low. The I/Os are either 3-stated or pulled up.

When ISC_Done is set in ISC_Complete state, the device moves to the Operational state. Otherwise, if ISC_Done is clear, the device moves to an Unprogrammed state. However, if the TAP controller goes to TLR state while the device is in ISC_Accessed state and if ISC_Done is set, then the device moves to the Operational state. However, the I/O is not active yet because the Startup sequence has not been performed. The Startup sequence has to be performed in the Operational state to bring the I/O active.
Clocking Startup and Shutdown Sequence (JTAG Version)

There are three clock sources for Startup and Shutdown sequence, CCLK, UserCLK, and JTAGCLK. Clock selection is set by bitgen. The Startup sequence is executed in ISC_Accessed state. When it is clocked by JTAGCLK, the Startup sequence receives the JTAGCLK in TAP Run/Test Idle state while ISC_DISABLE is the current JTAG instruction. The number of clock cycles in Run/Test Idle state for successful completion of ISC_DISABLE is determined by the number of clock cycles needed to complete the Startup sequence.

When UserCLK or CCLK is used to clock the Startup sequence, the user should know how many JTAGCLK cycles should be spent in Run/Test Idle to successfully complete the Startup sequence.

The Shutdown sequence is executed when the device transitions from an Operational to ISC_Accessed state. Shutdown is done while executing the ISC_ENABLE instruction. When the Shutdown sequence is clocked using JTAGCLK, the clock is supplied in the Run/Test Idle state of the ISC_ENABLE instruction. The number of clock cycles in Run/Test Idle is determined by the number of clock cycles needed to complete the Shutdown sequence.

When the Shutdown sequence is clocked by CCLK or UserCLK, the user is responsible for knowing how many JTAGCLK cycles in Run/Test Idle are needed to complete the Shutdown sequence.

Notes:
1. It has been decided that when configuring the device through JTAG, the startup and shutdown clock should come from TCK, regardless of the selection in bitgen.
2. In IEEE 1532 configuration mode, Startup and Shutdown clock source is always TCK.
Configuration Flows Using JTAG

(1) The JTAG interface becomes available after configuration memory is cleared. It is possible to shift in JTAG instructions for boundary scan operation, or configure the device through JTAG, while holding INIT_B Low to prevent the sampling of the Mode pins.

Figure 4-29: IEEE 1532 Configuration Flow
Chapter 4: Configuration

Figure 4-30: Signal Diagram for Successful First Time ISC Configuration

Figure 4-31: Signal Diagram for Successful ISC Partial and Full Reconfiguration
Configuration Details

All user programmable features inside a Virtex-II Pro device are controlled by memory cells that are volatile and must be configured on power-up. These memory cells are known as the configuration memory, and define the LUT equations, signal routing, IOB voltage standards, and all other aspects of the user design.

To program configuration memory, instructions for the configuration control logic and data for the configuration memory are provided in the form of a bitstream, which is delivered to the device through one of the JTAG, SelectMAP, or Serial configuration interfaces. The composition of the bitstream is largely independent of the configuration method; however, certain operations such as readback can only be performed via the SelectMAP and JTAG interfaces.

Configuration Memory: Columns and Frames

Virtex-II Pro configuration memory is arranged in vertical frames that are one bit wide and stretch from the top edge of the device to the bottom. These frames are the smallest addressable segments of the Virtex-II Pro configuration memory space; therefore, all operations must act on whole configuration frames. Configuration memory frames do not directly map to any single piece of hardware; rather, they configure a narrow vertical slice of many physical resources.

The length of a Virtex-II Pro frame depends on the size of the device, as shown in Table 4-17.

### Table 4-17: Virtex-II Pro Frame Count, Frame Length, and Bitstream Size

<table>
<thead>
<tr>
<th>Device</th>
<th>Frames</th>
<th>Frame Length (32-bit Words)</th>
<th>Configuration Bits (1)</th>
<th>Default Bitstream Size (2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC2VP2</td>
<td>884</td>
<td>46</td>
<td>1,301,248</td>
<td>1,305,376</td>
</tr>
<tr>
<td>XC2VP4</td>
<td>884</td>
<td>106</td>
<td>2,998,528</td>
<td>3,006,496</td>
</tr>
<tr>
<td>XC2VP7</td>
<td>1,320</td>
<td>106</td>
<td>4,477,440</td>
<td>4,485,408</td>
</tr>
<tr>
<td>XC2VP20</td>
<td>1,756</td>
<td>146</td>
<td>8,204,032</td>
<td>8,214,560</td>
</tr>
<tr>
<td>XC2VPX20</td>
<td>1,756</td>
<td>146</td>
<td>8,204,032</td>
<td>8,214,560</td>
</tr>
<tr>
<td>XC2VP30</td>
<td>1,756</td>
<td>206</td>
<td>11,575,552</td>
<td>11,589,920</td>
</tr>
<tr>
<td>XC2VP40</td>
<td>2,192</td>
<td>226</td>
<td>15,852,544</td>
<td>15,868,192</td>
</tr>
<tr>
<td>XC2VP50</td>
<td>2,628</td>
<td>226</td>
<td>19,005,696</td>
<td>19,021,344</td>
</tr>
<tr>
<td>XC2VP70</td>
<td>3,064</td>
<td>266</td>
<td>26,080,768</td>
<td>26,098,976</td>
</tr>
<tr>
<td>XC2VPX70</td>
<td>3,064</td>
<td>266</td>
<td>26,080,868</td>
<td>26,098,976</td>
</tr>
<tr>
<td>XC2VP100</td>
<td>3,500</td>
<td>306</td>
<td>34,272,000</td>
<td>34,292,768</td>
</tr>
</tbody>
</table>

**Notes:**
1. Number of configuration bits = (Number of Frames) x (Frame Length)
2. Default bitstream size = (Configuration bits) + Overhead

Virtex-II Pro devices do not "expect" a bitstream of any particular size. The length of the bitstream can vary according to Bitgen settings.
Configuration frames are grouped into six column types that correspond roughly to physical device resources. All Virtex-II Pro devices have the same configuration column types: IOB, IOI, CLB, GCLK, BlockRAM, and BlockRAM Interconnect.

All Virtex-II Pro devices have the same number of IOB, IOI, and GCLK columns. However, the number of CLB, BRAM, and BRAM Interconnect columns varies by device size. The number of frames per column type remains constant for all devices.

Table 4-18 shows the number of columns per device and the number of frames per column.

### Table 4-18: Virtex-II Pro Column Types and Frame Counts

<table>
<thead>
<tr>
<th>Column Type: →</th>
<th>IOB</th>
<th>IOI</th>
<th>CLB</th>
<th>BRAM</th>
<th>BRAM Interconnect</th>
<th>GCLK</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Columns per Device:</td>
<td>Frames per Column:</td>
<td>Columns per Device:</td>
<td>Frames per Column:</td>
<td>Columns per Device:</td>
<td>Frames per Column:</td>
</tr>
<tr>
<td>Device: ↓</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XC2VP2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>22</td>
<td>22</td>
<td>4</td>
</tr>
<tr>
<td>XC2VP4</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>22</td>
<td>22</td>
<td>4</td>
</tr>
<tr>
<td>XC2VP7</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>22</td>
<td>34</td>
<td>6</td>
</tr>
<tr>
<td>XC2VP20</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>22</td>
<td>46</td>
<td>8</td>
</tr>
<tr>
<td>XC2VPX20</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>22</td>
<td>46</td>
<td>8</td>
</tr>
<tr>
<td>XC2VP30</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>22</td>
<td>46</td>
<td>8</td>
</tr>
<tr>
<td>XC2VP40</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>22</td>
<td>58</td>
<td>10</td>
</tr>
<tr>
<td>XC2VP50</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>22</td>
<td>70</td>
<td>12</td>
</tr>
<tr>
<td>XC2VP70</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>22</td>
<td>82</td>
<td>14</td>
</tr>
<tr>
<td>XC2VPX70</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>22</td>
<td>82</td>
<td>14</td>
</tr>
<tr>
<td>XC2VP100</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>22</td>
<td>94</td>
<td>16</td>
</tr>
</tbody>
</table>

**IOB Columns**

IOB columns configure the voltage standard for the I/Os on the left and right edges of the device. IOBs on the top and bottom edges of the device are configured by the CLB Columns with which they are vertically aligned. There are two IOB columns per device.

**IOI Columns**

IOI columns configure the IOB registers, multiplexers, and 3-state buffers in the IOBs on the left and right edges of the device. IOBs on the top and bottom edges of the device are configured by the CLB columns with which they are vertically aligned. There are two IOI columns per device.

**CLB Columns**

The CLB columns program the configurable logic blocks, routing, and most interconnect. IOBs on the top and bottom edges of the device are also programmed by CLB configuration columns. The number of CLB configuration columns matches the number of physical CLB columns in the device.
BlockRAM Columns

BlockRAM configuration columns program only the BlockRAM user memory space. BlockRAMs are accessed by the configuration control logic in the same way that user designs access BlockRAMs: via the address and data pins that are available on the BlockRAM primitive. Consequently, the user design is not able to access BlockRAM while BlockRAM columns are being addressed by the configuration logic. For this reason, active reconfiguration and readback on BlockRAMs should not be attempted. The number of BlockRAM configuration columns matches the number of physical BlockRAM columns in the device.

BlockRAM Interconnect Columns

BlockRAM Interconnect columns program all other BlockRAM and multiplier features, including aspect ratios. The number of BlockRAM Interconnect configuration columns matches the number of physical BlockRAM columns in the device.

GCLK Column

The global clock column configures most global clock resources, including clock buffers and DCMs. There is one global clock column per device.

PPC405 and MGT Cores

XC2VP4 and larger devices feature one or more PPC405 cores, and the Virtex-II Pro X devices (XC2VPX20 and XC2VPX70) feature both PPC405 and Multi-Gigabit Transceiver (MGT) cores. In all devices, the PPC405 cores span two BRAM columns, two BRAM Interconnect columns, and several CLB columns, replacing some of the components that would normally be found in these columns. Each MGT core replaces two BRAMs in a single BRAM column.

The PPC405 core has no configuration bits, although there are several static inputs (constant ‘1’ or ‘0’) that are driven from configurable connections in adjacent BRAM Interconnect columns. These configuration bits are automatically set by Bitgen for designs that use the PPC405 core; the user need take no special action to accommodate them. The MGT core has around 300 configuration bits that are located in the BRAM Interconnect column. Like all other configuration bits, these bits are set by Bitgen according to the user design.

The configuration bits for PPC405 and MGT cores are included in the bit counts in Table 4-17.

Configuration Memory Addressing

Each configuration frame has a unique 32-bit address that is composed of a block address (BA), a major address (MJA), a minor address (MNA), and a byte number, shown in Figure 4-32. The major address identifies a specific column within a block, and the minor address identifies a specific frame within a column.

The byte number is used only by the Virtex-II Pro configuration logic (the frame length counter, described later). Users cannot address a byte within a frame, and must always set the nine least significant bits of the frame address to ‘0’.
Virtex-II Pro configuration memory is divided into three independently addressable blocks:

- Block Address 0 (BA 00) contains all GCLK, IOB, IOI, and CLB configuration columns
- Block Address 1 (BA 01) contains all BRAM columns
- Block Address 2 (BA 10) contains all BRAM Interconnect columns

Figure 4-33 illustrates the Virtex-II Pro configuration memory map. The MJA for each block type is a function of the device size, and is determined by the number of CLB columns \( n \) and the number of BlockRAM columns \( m \) in a device. During configuration, frames are programmed in order of increasing block, major, and minor address (i.e., from left to right in Figure 4-33).

### Bitstream Packets

The Virtex-II Pro bitstream is composed of a 32-bit synchronization word (0xAA995566) and numerous data packets. The purpose of the synchronization word is to align the device configuration logic with the beginning of the first packet in the bitstream. Each packet targets a specific configuration register to set configuration options, program configuration memory, or toggle internal signals.

Bitstream data packets consist of a 32-bit header and a body of variable length. There are two packet types:

- Type 1, used for smaller packets (up to \( 2^{11} - 1 \) words)
- Type 2, used for larger packets (up to \( 2^{27} - 1 \) words)

Figure 4-34 and Figure 4-35 describe the Type 1 and Type 2 packet headers.
Both packet types specify read or write access and the size of the packet in words. However, only Type 1 packet headers can specify a register address. The register set by an earlier Type 1 packet header remains active when a Type 2 packet header is processed; therefore, all Type 2 packets appear as a part of a "Type 1, Type 2" combination. Table 4-19 gives an example of a Type 1, Type 2 packet combination. Note that the register address is entirely different than the frame address described earlier.

Table 4-19:  Type 1, Type 2 Packet Combination

<table>
<thead>
<tr>
<th>Configuration Data</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x30004000</td>
<td>Type 1 Packet Header: Write 0 words to FDRI</td>
</tr>
<tr>
<td>0x50002922</td>
<td>Type 2 Packet Header: Write 10530 words to the active register (FDRI)</td>
</tr>
<tr>
<td>0x----------</td>
<td>10530 data words</td>
</tr>
</tbody>
</table>

Configuration Control Logic

The Virtex-II Pro configuration logic consists of a packet processor, a set of registers, and global signals that are controlled by the configuration registers. The packet processor controls the flow of data from the configuration interface (SelectMAP, JTAG, or Serial) to the appropriate register, while the registers control all other aspects of configuration. This section describes how bitstream packets are processed by the configuration control logic.

Packet Processor

After power-on, the packet processor waits for the synchronization word to appear on the configuration interface before interpreting any data. Once the device is synchronized, the packet processor waits for a valid packet header; when one is received it drives incoming data into the targeted configuration register until the word count specified by the packet header reaches zero. At the end of a packet the processor waits for a new packet header.

All packet headers pass through a 64-bit buffer before reaching the packet processor; the packet processor itself interprets all commands in 32-bit words. To flush the last command in a sequence from the packet buffer, a sequence of configuration commands must end with at least four 32-bit NOOP commands. Typically, this is found at the end of a bitstream, as shown in Table 4-20.
This means that read command sequences must end with exactly two or exactly three NOOPs, depending on the number of words sent prior to the NOOP commands. Do not include the synchronization word or anything prior to it in this count, and do not use any number of NOOPs other than two or three to finish sending a JTAG read command sequence. See “Configuration Register Read Procedure (JTAG),” page 369 and “Configuration Memory Read Procedure (1149.1 JTAG),” page 372 for examples.

Table 4-20: NOOP Commands at the End of the Virtex-II Pro Bitstream

<table>
<thead>
<tr>
<th>Configuration Data</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>30000001</td>
<td>Type 1 write 1 words to CRC</td>
</tr>
<tr>
<td>00009A32</td>
<td>CRC = 0x00009A32</td>
</tr>
<tr>
<td>30008001</td>
<td>Type 1 write 1 words to CMD</td>
</tr>
<tr>
<td>000000D</td>
<td>DESYNCH command</td>
</tr>
<tr>
<td>20000000</td>
<td>Type 1 NOOP word 0</td>
</tr>
<tr>
<td>20000000</td>
<td>Type 1 NOOP word 1</td>
</tr>
</tbody>
</table>

Configuration Registers

All configuration operations are executed by reading or writing to the configuration registers. Table 4-21 summarizes these registers; a detailed explanation of each register follows.

Table 4-21: Virtex-II Pro Configuration Registers

<table>
<thead>
<tr>
<th>Register Name</th>
<th>Read</th>
<th>Write</th>
<th>Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC</td>
<td>Y</td>
<td>Y</td>
<td>00000</td>
<td>CRC register</td>
</tr>
<tr>
<td>FAR</td>
<td>Y</td>
<td>Y</td>
<td>00001</td>
<td>Frame address register</td>
</tr>
<tr>
<td>FDRI</td>
<td>N</td>
<td>Y</td>
<td>00010</td>
<td>Frame data input register (write configuration data)</td>
</tr>
<tr>
<td>FDRO</td>
<td>Y</td>
<td>N</td>
<td>00011</td>
<td>Frame data output register (readback configuration data)</td>
</tr>
<tr>
<td>CMD</td>
<td>Y</td>
<td>Y</td>
<td>00100</td>
<td>Command register</td>
</tr>
<tr>
<td>CTL</td>
<td>Y</td>
<td>Y</td>
<td>00101</td>
<td>Control register</td>
</tr>
<tr>
<td>MASK</td>
<td>Y</td>
<td>Y</td>
<td>00110</td>
<td>Masking register for CTL</td>
</tr>
<tr>
<td>STAT</td>
<td>Y</td>
<td>N</td>
<td>00111</td>
<td>Status register</td>
</tr>
<tr>
<td>LOUT</td>
<td>N</td>
<td>Y</td>
<td>01000</td>
<td>Legacy output register (DOUT for daisy chain)</td>
</tr>
<tr>
<td>COR</td>
<td>Y</td>
<td>Y</td>
<td>01001</td>
<td>Configuration option register</td>
</tr>
<tr>
<td>MFWWR</td>
<td>N</td>
<td>Y</td>
<td>01010</td>
<td>Multiple frame write register</td>
</tr>
<tr>
<td>FLR</td>
<td>Y</td>
<td>Y</td>
<td>01011</td>
<td>Frame length register</td>
</tr>
<tr>
<td>KEY</td>
<td>N</td>
<td>Y</td>
<td>01100</td>
<td>Initial key address register</td>
</tr>
<tr>
<td>CBC</td>
<td>N</td>
<td>Y</td>
<td>01101</td>
<td>Initial CBC value register</td>
</tr>
<tr>
<td>IDCODE</td>
<td>Y</td>
<td>Y</td>
<td>01110</td>
<td>Device ID register</td>
</tr>
</tbody>
</table>
CRC Register (CRC)
Configuration data integrity is assured by 16-bit Cyclic Redundancy Checks (CRCs) that must be performed prior to device startup. As configuration data are written to any register other than LOUT, a CRC value is calculated using the register data and address bits. The CRC check is performed after a write to the CRC register by comparing the calculated value to the register value. If the two values differ, the configuration logic goes into the ERROR mode, indicated by driving the INIT_B pin Low and writing to the CRC_ERROR bit in the STATUS register.

CRC checks are performed in two ways:
- Explicit writes to the CRC register
- The "AutoCRC" feature

Explicit CRC checks are performed by writing a pre-calculated CRC value into the CRC register in a distinct configuration packet. Table 4-22 shows the command and data sequence for an explicit CRC check.

**Table 4-22: Example Packet for Virtex-II Pro CRC Check**

<table>
<thead>
<tr>
<th>Configuration Data</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>300000001</td>
<td>Type 1 write 1 words to CRC</td>
</tr>
<tr>
<td>00009A32</td>
<td>CRC = 0x00009A32</td>
</tr>
</tbody>
</table>

AutoCRC is an implicit CRC write that occurs at the end of a write to the Frame Data Input Register (FDRI). The last 32-bit word in a write packet to the FDRI register is automatically interpreted as a CRC value. The AutoCRC word is not included in the packet word count. Note that if the entire bitstream is written in a single Type 2 packet, only one AutoCRC check occurs. Table 4-23 shows an AutoCRC check.

**Table 4-23: Example Packets for Frame Write with AutoCRC**

<table>
<thead>
<tr>
<th>Configuration Data</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>30008001</td>
<td>Type 1 Packet Header: Write 1 word to CMD</td>
</tr>
<tr>
<td>00000001</td>
<td>Packet Data: WCFG command</td>
</tr>
<tr>
<td>30002001</td>
<td>Type 1 Packet Header: Write 1 word to FAR</td>
</tr>
<tr>
<td>02000000</td>
<td>Packet Data: Frame Address = 0x0200000000</td>
</tr>
<tr>
<td>3000401A</td>
<td>Type 1 Packet Header: Write 26 words to FDRI</td>
</tr>
<tr>
<td>00000000</td>
<td>Packet Data: word 1</td>
</tr>
<tr>
<td></td>
<td>...</td>
</tr>
<tr>
<td>00000000</td>
<td>Packet Data: word 26</td>
</tr>
<tr>
<td>0000A53B</td>
<td>AutoCRC word</td>
</tr>
</tbody>
</table>

CRC checking is disabled by setting the -g CRC:Disable option in Bitgen. This option sets the CRC_DISABLE bit in the Configuration Options Register (COR) to '1'. When CRC checking is disabled, AutoCRC and explicit CRC checks still appear, although they use the default CRC value of (0x0000DEFC), shown in Table 4-24.
Frame Address Register (FAR)

Configuration frames are addressed by the value in the Frame Address Register (FAR). Writes to the FDRI and reads from the FDRO act on the address specified in the FAR. Each time the FAR is updated with a new value, the command in the Command Register (CMD) is executed.

The FAR is automatically incremented during writes to FDRI and reads from FDRO when a number of words equal to the value in the FLR have been shifted in to FDRI or out of FDRO. Note that only the starting frame address is written before an FDRI write (“Configuration,” page 354). The frame address composition is given in Figure 4-32, page 340.

Frame Data Input Register (FDRI) and Frame Data Output Register (FDRO)

Configuration data frames are written to memory by shifting frame data into the Frame Data Input Register. Data written to the FDRI is placed in the configuration memory address specified by the FAR. Prior to writing to the FDRI, the CMD register must be loaded with the WCFG command and the correct Device ID must have been written to the IDCODE register.

The write operation is pipelined through the frame buffer such that the first frame of data is written to the configuration memory while the second frame is being shifted through the frame buffer. For this reason, the FDRI must be flushed with one frame of pad data at the end of a write to configuration memory so that the last data frame is loaded. Therefore, the smallest FDRI write operation requires two frames of data: the frame to be loaded into configuration memory, followed by a frame of pad data to flush the first out of the FDRI (see Figure 4-36). The configuration process is explained in detail in the “Configuration” section.

Configuration data frames are read from memory through the Frame Data Output Register. Prior to reading from the FDRO, the CMD register must be loaded with the RCFG command. The length of the read from the FDRO is specified by the FDRO read packet header.

Table 4-24: Example Packets for CRC/AutoCRC Checks with Checking Disabled

<table>
<thead>
<tr>
<th>Configuration Data</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>30004000</td>
<td>Type 1 write 0 words to FDRI</td>
</tr>
<tr>
<td>50002922</td>
<td>Type 2 write 10530 words to FDRI</td>
</tr>
<tr>
<td>00100000</td>
<td>Packet data write FDRI word 0</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>00000000</td>
<td>Packet data write FDRI word 10529</td>
</tr>
<tr>
<td>0000DEFC</td>
<td>Auto CRC word</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td>30000001</td>
<td>Type 1 write 1 words to CRC</td>
</tr>
<tr>
<td>0000DEFC</td>
<td>CRC = 0x0000DEFC</td>
</tr>
</tbody>
</table>
The read operation is pipelined through the frame buffer such that the first frame is driven to the configuration interface as the second frame is being read from memory (Figure 4-36). For this reason, the first frame read out of the device is not valid data; readback data begins with the second word read from the device. Because of the frame buffer, the smallest FDRO read requires two frames of data: the first frame to be clear the frame buffer, followed by the frame of configuration data. The readback process is explained in detail in the “Readback” section.

Reads from the FDRO register do not generate a CRC value, nor is an AutoCRC performed after reading from the FDRO.

**Figure 4-36: Configuration Frame Data Path**

**Command Register (CMD)**

The Command Register is used to instruct the configuration control logic to strobe global signals and perform other configuration functions. Commands are executed immediately after being written to the CMD register. Additionally, whenever there is a write to the FAR, the last command written to the CMD register will be executed again. Table 4-25 gives the Command Register commands and codes.

**Table 4-25: Command Register Codes**

<table>
<thead>
<tr>
<th>Command</th>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WCFG</td>
<td>0001</td>
<td>Write Configuration Data. Used prior to writing configuration data to the FDRI.</td>
</tr>
<tr>
<td>MFWR</td>
<td>0010</td>
<td>Multiple Frame Write Register. Used to perform a write of a single frame to multiple frame addresses.</td>
</tr>
<tr>
<td>LFRM</td>
<td>0011</td>
<td>Last Frame. Deasserts the GHIGH_B signal, activating all interconnect. The GHIGH_B signal is asserted with the AGHIGH command.</td>
</tr>
</tbody>
</table>
Control Register (CTL)

The Control Register is used to set the configuration security level, the persist setting, and to toggle the Global Three-State signal. Writes to the Control Register are masked by the value in the MASK Register, which allows the GTS_USR_B signal to be toggled without re-specifying the Security and Persist bits. The fields are illustrated in Figure 4-37 and defined in Table 4-26.
Table 4-26: Control Register Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reserved</td>
<td>Various</td>
<td>Reserved CTL register bits for testing. Always leave these bits set to 0.</td>
</tr>
</tbody>
</table>
| SBITS         | 5:4       | Security Level:  
0: Read/Write OK (default)  
1: Readback disabled  
2, 3: Readback disabled, writing disabled except to the CRC register.                       |
| PERSIST       | 3         | The configuration interface defined by M2:M0 remains after configuration. Typically used only with the SelectMAP interface to allow reconfiguration and readback.  
0: No (default)  
1: Yes |
| GTS_USER_B    | 0         | Active Low signal to place I/Os into a high-Z state.  
0: I/Os placed in high-impedance state  
1: I/Os active |

Mask Register (MASK)

The Mask Register masks writes into the Control Register. A '1' in any bit within the MASK register allows the corresponding bit position to be written in the Control Register. The default value of the MASK Register is all '0's.

Status Register (STAT)

The Status Register indicate the value of numerous global signals. The register can be read via the SelectMAP or JTAG interfaces. Figure 4-38 gives the name of each bit position in the STAT register; a detailed explanation of each bit position is given in Table 4-27.

Figure 4-38: Status Register

<table>
<thead>
<tr>
<th>BAD_KEY_SEQ</th>
<th>DEC_ERROR</th>
<th>ID_ERROR</th>
<th>DONE</th>
<th>INIT</th>
<th>MODE</th>
<th>GHIGH_B</th>
<th>GWE</th>
<th>GTS_CFG_B</th>
<th>IN_ERROR</th>
<th>DCI_MATCH</th>
<th>DCM_LOCK</th>
<th>PART_SECURED</th>
<th>CRC_ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 x x x x x x x x x x x x x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### Table 4-27: Status Register Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAD_KEY_SEQ</td>
<td>15</td>
<td>Encryption keys are stored with a tag indicating their position among the other keys. If the keys are sent in the wrong order, it will be indicated by this status bit.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: No decryptor key sequence error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Decryptor keys were not used in the correct sequence</td>
</tr>
<tr>
<td>DEC_ERROR</td>
<td>14</td>
<td>Write FDRI issued before or after decrypt operation.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: No decryptor error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Decryptor error. Assert PROG to recover</td>
</tr>
<tr>
<td>ID_ERROR</td>
<td>13</td>
<td>Attempt to write to FDRI without successful IDCODE check.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: No ID_ERROR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: ID_ERROR. Assert PROG to recover</td>
</tr>
<tr>
<td>DONE</td>
<td>12</td>
<td>The DONE bit in the STATUS register reflects the actual level on the DONE pin. If the DONE signal is released by the device but held Low externally, this bit remains Low.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: DONE pin = logic 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: DONE pin = logic 1</td>
</tr>
<tr>
<td>INIT</td>
<td>11</td>
<td>The INIT bit in the STATUS register reflects the actual level on the INIT pin.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: INIT pin = logic 0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: INIT pin = logic 1</td>
</tr>
<tr>
<td>MODE</td>
<td>10:8</td>
<td>Status of the MODE pins (M2:M0).</td>
</tr>
<tr>
<td>GHIGH_B</td>
<td>7</td>
<td>Status of the GHIGH_B signal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: GHIGH_B is asserted</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: GHIGH_B is deasserted</td>
</tr>
<tr>
<td>GWE</td>
<td>6</td>
<td>Status of the Global Write Enable signal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: All FFs and BRAMs are write disabled</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: FFs and BRAMs are write enabled</td>
</tr>
<tr>
<td>GTS_CFG_B</td>
<td>5</td>
<td>Status of the GTS_CFG_B signal.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: All I/Os in high-Z state</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: I/Os are not placed in High-Z by this signal</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note that there are other ways of forcing I/Os into a high-Z state, including GTS_USR_B, as well as through various JTAG instructions.</td>
</tr>
<tr>
<td>IN_ERROR</td>
<td>4</td>
<td>Indicates that input data is being clocked in too fast. This can happen if the CCLK frequency exceeds $F_{CC_SELECTMAP}$ in SelectMAP mode.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: No legacy input error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Legacy input error</td>
</tr>
</tbody>
</table>
Configuration Details

Table 4-27: Status Register Description (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DCI_MATCH</td>
<td>3</td>
<td>This signal reflects the logical AND of all MATCH signals. There is one MATCH signal per I/O bank. If no DCI I/Os are instantiated in a given bank, the bank’s MATCH signal = 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: DCI not matched</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: DCI is matched</td>
</tr>
<tr>
<td>DCM_LOCK</td>
<td>2</td>
<td>This is a logical AND function of all DCM LOCKED signals. Unused DCM LOCKED signals = 1.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: DCMs not locked</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: DCMs are locked</td>
</tr>
<tr>
<td>PART_SECURED</td>
<td>1</td>
<td>Status of the Triple-DES decryptor.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: Decryptor security not set</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: Decryptor security set</td>
</tr>
<tr>
<td>CRC_ERROR</td>
<td>0</td>
<td>Status of the CRC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0: No CRC error</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1: CRC error</td>
</tr>
</tbody>
</table>

Legacy Output Register (LOUT)

The Legacy Output Register is used to write data to the DOUT pin when configuring downstream devices in a serial configuration daisy-chain. The LOUT register is not available for SelectMAP and JTAG configuration modes.

Configuration Options Register (COR)

The Configuration Options Register is used to set certain configuration options for the device. The name of each bit position in the COR is given in Figure 4-39 and described in Table 4-28.

Figure 4-39: Configuration Options Register
### Table 4-28: Configuration Options Register Description

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CRC_BYPASS</td>
<td>29</td>
<td>0: CRC enabled&lt;br&gt;1: CRC disabled</td>
</tr>
<tr>
<td>SHUT_RST_DCM</td>
<td>26</td>
<td>When this bit is set, all DCMs will be reset if the AGHIGH command is issued.&lt;br&gt;0: DCMs cannot be reset through the configuration logic&lt;br&gt;1: DCMs will reset during shutdown readback or reconfiguration</td>
</tr>
<tr>
<td>DONE_PIPE</td>
<td>25</td>
<td>Adds a pipeline stage for the DONEIN signal. The DONEIN signal reflects the logic level on the DONE pin, not whether the device has released the DONE pin. This pipeline stage can be useful if the rise time on the DONE pin is slow, which could otherwise cause configuration to fail.&lt;br&gt;0: No pipeline stage for DONEIN&lt;br&gt;1: Add pipeline stage for DONEIN</td>
</tr>
<tr>
<td>DRIVE_DONE</td>
<td>24</td>
<td>By default the DONE pin is an open-drain output, meaning that the pin actively drives a logic '0' before configuration, then goes into a high-Z state when it is &quot;released&quot; during Startup. This setting requires an external pull-up resistor on the DONE pin, and is typically applied to Slave devices in a serial daisy-chain.&lt;br&gt;Alternately, the DONE pin can be configured as an actively-driven output, meaning that the pin actively drives a logic '1' when it is released.&lt;br&gt;0: DONE pin is an open-drain output&lt;br&gt;1: DONE pin is an actively driven output (released = logic '1')</td>
</tr>
<tr>
<td>SINGLE</td>
<td>23</td>
<td>The SINGLE bit determines whether the CAPTURE_VIRTEX2 primitive operates in one-shot mode or in continuous sample mode. This bit is set when the &quot;one-shot&quot; attribute is attached to the CAPTURE_VIRTEX2 primitive (see “Readback Capture,” page 380).&lt;br&gt;0: SINGLE mode disabled. In this mode, capture values are loaded on every clock cycle while CAP is asserted.&lt;br&gt;1: SINGLE mode enabled. In this mode, capture values are loaded only once, regardless of how many clock cycles the CAF signal is asserted for.</td>
</tr>
<tr>
<td>OSCFSEL</td>
<td>22:17</td>
<td>Selects the CCLK frequency for Master configuration modes (see Table 4-29).</td>
</tr>
</tbody>
</table>
### Table 4-28: Configuration Options Register Description (Continued)

<table>
<thead>
<tr>
<th>Name</th>
<th>Bit Index</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSCLKSRC</td>
<td>16:15</td>
<td>Startup sequence clock source:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00: CCLK</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01: UserClk (per connection on the CAPTURE_VIRTEX2 block)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1x: JTAGClk</td>
</tr>
<tr>
<td>DONE_CYCLE</td>
<td>14:12</td>
<td>Startup cycle to release the DONE pin:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000: Startup cycle 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>001: Startup cycle 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010: Startup cycle 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>011: Startup cycle 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100: Startup cycle 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101: Startup cycle 6</td>
</tr>
<tr>
<td>MATCH_CYCLE</td>
<td>11:9</td>
<td>Startup cycle to stall in until DCI matches:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000: Startup cycle 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>001: Startup cycle 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010: Startup cycle 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>011: Startup cycle 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100: Startup cycle 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101: Startup cycle 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>111: No Wait</td>
</tr>
<tr>
<td>LOCK_CYCLE</td>
<td>8:6</td>
<td>Startup cycle to stall in until DCMs lock:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000: Startup cycle 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>001: Startup cycle 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010: Startup cycle 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>011: Startup cycle 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100: Startup cycle 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101: Startup cycle 6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>111: No Wait</td>
</tr>
<tr>
<td>GTS_CYCLE</td>
<td>5:3</td>
<td>Startup cycle to deassert the Global Three-State (GTS) signal:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000: Startup cycle 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>001: Startup cycle 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010: Startup cycle 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>011: Startup cycle 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100: Startup cycle 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101: Startup cycle 6</td>
</tr>
<tr>
<td>GWE_CYCLE</td>
<td>2:0</td>
<td>Startup phase to deassert the Global Write Enable (GWE) signal:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>000: Startup cycle 1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>001: Startup cycle 2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>010: Startup cycle 3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>011: Startup cycle 4</td>
</tr>
<tr>
<td></td>
<td></td>
<td>100: Startup cycle 5</td>
</tr>
<tr>
<td></td>
<td></td>
<td>101: Startup cycle 6</td>
</tr>
</tbody>
</table>
Table 4-29: OSCFSEL-Specified Master CCLK Settings

<table>
<thead>
<tr>
<th>CCLK (MHz)</th>
<th>OSCFSEL (binary)</th>
<th>CCLK (MHz)</th>
<th>OSCFSEL (binary)</th>
<th>CCLK (MHz)</th>
<th>OSCFSEL (binary)</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>000010</td>
<td>13</td>
<td>001010</td>
<td>41</td>
<td>100111</td>
</tr>
<tr>
<td>5</td>
<td>010001</td>
<td>15</td>
<td>001101</td>
<td>45</td>
<td>110011</td>
</tr>
<tr>
<td>7</td>
<td>000100</td>
<td>20</td>
<td>010111</td>
<td>51</td>
<td>101010</td>
</tr>
<tr>
<td>8</td>
<td>000101</td>
<td>26</td>
<td>011010</td>
<td>55</td>
<td>110100</td>
</tr>
<tr>
<td>9</td>
<td>000110</td>
<td>30</td>
<td>011101</td>
<td>60</td>
<td>101101</td>
</tr>
<tr>
<td>10</td>
<td>000111</td>
<td>34</td>
<td>110010</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Multiple Frame Write Register (MFWR)

The Multiple Frame Write Register is used when the bitstream compression option is enabled in Bitgen (-g Compress:yes). When more than one frame has identical data, the frame data can be loaded once into the Multiple Frame Write Register then copied into multiple memory address locations. In some cases this can decrease the size of the bitstream considerably.

To write multiple frames with the same data, Bitgen constructs a bitstream with the following commands:

1. Write WCFG command to CMD register
2. Write desired frame to FDRI
3. Write to FAR register with the first desired address
4. Write MFWR command to CMD register
5. Write two pad words to the MFWR register
6. Write to FAR register with the second desired address
7. Write two pad words to MFWR
8. Repeat step 6 and step 7 until the last desired address

Table 4-30: Example Packets for Bitstream Compression Using MFWR

<table>
<thead>
<tr>
<th>Step</th>
<th>Configuration Data (hex)</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>30002001</td>
<td>Type 1 write 1 words to FAR</td>
</tr>
<tr>
<td></td>
<td>00000000</td>
<td>FAR Address = 0x00000000</td>
</tr>
<tr>
<td>2</td>
<td>30008001</td>
<td>Type 1 write 1 words to CMD</td>
</tr>
<tr>
<td></td>
<td>00000001</td>
<td>WCFG command</td>
</tr>
<tr>
<td>3</td>
<td>30004034</td>
<td>Type 1 write 52 words to FDRI</td>
</tr>
<tr>
<td></td>
<td>00000080</td>
<td>packet data write FDRI word 0</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>00000000</td>
<td>packet data write FDRI word 51</td>
</tr>
<tr>
<td></td>
<td>0000A11D</td>
<td>Auto CRC word</td>
</tr>
</tbody>
</table>
Since configuration frames are arranged vertically, designs that span the fewest possible configuration frames achieve greater compression. Identical frames are usually found in Block RAM columns that are initialized with identical values or in unused CLB or Block RAM columns. To estimate the amount of compression achieved:

- Identify the number of identical CLB or BRAM columns. CLB columns are only identical if they are unused and have no routing passing through them. BRAM columns are identical if the same initialization values are used for multiple BRAM columns.
- Calculate the number of configuration frames for the column type of interest:
  \[
  \text{# config frames} = (\text{# frames per column}) \times (\text{# of columns})
  \]
- The number of saved configuration frames is given by:
  \[
  \text{#Saved configuration frames} = (\text{#identical frames} - 1).
  \]
- A rough estimate of the number of saved configuration bits is therefore given by:
  \[
  \text{#Saved configuration bits} = (\text{# of saved configuration frames}) \times (\text{# of bits per frame})
  \]

### Frame Length Register (FLR)

Virtex-II frame lengths are not hard-coded into the device because the same configuration logic is used for all devices. The frame length must be programmed into the device before FDRI or FDRO operations can be performed.

The Frame Length Register (FLR) stores the device frame length, and must be written to before any FDRI or FDRO operation can be performed.

Note that the value written to the FLR is always one fewer than the actual number of words per frame. For example, the actual frame length of the XC2VP7 is 106 words; however the value 0x00000069, or 105, is written to the FLR.

<table>
<thead>
<tr>
<th>Step</th>
<th>Configuration Data (hex)</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>30002001 00000200</td>
<td>Type 1 write 1 words to FAR FAR Address = 0x000000200</td>
</tr>
<tr>
<td>5</td>
<td>30008001 00000002</td>
<td>Type 1 write 1 words to CMD MFWR command</td>
</tr>
<tr>
<td>6</td>
<td>30014002 00000000 00000000</td>
<td>Type 1 write 2 words to MFWR packet data write MFWR word 0 packet data write MFWR word 1</td>
</tr>
<tr>
<td>7</td>
<td>30002001 00000400</td>
<td>Type 1 write 1 words to FAR FAR Address = 0x000000400</td>
</tr>
<tr>
<td>8</td>
<td>30014002 00000000 00000000</td>
<td>Type 1 write 2 words to MFWR packet data write MFWR word 0 packet data write MFWR word 1</td>
</tr>
</tbody>
</table>

Table 4-30: Example Packets for Bitstream Compression Using MFWR (Continued)
Initial Key Address Register (KEY)

The Initial Key Address Register (KEY) indicates the number of DES keys that are used for bitstream encryption. This is a write-only register; there is no way to read this register through any configuration interface or user logic resources.

Cipher Block Chaining Register (CBC)

The Cipher Block Chaining Register stores the DES encryption keys. This is a write-only register; there is no way to read the keys out of the device through any configuration interface or user logic resources.

Device ID (IDCODE)

The Device ID (IDCODE) Register contains the device-specific identification code. This register is separate from the JTAG IDCODE register although it uses the same 32-bit identification code. The purpose of this register is to ensure that the bitstream has been created for the correct device.

After writing to the IDCODE register the value is automatically compared; if it does not match the expected IDCODE value the ID_ERROR signal is asserted, halting configuration. This prevents a Virtex-II or Virtex-II Pro bitstream that is targeted for one device from successfully programming a different Virtex-II or Virtex-II Pro device—for example, an XC2VP2 bitstream is prevented from programming an XC2VP4 device by this mechanism. A successful write to the Device ID register is required prior to writing frame data. Table 4-31 lists Virtex-II Pro IDCODEs.

Table 4-31: Virtex-II Pro IDCODEs

<table>
<thead>
<tr>
<th>Device</th>
<th>IDCODE</th>
<th>Device</th>
<th>IDCODE</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC2VP2</td>
<td>0x01226093</td>
<td>XC2VP40</td>
<td>0x01292093</td>
</tr>
<tr>
<td>XC2VP4</td>
<td>0x0123E093</td>
<td>XC2VP50</td>
<td>0x0129E093</td>
</tr>
<tr>
<td>XC2VP7</td>
<td>0x0124A093</td>
<td>XC2VP70</td>
<td>0x012BA093</td>
</tr>
<tr>
<td>XC2VP20</td>
<td>0x01266093</td>
<td>XC2VPX70</td>
<td>0x018BA093</td>
</tr>
<tr>
<td>XC2VPX20</td>
<td>0x01866093</td>
<td>XC2VP100</td>
<td>0x012D6093</td>
</tr>
<tr>
<td>XC2VP30</td>
<td>0x0127E093</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Configuration

Configuration can begin after the device is powered and initialization has finished, as indicated by the INIT pin being released. After initialization, the packet processor ignores all data presented on the configuration interface until it receives the synchronization word; after synchronization the packet processor waits for a valid packet header to begin the configuration process.

Default Initial Configuration Process

Initial configuration using a "default" bitstream (a bitstream generated using the default Bitgen settings) begins by pulsing the PROG_B pin for SelectMAP and Serial configuration modes or by issuing the JPROG_B instruction for JTAG configuration mode. Configuration proceeds as follows:
Default Initial Configuration

1. **Device Synchronization.** Virtex-II Pro devices are synchronized by sending the synchronization word (0xAA995566) to the configuration logic. The synchronization word tells the device to begin parsing packet data.

2. **Reset CRC Register.** The CRC register is reset to ensure a correct CRC calculation. See “CRC Register (CRC),” page 343 for details.

3. **Write to Frame Length Register.** Virtex-II Pro devices are not hard-coded with the correct frame length. Prior to configuration, the FLR must be written with a value equal to:

   \[(\text{number of words per frame}) - 1\]

   See “Frame Length Register (FLR),” page 353 for details.

4. **Write to Configuration Options Register** (startup sequence options, etc.). See “Configuration Options Register (COR),” page 349 for details on which configuration options are programmed into the COR.

5. **Write to Device ID Register.** A successful write to the Device ID register is required before frame data can be written. See “Device ID (IDCODE),” page 354 for details.

6. **Write to MASK register** (depends on user settings). The SBITS and PERSIST bits in the CTL register are masked prior to configuration so the GTS_USR_B signal can be pulsed as needed after configuration.

7. **Write SWITCH command to CMD register** (needed for Master configuration modes only). The SWITCH command changes the CCLK output frequency according to the Bitgen -g configrate configuration setting. This command has no effect during slave and JTAG configuration modes.

8. **Write starting Frame Address to FAR** (0x00000000). The FAR is set with the starting frame address for initial configuration. Only the starting frame address needs to be written during configuration; the FAR is automatically incremented after every frame is written to FDRI.

9. **Write WCFG command to CMD register.** The WCFG command allows frame data to be written to configuration memory.

10. **Write configuration frame data to FDRI, including 1 frame of pad data.** One frame of pad data must follow the frame data write to flush the frame buffer, as explained in “Frame Data Input Register (FDRI) and Frame Data Output Register (FDRO),” page 344. For full configuration, the length of the FDRI write is given by:

    \[\#\text{FDRI Words} = (\#\text{frames} + 1 \text{ pad frame}) \times \text{frame length in words}\]

    After the last frame of data is sent to the device, a frame of pad data is sent in the same FDRI write to push the last data frame from the Frame Buffer to the FDRI register. This pad frame is included in the packet header word count. The data content of the pad frame does not matter.

    After the FDRI write finishes, the next 32-bit word is interpreted by the device as an AutoCRC value. The AutoCRC value is not included in the packet word count.

11. **Write the GRESTORE command to the CMD register.** GRESTORE command updates all slice and IOB registers with their initial values.

12. **Write the Last Frame (LFRM) command to the CMD register.** The LFRM command deasserts the internal GHIGH signal, activating all interconnect. The GHIGH signal is first activated during initialization (i.e., after the PROG pin is pulsed), and remains activated until the LFRM command is issued.
13. **Send START command to bring device into startup sequence.** The START command begins the startup sequence for SelectMAP and Serial configuration modes, using the CCLK or USRCLK signal to clock the sequence. For JTAG the JTAG JSTART instruction must be issued so that the JTAG TCK clocks the startup sequence.

14. **Write to Control Register.** See “Control Register (CTL),” page 346 for more information.

15. **Perform final CRC check.** A CRC value is always created, even if CRC checking is disabled in Bitgen. When CRC checking is disabled, the default CRC value of 0x0000DEFC is written to the CRC register.

16. **Send DESYNC command.** The DESYNC command "desynchronizes" the configuration logic. Further configuration data are ignored until a new synchronization word is received.

17. **Write at least 64 bits of pad data (NOOP commands) to flush packet buffer.** The packet buffer must be flushed with 64 bits of packet data. For this sequence of commands, the pad bits push the DESYNC command into the packet processor.

18. **Continue sending CCLK or TCK pulses until DONE goes High.** Depending on the startup options and external conditions (for example, if the DONE pin is held Low externally), additional CCLK pulses might be needed to bring the device out of the startup sequence. Refer to “Configuring Through Boundary-Scan,” page 327 for further details on using this procedure with the JTAG configuration interface.

Each of these steps is performed by a single configuration packet except for synchronization (step 1) and the large FDRI write (step 10), which is performed by a Type 1, Type 2 packet combination. Table 4-32 shows how the default configuration steps are written in an XC2VP7 bitstream.

### Table 4-32: Default XC2VP7 Bitstream for Initial Configuration

<table>
<thead>
<tr>
<th>Step</th>
<th>Configuration Data (hex)</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AA995566</td>
<td>Sync word</td>
</tr>
<tr>
<td>2</td>
<td>30008001, 00000007</td>
<td>Type 1 write 1 words to CMD, RCRC command</td>
</tr>
<tr>
<td>3</td>
<td>30016001, 00000069</td>
<td>Type 1 write 1 words to FLR, Frame Length = 105 words (actual XC2VP7 frame length – 1 = 105 words = 0x69)</td>
</tr>
<tr>
<td>4</td>
<td>30012001</td>
<td>Type 1 write 1 words to COR</td>
</tr>
<tr>
<td></td>
<td>XXXXXXXXXXX</td>
<td>COR settings (dependent on user options)</td>
</tr>
<tr>
<td>5</td>
<td>3001C001, 0124A093</td>
<td>Type 1 write 1 words to ID, Device ID = 0x0124A093 (XC2VP7)</td>
</tr>
<tr>
<td>6</td>
<td>3000C001</td>
<td>Type 1 write 1 words to MASK</td>
</tr>
<tr>
<td></td>
<td>XXXXXXXXXXX</td>
<td>CTL Register MASK (dependent on user options)</td>
</tr>
<tr>
<td>7</td>
<td>30008001, 00000009</td>
<td>Type 1 write 1 words to CMD, SWITCH command</td>
</tr>
</tbody>
</table>
### Configuration Details

**Active and Shutdown Reconfiguration**

The term *reconfiguration* in this context describes the process of reprogramming a device without pulsing the PROG pin or cycling power. Reconfiguration can involve the whole device or just a portion of it, and can occur while the device is in the Shutdown state (*shutdown reconfiguration*), or while the device continues to operate (*active reconfiguration*).

#### Table 4-32: Default XC2VP7 Bitstream for Initial Configuration (Continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Configuration Data (hex)</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>30002001</td>
<td>Type 1 write 1 words to FAR</td>
</tr>
<tr>
<td></td>
<td>00000000</td>
<td>FAR Address= 0x0000000000</td>
</tr>
<tr>
<td>9</td>
<td>30008001</td>
<td>Type 1 write 1 words to CMD</td>
</tr>
<tr>
<td></td>
<td>00000000</td>
<td>WCFG command</td>
</tr>
<tr>
<td>10</td>
<td>30004000</td>
<td>Type 1 write 0 words to FDRI</td>
</tr>
<tr>
<td></td>
<td>500222FA</td>
<td>Type 2 write 140026 words to FDRI: (1320 frames + 1) x (106 words per frame) = 140026 words</td>
</tr>
<tr>
<td></td>
<td>00100000</td>
<td>Packet data write FDRI word 0</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td>Packet data write FDRI word 140025</td>
</tr>
<tr>
<td></td>
<td>XXXXXXXX</td>
<td>Auto CRC word (design dependent)</td>
</tr>
<tr>
<td>11</td>
<td>30008001</td>
<td>Type 1 write 1 words to CMD</td>
</tr>
<tr>
<td></td>
<td>00000000A</td>
<td>GRESTORE command</td>
</tr>
<tr>
<td>12</td>
<td>30008001</td>
<td>Type 1 write 1 words to CMD</td>
</tr>
<tr>
<td></td>
<td>00000003</td>
<td>LFRM command</td>
</tr>
<tr>
<td>13</td>
<td>30008001</td>
<td>Type 1 write 1 words to CMD</td>
</tr>
<tr>
<td></td>
<td>00000005</td>
<td>START command</td>
</tr>
<tr>
<td>14</td>
<td>3000A001</td>
<td>Type 1 write 1 words to CTL</td>
</tr>
<tr>
<td></td>
<td>XXXXXXXX</td>
<td>CTL settings (dependent on user options)</td>
</tr>
<tr>
<td>15</td>
<td>30000001</td>
<td>Type 1 write 1 words to CRC</td>
</tr>
<tr>
<td></td>
<td>XXXXXXXX</td>
<td>CRC (design dependent)</td>
</tr>
<tr>
<td>16</td>
<td>30008001</td>
<td>Type 1 write 1 words to CMD</td>
</tr>
<tr>
<td></td>
<td>0000000D</td>
<td>DESYNCH command</td>
</tr>
<tr>
<td>17</td>
<td>20000000</td>
<td>Type 1 NOOP word 0</td>
</tr>
<tr>
<td></td>
<td>20000000</td>
<td>Type 1 NOOP word 1</td>
</tr>
<tr>
<td>18</td>
<td>XXXXXXXX</td>
<td>Continue sending CCLK or TCK pulses until DONE goes High.</td>
</tr>
</tbody>
</table>
Since the user design continues to operate during active reconfiguration, this style of reconfiguration must be used with caution. Internal contention can occur if a device is overwritten with a new design through active reconfiguration, potentially damaging the device. Furthermore, all SRL16, Distributed RAM, and BlockRAM contents are reinitialized by a full bitstream, clearing any data written into these memory elements by the user design.

To bring the device into the Shutdown state through the SelectMAP interface, the configuration logic is synchronized and the SHUTDOWN command is written to the CMD register (Table 4-33). To bring the device into the Shutdown state through the JTAG interface, the JTAG JSHUTDOWN command is issued and the TAP controller is brought to the RTI state for 12 TCK cycles.

The Shutdown sequence runs the startup sequence in reverse:

- Deassert EOS
- Pull the DONE pin Low
- Assert GTS, placing all I/Os in a high-Z state (except for persistent configuration pins)
- De-assert GWE, preventing RAMs and flip-flops from changing state

Neither the start-up sequence nor the shutdown sequence can finish if the CRC_ERROR signal is asserted. To ensure the shutdown sequence completes successfully, it should always be preceded by the RCRC (Reset CRC) command.

To bring a device out of the shutdown state, the device must be brought through the startup sequence.  

<table>
<thead>
<tr>
<th>Configuration Data</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>30008001</td>
<td>Type 1 write 1 words to CMD</td>
</tr>
<tr>
<td>0000000B</td>
<td>SHUTDOWN command</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Description</th>
<th>Set and Hold</th>
<th># of Clocks</th>
</tr>
</thead>
<tbody>
<tr>
<td>TDI</td>
<td>TMS</td>
<td>TCK</td>
</tr>
<tr>
<td>Clock five 1s on TMS to bring the device to the TLR state</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>Move into the RTI state</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>Move into the Select-IR state</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>Move into the Shift-IR State</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>Shift the first 9 bits of the JSHUTDOWN instruction, LSB first</td>
<td>111001101</td>
<td>0</td>
</tr>
<tr>
<td>Shift the MSB of the JSHUTDOWN instruction while exiting SHIFT-IR</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Move into the Update-IR State</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>Move into the RTI state; remain there for 12 TCK cycles</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>
The procedure for shutdown reconfiguration is identical to the "default" configuration sequence described above, except that the SHUTDOWN command is issued to the device immediately after synchronization.

**Shutdown Reconfiguration Sequence**
1. Device Synchronization
2. Write SHUTDOWN command to CMD register
3. Reset CRC Register
4. Write to Frame Length Register (actual frame length – 1)
5. Write to Configuration Options Register (startup sequence options, etc.)
6. Write to Device ID Register
7. Write to MASK register (depends on user settings)
8. Write SWITCH command to CMD register (needed for Master configuration modes only)
9. Write starting Frame Address to FAR (0x00000000)
10. Write WCFG command to CMD register
11. Write configuration frame data to FDRI, including 1 frame of pad data
12. Write the GRESTORE command to the CMD register
13. Write Last Frame (LFRM) command
14. Write 1 frame of NOOP commands
15. Write START command to CMD register to begin startup sequence
16. Perform final CRC check
17. Write DESYNC command to CMD register
18. Write at least 64 bits of pad data (NOOP commands) to flush packet buffer
19. Continue sending CCLK pulses until DONE goes High

**Partial Reconfiguration**

Virtex-II Pro devices allow *partial reconfiguration*—rewriting a subset of configuration frames, either while user design is suspended ("Shutdown" partial reconfiguration) or while the user design is operating ("Active" partial reconfiguration). There are two reasons to perform partial reconfiguration:

- To change design behavior without fully reconfiguring a device
- To correct memory upsets in high-radiation environments

This document addresses the needs of commercial customers wishing to change design behavior. Aerospace customers seeking more information on the use of partial reconfiguration for correcting configuration memory upsets in high-radiation environments should contact their Xilinx FAE or Sales Representative, or open a technical support Webcase at [http://www.support.xilinx.com](http://www.support.xilinx.com).

Since Xilinx does not document the function of the configuration bits within every frame, users should combine the information in the “Partial Reconfiguration Process” section with one of the software flows described in the “Partial Reconfiguration Applications and Software Flows” section below.
Partial Reconfiguration Process

This section explains the mechanics of partially reconfiguring a single frame of configuration data. Similar to full reconfiguration, partial reconfiguration can be performed while the user design continues to operate or after placing the design in a shutdown state. Partial reconfiguration is only possible through the SelectMAP and JTAG interfaces. For SelectMAP partial reconfiguration, the "persist" setting must be enabled to retain the function of the SelectMAP pins after configuration.

Shutdown Reconfiguration Sequence

1. Device Synchronization
2. Write SHUTDOWN command to CMD register
3. Reset CRC Register
4. Write to Frame Length Register (actual frame length – 1)
5. Write to Configuration Options Register (optional)
6. Write to Device ID Register
7. Write to MASK register (depends on user settings)
8. Write Frame Address to FAR
9. Write WCFG command to CMD register
10. Write frame data to FDRI (packet size = # frames + 1 pad frame)
11. Write the default AutoCRC value of $0x0000DEFC$
12. Write the GRESTORE command to the CMD register.
13. Send Last Frame (LFRM) command
14. Send 1 frame of NOOP commands
15. Send START command to bring device into startup sequence
16. Perform final CRC check
17. Send DESYNC command
18. Send at least 64 bits of pad data to flush packet buffer
19. Continue sending CCLKs pulses until DONE goes High

Active Partial Reconfiguration Sequence (XC2VP4 or XC2VP7)

1. Synchronize device
2. Reset CRC
3. Write IDCODE register
4. Write Frame Length to FLR (actual frame length – 1)
5. Write WCFG command to CMD register
6. Write frame address to FAR
7. Write 212 words to FDRI (1 frame + 1) x (106 words per frame) = 212 words
8. Write default AutoCRC value of $0000DEFC$
9. Reset CRC
10. Write default CRC value ($0x0000DEFC$)
11. Write DESYNCH command
12. Write two NOOP words ($0x20000000$) to flush packet buffer
Consecutive frames of configuration frame data can be written to FDRI in a single packet (step 10 in the Shutdown Partial Reconfiguration process; step 7 in the Active Reconfiguration process), since the FAR automatically increments when each frame is written to configuration memory. In some cases, it is necessary to write to non-consecutive frames; in such instances, the new frame address must be explicitly written to the FAR and steps 8-11 repeated. As always, one frame of pad data must be included at the end of every FDRI packet, and the word following each FDRI packet is interpreted as an AutoCRC.

The precautions that apply to full active reconfiguration (see “Active and Shutdown Reconfiguration,” page 357) also apply to partial active reconfiguration. Table 4-35 shows the configuration packets needed to actively reconfigure a single frame on an XC2VP7 device.

**Table 4-35: Active Partial Reconfiguration for a Single XC2VP7 Frame**

<table>
<thead>
<tr>
<th>Step</th>
<th>Configuration Data</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AA995566</td>
<td>Sync Word</td>
</tr>
<tr>
<td>2</td>
<td>30008001</td>
<td>Write to CMD</td>
</tr>
<tr>
<td></td>
<td>00000007</td>
<td>RCRC</td>
</tr>
<tr>
<td>3</td>
<td>3001C001</td>
<td>Write to IDCODE register</td>
</tr>
<tr>
<td></td>
<td>0124A093</td>
<td>Device ID</td>
</tr>
<tr>
<td>4</td>
<td>30016001</td>
<td>Write to FLR</td>
</tr>
<tr>
<td></td>
<td>00000069</td>
<td>Frame Length</td>
</tr>
<tr>
<td>5</td>
<td>30008001</td>
<td>Write to CMD</td>
</tr>
<tr>
<td></td>
<td>00000001</td>
<td>WCFG</td>
</tr>
<tr>
<td>6</td>
<td>30002001</td>
<td>Write to FAR</td>
</tr>
<tr>
<td></td>
<td>XXXXXXXX</td>
<td>Frame address</td>
</tr>
<tr>
<td>7</td>
<td>300040D4</td>
<td>Write 212 words to FDRI (2 x 106 words per frame)</td>
</tr>
<tr>
<td></td>
<td>...</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>0000DEFC</td>
<td>Default AUTOCRC value</td>
</tr>
<tr>
<td>9</td>
<td>30008001</td>
<td>Write to CMD</td>
</tr>
<tr>
<td></td>
<td>00000007</td>
<td>RCRC</td>
</tr>
<tr>
<td>10</td>
<td>30000001</td>
<td>Type 1 write 1 words to CRC</td>
</tr>
<tr>
<td></td>
<td>0000DEFC</td>
<td>CRC = 0x0000DEFC (Default value)</td>
</tr>
<tr>
<td>11</td>
<td>30008001</td>
<td>Write to CMD</td>
</tr>
<tr>
<td></td>
<td>0000000D</td>
<td>Desync command</td>
</tr>
<tr>
<td>12</td>
<td>00000002</td>
<td>NOOP</td>
</tr>
<tr>
<td></td>
<td>00000002</td>
<td>NOOP</td>
</tr>
</tbody>
</table>
Partial Reconfiguration Applications and Software Flows

One challenge for designers seeking to use partial reconfiguration is determining which frames to reconfigure and what the new contents of those frames should be. There are four software flows available to partial reconfiguration users: the difference-based Bitgen `-r` flow, the Modular Design flow, the Bitgen "PartialMask" flow, and the BlockRAM "savedata" feature.

Difference-Based Bitgen `-r` Flow

In the difference-based Bitgen "-r" flow, users provide two input design (.ncd) files to Bitgen, designating one as the initial configuration and the other as the secondary configuration. Bitgen produces an active partial bitstream that only configures the frames that are different between the two design files. The partial bitstream is intended only for downloading to a device that has been configured by a bitstream for the initial configuration. The difference-based flow is well-suited partial reconfiguration applications where only small design changes are needed—changing LUT equations on the fly, for example.

Partial bit files generated with the Bitgen "-r" setting automatically use the multiple frame write feature and are therefore "compressed" bitstreams. The Difference-Based flow is described in greater detail in Xilinx Application note XAPP290, *Two Flows for Partial Reconfiguration: Module Based or Difference Based*.

Modular Design

The Modular Design flow for partial reconfiguration, also described in Xilinx Application Note XAPP290, is intended for partially reconfiguring larger design changes into a system. The flow provides for a static, non-reconfigurable design area and a dynamic, reconfigurable area that can be partially reconfigured with two or more partial reconfiguration modules.

PartialMask

The Bitgen "PartialMask" feature allows users to pick which configuration columns are included in an active reconfiguration bitstream. PartialMask bitstreams are intended only for active partial reconfiguration, and must be used with the `-g activereconfig:yes` Bitgen switch. Bitstreams that are created using this flow cannot be used for initial configuration, since they do not include the START command or allow for the startup sequence.

Per the Development System Reference Guide, there are six PartialMask settings:

- **PartialGCLK**: Adds the center global clock column to the list of columns written to a partial bitstream. Equivalent to the PartialMask0:1 setting.
- **PartialLeft**: Adds all columns on the left side of the device, excluding the global clock column, to the list of columns written to a partial bitstream.
- **PartialRight**: Adds all columns on the right side of the device, excluding the global clock column, to the list of columns written to a partial bitstream.
- **PartialMask0 <mask>**: Adds columns in BA0 (Block Address 0: GCLK, IOB, IOI, and CLB columns) as indicated by the hexadecimal mask to the list of columns written to a partial bitstream.
- **PartialMask1 <mask>**: Adds columns in BA1 (BRAM columns) as indicated by the hexadecimal mask to the list of columns written to a partial bitstream.
- **PartialMask2 <mask>**: Adds columns in BA2 (BRAM Interconnect columns) as indicated by the hexadecimal mask to the list of columns written to a partial bitstream.
For the PartialMaskN settings, <mask> is a hexadecimal string that indicates which columns within a memory block are included in the partial bitstream. The bits in the hex string map to configuration columns; each '1' indicates a column that is included in the partial bitstream, while each '0' indicates a column that is excluded from the partial bitstream. Nibbles are rounded out by appending '0's to the bit positions that do not have corresponding configuration columns. Figure 4-40 illustrates the mapping between the <mask> bit positions and the configuration columns.

Two examples follow for deriving masks for a XC2V40 Virtex-II device. (The procedure is identical for Virtex-II Pro devices.) The XC2V40 device has 8 CLB columns, 2 BlockRAM columns, and 2 BlockRAM Interconnect columns.

**PartialMask Example 1: All columns included in bitstream except BRAM columns.**

To include all columns in BA0 and BA1, the <mask> for PartialMask0 and PartialMask1 is created using '1's for all BA0 and BA1 columns. To exclude all columns in BA2, the <mask> for PartialMask2 is all '0's. Figure 4-41 shows how the <mask> values are derived.
To generate a partial bitstream for Example 1, use the command line:

```
bitgen -g ActiveReconfig:Yes -g PartialMask0:1FFF -g PartialMask1:0 -g PartialMask2:3 <design>.ncd
```

PartialMask Example 2: Only BRAM columns included in bitstream.

To exclude all columns in BA0 and BA1, the \texttt{<mask>} for PartialMask0 and PartialMask1 is all '0's. To include all columns in BA2, the \texttt{<mask>} for PartialMask2 is all '1's. Figure 4-42 shows how the \texttt{<mask>} values are derived.

To generate a partial bitstream for Example 2, use the command line:

```
bitgen -g ActiveReconfig:Yes -g PartialMask0:0000 -g PartialMask1:3 -g PartialMask2:0 <design>.ncd
```
BlockRAM "Savedata"

Writes to Virtex-II Pro BlockRAMs during shutdown reconfiguration can be disabled by setting the BlockRAM "Savedata" option in the original configuration bitstream. The Savedata option is enabled using FPGA Editor, as described in Xilinx Application Note XAPP290.

The Savedata option is not intended for use during active reconfiguration, as it can interfere with BlockRAM operation. The Savedata option is safe to use with Shutdown reconfiguration.

Readback

Virtex-II Pro devices allow users to read configuration memory through the SelectMAP or JTAG interface. There are two styles of readback: Readback Verify and Readback Capture. During Readback Verify, the user reads all configuration memory cells, including the current values on all user memory elements (LUT RAM, SRL16, and BlockRAM). Readback Capture is a superset of Readback Verify: in addition to reading all configuration memory cells, the current state of all internal CLB and IOB registers is read, which can be useful for design debugging. Readback is possible while the FPGA design is active or in a shutdown state, although users are advised to avoid active readback because it can corrupt BlockRAM, Distributed RAM, and SRL16 contents. During active readback, the user design continues to operate while configuration memory are being read.

To read configuration memory, users must send a sequence of commands to the device, indicating the starting configuration memory readback address and the number of words to be read from memory. Once initiated, the device sends the specified memory contents to the SelectMAP or JTAG interface. The following sections describe the steps for reading configuration memory:

- “Configuration Memory Read Procedure (SelectMAP),” page 367
- “Configuration Memory Read Procedure (1149.1 JTAG),” page 372
- “Configuration Memory Read Procedure (1532 JTAG),” page 376

Users can send the readback command sequence from a microprocessor, CPLD, or FPGA-based system, or use iMPACT to perform JTAG-based readback verify. iMPACT, the device programming software provided with the Xilinx Integrated Software Environment (ISE), can perform all readback and comparison functions for Virtex-II devices and report to the user whether there were any configuration errors. iMPACT cannot, however, perform capture operations, although Readback Capture is seldom used for design debugging since Chipscope ILA, sold separately through the Xilinx website, provides superior design debugging functionality in a user-friendly interface. See “Using ChipScope ILA,” page 381 for more information.

Once configuration memory has been read from the device, the next step is to determine if there are any errors by comparing the readback bitstream to the configuration bitstream. An explanation of how this is done can be found in “Verifying Readback Data,” page 377.

Preparing a Design for Readback

There are two mandatory bitstream settings for readback: the BitGen security setting must not prohibit readback (-g security:none), and bitstream encryption must not be used. Additionally, if readback is to be performed through the SelectMAP interface, the port must be set to retain its function after configuration by setting the "persist" option in BitGen (-g Persist:Yes). Otherwise, the SelectMAP data pins revert to user I/O,
precluding further configuration operations. Beyond these security and encryption requirements, no special considerations are necessary to enable readback through the Boundary-Scan port.

If capture functionality is needed, the CAPTURE_VIRTEX2 primitive must be instantiated in the user design (Figure 4-48, page 380). To capture the state of user registers, the user design triggers the CAP input on this primitive, storing the current register values in configuration memory. The register values are later read out of the device along with all other configuration memory. See “Readback Capture,” page 380 for more information.

Readback Command Sequences

Virtex-II configuration memory is read from the FDRO (Frame Data Register - Output) register, which can be accessed from the JTAG and SelectMAP interfaces. Readback is possible while the FPGA design is active or in a shutdown state, although users are advised to avoid active readback because it can corrupt BlockRAM, Distributed RAM, and SRL16 contents.

Configuration Register Read Procedure (SelectMAP)

The simplest read operation targets a configuration register such as the COR or STAT register. Any configuration register with read access can be read through the SelectMAP interface, although not all registers offer read access, as noted in Table 4-21, page 342. The procedure for reading the STAT register through the SelectMAP interface follows:

1. Write the Synchronization word to the device.
2. Write the "read STAT register" packet header to the device.
3. Write two pad words to the device to flush the packet buffer.
4. Read one word from the SelectMAP interface. This is the Status register value.
5. Write the DESYNCH command to the device
6. Write two pad words to the device to flush the packet buffer.

Table 4-36: Status Register Readback Command Sequence (SelectMAP)

<table>
<thead>
<tr>
<th>Step</th>
<th>SelectMAP Port Direction</th>
<th>Configuration Data (hex)</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Write</td>
<td>AA95566</td>
<td>Sync Word</td>
</tr>
<tr>
<td>2</td>
<td>Write</td>
<td>2800E001</td>
<td>Read 1 word from STAT register</td>
</tr>
<tr>
<td>3</td>
<td>Write</td>
<td>20000000</td>
<td>NOOP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20000000</td>
<td>NOOP</td>
</tr>
<tr>
<td>4</td>
<td>Read</td>
<td>SSSSSSSS</td>
<td>Device writes 1 word from the STAT register to the configuration interface</td>
</tr>
<tr>
<td>5</td>
<td>Write</td>
<td>0000000D</td>
<td>Desync command</td>
</tr>
<tr>
<td>6</td>
<td>Write</td>
<td>20000000</td>
<td>NOOP</td>
</tr>
<tr>
<td></td>
<td></td>
<td>20000000</td>
<td>NOOP</td>
</tr>
</tbody>
</table>

The user must change the SelectMAP interface from write to read control between steps 3 and 4, and back to write control after step 4, as illustrated in Figure 4-43.
To read registers other than STAT, the address specified in the Type 1 packet header in Step 2 of Table 4-36 should be modified and the word count changed if necessary. Reading from the FDRO register is a special case that is described in the procedure below.

**Configuration Memory Read Procedure (SelectMAP)**

The process for reading configuration memory from the FDRO register is similar to the process for reading from other registers. However, additional steps are needed to accommodate the configuration logic. Configuration data coming from the FDRO register pass through the frame buffer; therefore, the first frame of readback data should be discarded. (Refer to “Frame Data Input Register (FDRI) and Frame Data Output Register (FDRO),” page 344).

1. Write the Synchronization word to the device.
2. Write the SHUTDOWN command.
3. Write the RCRC command.
4. Write four NOOP instructions to ensure the shutdown sequence has completed. DONE goes Low after the SHUTDOWN sequence has completed.
5. Write the Starting Frame Address to the FAR (typically \(0x00000000\)).
6. Write the RCFG command to the CMD register.
7. Write the "read FDRO register" packet header to the device. The FDRO read length is given by:

\[
\text{FDRO Read Length} = (\text{words per frame}) \times (\text{frames to read} + 1)
\]

One extra frame is read to account for the frame buffer, which produces one pad frame at the beginning of the read.

8. Write two pad words to the device to flush the packet buffer.
9. Read the FDRO register through the SelectMAP interface. The FDRO read length is the same as in step 7 above. Frame data will be preceded by one frame of pad bits.
10. Write the START command.
11. Write the RCRC command.
12. Write the DESYNCH command.
13. Write at least 64 bits of NOOP commands to flush the packet buffer. Continue sending CCLK pulses until DONE goes High.

*Figure 4-43: SelectMAP Status Register Read*
Each of these steps is performed by a single configuration packet except for synchronization (step 1) and the large FDRO read (step 7), which is performed by a Type 1, Type 2 packet combination.

For active readback, step 5 and step 10 are omitted. Remember that BRAM frames and CLB frames that contain SRL16s or Distributed RAM must not be read back actively. In practice, active readback of the CLB frames should stop at the second-to-last CLB frame, since reading back the last CLB frame will cause the FAR to increment into the first BRAM frame, corrupting the BRAM contents.

Table 4-37 shows the readback command sequence.

**Table 4-37: Shutdown Readback Command Sequence (SelectMAP), XC2VP7 Device**

<table>
<thead>
<tr>
<th>Step</th>
<th>SelectMAP Port Direction</th>
<th>Configuration Data (hex)</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Write</td>
<td>AA955566</td>
<td>Sync word</td>
</tr>
<tr>
<td>2</td>
<td>Write</td>
<td>30008001</td>
<td>Type 1 write 1 words to CMD</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>0000000B</td>
<td>SHUTDOWN command</td>
</tr>
<tr>
<td>3</td>
<td>Write</td>
<td>30008001</td>
<td>Type 1 write 1 words to CMD</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>00000007</td>
<td>RCRC command</td>
</tr>
<tr>
<td>4</td>
<td>Write</td>
<td>20000000</td>
<td>Type 1 NOOP word 0</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>20000000</td>
<td>Type 1 NOOP word 1</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>20000000</td>
<td>Type 1 NOOP word 2</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>20000000</td>
<td>Type 1 NOOP word 3</td>
</tr>
<tr>
<td>5</td>
<td>Write</td>
<td>30002001</td>
<td>Type 1 write 1 words to FAR</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>00000000</td>
<td>FAR address= 0x00000000</td>
</tr>
<tr>
<td>6</td>
<td>Write</td>
<td>30008001</td>
<td>Type 1 write 1 words to CMD</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>00000004</td>
<td>RCFG command</td>
</tr>
<tr>
<td>7</td>
<td>Write</td>
<td>28006000</td>
<td>Type 1 read 0 words from FDRO</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>48002922</td>
<td>Type 2 read 140026 words from active register (FDRO)</td>
</tr>
<tr>
<td>8</td>
<td>Write</td>
<td>20000000</td>
<td>Type 1 NOOP word 0</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>20000000</td>
<td>Type 1 NOOP word 1</td>
</tr>
<tr>
<td>9</td>
<td>Read</td>
<td>00000000</td>
<td>Packet data read FDRO word 0</td>
</tr>
<tr>
<td></td>
<td>Read</td>
<td>...</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Read</td>
<td>00000000</td>
<td>Packet data read FDRO word 140025</td>
</tr>
<tr>
<td>10</td>
<td>Write</td>
<td>30008001</td>
<td>Type 1 write 1 words to CMD</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>00000005</td>
<td>START command</td>
</tr>
<tr>
<td>11</td>
<td>Write</td>
<td>30008001</td>
<td>Type 1 write 1 words to CMD</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>00000007</td>
<td>RCRC command</td>
</tr>
<tr>
<td>12</td>
<td>Write</td>
<td>30008001</td>
<td>Type 1 write 1 words to CMD</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>0000000D</td>
<td>DESYNCH command</td>
</tr>
<tr>
<td>13</td>
<td>Write</td>
<td>20000000</td>
<td>Type 1 NOOP word 0</td>
</tr>
<tr>
<td></td>
<td>Write</td>
<td>20000000</td>
<td>Type 1 NOOP word 1</td>
</tr>
</tbody>
</table>
Accessing Configuration Registers through the JTAG Interface

JTAG access to the Virtex-II configuration logic is provided through the JTAG CFG_IN and CFG_OUT registers. Note that the CFG_IN and CFG_OUT registers are not configuration registers; rather, they are JTAG registers like BYPASS and BOUNDARY_SCAN. Data shifted in to the CFG_IN register go to the configuration packet processor, where they are processed in the same way commands from the SelectMAP interface are processed.

Readback commands are written to the configuration logic through the CFG_IN register; configuration memory is read through the CFG_OUT register. The JTAG state transitions for accessing the CFG_IN and CFG_OUT registers XC2VP2 through XC2VP7 devices are described in Table 4-38. Other devices have larger instruction registers, so the instructions given in Table 4-38 would have to be modified by adding leading 1’s according to the device IR length. See “Boundary-Scan Instruction Set,” page 320 for more information.

Table 4-38:  Shifting in the JTAG CFG_IN and CFG_OUT Instructions, XC2VP2 through XC2VP7 Devices

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Set and Hold</th>
<th># of Clocks (TCK)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TDI</td>
<td>TMS</td>
</tr>
<tr>
<td>1</td>
<td>Clock five ‘1’s on TMS to bring the device to the TLR state</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Move into the RTI state</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>Move into the SELECT-IR state</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>Move into the SHIFT-IR State</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>Shift the first 9 bits of the CFG_IN or CFG_OUT instruction, LSB first</td>
<td>111000101 (CFG_IN)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>111000100 (CFG_OUT)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Shift the MSB of the CFG_IN or CFG_OUT instruction while exiting SHIFT-IR</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>Move into the SELECT-DR state</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>Move into the SHIFT-DR state</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>Shift data into the CFG_IN register or out of the CFG_OUT register while in SHIFT_DR, MSB first.</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>Shift the LSB while exiting SHIFT-DR</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>Reset the TAP by clocking five 1’s on TMS</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

Notes:
1. The JTAG instruction opcodes given in this table are for XC2VP2 through XC2VP7 devices. Other Virtex-II Pro devices have larger instruction registers, so the Instruction Opcodes for these devices must be modified. See “Boundary-Scan Instruction Set,” page 320 for more information, or refer to the appropriate device BSDL file.

Configuration Register Read Procedure (JTAG)

The simplest read operation targets a configuration register such as the COR or STAT register. Any configuration register with read access can be read through the JTAG interface, although not all registers offer read access, as noted in Table 4-21, page 342. The procedure for reading the STAT register through the JTAG interface follows:
1. Reset the TAP controller.
2. Shift the CFG_IN instruction into the JTAG Instruction Register through the SHIFT-IR state. The LSB of the CFG_IN instruction is shifted first; the MSB is shifted while moving the TAP controller out of the SHIFT-IR state.
3. Shift packet write commands into the CFG_IN register through the SHIFT-DR state:
   a. Write the Synchronization word to the device.
   b. Write the “Read STAT register” packet header to the device.
   c. Write exactly three pad words to the device to flush the packet buffer.

The MSB of all configuration packets sent through the CFG_IN register must be sent first; the LSB is shifted while moving the TAP controller out of the SHIFT-DR state. Note that this is the opposite of how JTAG instructions are shifted into the JTAG Instruction Register.

Since an odd number of words (1) is sent to the device for this JTAG read operation, it must end with exactly three pad words. This is because all JTAG read command sequences must consist of an even number of words (see “Packet Processor,” page 341 for details).
4. Shift the CFG_OUT instruction into the JTAG Instruction Register through the Shift-IR state. The LSB of the CFG_OUT instruction is shifted first; the MSB is shifted while moving the TAP controller out of the SHIFT-IR state.
5. Shift 32 bits out of the Status register through the SHIFT-DR state.
6. Shift the CFG_IN instruction into the JTAG Instruction Register through the Shift-IR state.
7. Shift packet write commands into the CFG_IN register through the Shift-DR state:
   a. Write the “Write to CMD register” packet header to the device.
   b. Write the DESYNC command to the device.
   c. Write two pad words to the device to flush the packet processor.
8. Reset the TAP controller. The device is desynchronized when the TAP controller moves into the Test-Logic-Reset state.

**Table 4-39: Status Register Readback Command Sequence (JTAG), XC2VP2 through XC2VP7 Devices**

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Set and Hold</th>
<th># of Clocks (TCK)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TDI</td>
<td>TMS</td>
</tr>
<tr>
<td>1</td>
<td>Clock five 1s on TMS to bring the device to the TLR state.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the RTI state.</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Move into the SELECT-IR state.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the SHIFT-IR state.</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Shift the first 9 bits of the CFG_IN instruction, LSB first.</td>
<td>111000101 (CFG_IN)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Shift the MSB of the CFG_IN instruction while exiting the SHIFT-IR state.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the SELECT-DR state.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the SHIFT-DR state.</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>
### Table 4-39: Status Register Readback Command Sequence (JTAG), XC2VP2 through XC2VP7 Devices (Continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Set and Hold</th>
<th># of Clocks (TCK)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td><strong>TDI</strong></td>
<td><strong>TMS</strong></td>
</tr>
<tr>
<td>3</td>
<td>Shift configuration packets into the CFG_IN data register, MSB first.</td>
<td>a: 0xAA995566</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b: 0x28088001</td>
<td>159</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c: 0x20000000</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x20000000</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Shift the LSB of the last configuration packet while exiting the SHIFT-DR state.</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the SELECT-IR state.</td>
<td>X</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Move into the SHIFT-IR state.</td>
<td>X</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>Shift the first 9 bits of the CFG_OUT instruction, LSB first.</td>
<td>111000100(CFG_OUT)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Shift the MSB of the CFG_OUT instruction while exiting the SHIFT-IR state.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the SELECT-DR state.</td>
<td>X</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Move into the SHIFT-DR state.</td>
<td>X</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>Shift the contents of the STAT register out of the CFG_OUT data register.</td>
<td>0xSSSSSSSSS</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Shift the last bit of the STAT register out of the CFG_OUT data register while exiting the SHIFT-DR state.</td>
<td>S</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the SELECT-IR state.</td>
<td>X</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>Move into the SHIFT-IR state.</td>
<td>X</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>Shift the first 5 bits of the CFG_IN instruction, LSB first</td>
<td>111000101(CFG_IN)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Shift the MSB of the CFG_IN instruction while exiting SHIFT-IR</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the SELECT-DR state.</td>
<td>X</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Move into the SHIFT-DR state.</td>
<td>X</td>
<td>2</td>
</tr>
<tr>
<td>7</td>
<td>Shift configuration packets into the CFG_IN data register, MSB first.</td>
<td>a: 0x30008001</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>b: 0x0000000D</td>
<td>127</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c: 0x20000000</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x20000000</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Shift the LSB of the last configuration packet while exiting SHIFT-DR</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>8</td>
<td>Reset the TAP Controller. The device is desynchronized when the TAP controller moves into the Test-Logic-Reset state.</td>
<td>X</td>
<td>5</td>
</tr>
</tbody>
</table>

### Notes:
1. The JTAG instruction opcodes given in this table are for XC2VP2 through XC2VP7 devices. Other Virtex-II Pro devices have larger instruction registers, so the instruction opcodes for these devices must be modified. See “Boundary-Scan Instruction Set,” page 320 for more information, or refer to the appropriate device BSDL file.
The packets shifted in to the JTAG CFG_IN register are identical to the packets shifted in through the SelectMAP interface when reading the STAT register through SelectMAP.

Configuration Memory Read Procedure (1149.1 JTAG)

The process for reading configuration memory from the FDRO register through the JTAG interface is similar to the process for reading from other registers. However, additional steps are needed to accommodate frame logic. Configuration data coming from the FDRO register pass through the frame buffer; therefore, the first frame of readback data is "pad data" and should be discarded (refer to the FDRI and FDRO Register description). The 1149.1 JTAG readback flow is recommended for most users.

1. Reset the TAP controller.
2. Shift the JSHUTDOWN instruction into the JTAG Instruction Register.
3. Move into the RTI state, and remain there for 12 TCK cycles to complete the Shutdown sequence. The DONE pin goes Low during the Shutdown sequence.
4. Shift the CFG_IN instruction into the JTAG Instruction Register. The LSB of the CFG_IN instruction is shifted first; the MSB is shifted while moving the TAP controller out of the SHIFT-IR state.
5. Shift packet write commands into the CFG_IN register through the SHIFT-DR state:
   a. Write the Synchronization word to the device.
   b. Write the "Write FAR register" Type 1 packet header to the device.
   c. Write the starting frame address to the FAR (typically 0x00000000).
   d. Write the "Write CMD register" Type 1 packet header to the device.
   e. Write the "RCRC" command to the device.
   f. Write the "Read FDRO register" Type 1 packet header to the device. The FDRO read length is given by:

   \[ \text{FDRO Read Length} = (\text{words per frame}) \times (\text{frames to read} + 1) \]

   One extra frame is read to account for the frame buffer, which produced one pad frame at the beginning of the read.
   g. Write a Type 2 packet header to indicate the number of words to read from the device.
   h. Write exactly two pad words to the device to flush the packet buffer.

The MSB of all configuration packets sent through the CFG_IN register must be sent first. The LSB is shifted while moving the TAP controller out of the SHIFT-DR state.

Since an even number of words (2) are sent to the device for this JTAG read operation, it must end with exactly two pad words. If the read operation had consisted of only a Type 1 packet header, instead of the Type 1, Type 2 packet combination that is used here, the sequence would have ended with exactly three pad words. This is because all JTAG read command sequences must consist of an even number of words (see “Packet Processor,” page 341 for details).

6. Shift the CFG_OUT instruction into the JTAG Instruction Register through the SHIFT-IR state. The LSB of the CFG_OUT instruction is shifted first; the MSB is shifted while moving the TAP controller out of the SHIFT-IR state.
7. Shift frame data from the FDRO register through the SHIFT-DR state.
8. Shift the CFG_IN instruction into the JTAG Instruction Register through the SHIFT-IR state.
9. Shift packet write commands into the CFG_IN register through the SHIFT-DR state:
   a. Write a NOOP command.
   b. Write the “Write to CMD register” packet header to the device.
   c. Write the START command.
   d. Write the “Write to CMD register” packet header to the device.
   e. Write the RCRC command.
   f. Write the “Write to CMD register” packet header to the device.
   g. Write the DESYNCH command to the device.
   h. Write two pad words to the device to flush the packet processor.
10. Shift the JSTART instruction into the JTAG Instruction Register.
11. Move to the RTI state and clock the STARTUP sequence by applying a minimum of 12 TCK cycles.
12. Reset the TAP controller.

### Table 4-40: Shutdown Readback Command Sequence (JTAG), XC2VP2 through XC2VP7 Devices

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Set and Hold</th>
<th># of Clocks (TCK)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TDI</td>
<td>TMS</td>
</tr>
<tr>
<td>1</td>
<td>Clock five 1s on TMS to bring the device to the TLR state.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the RTI state.</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Move into the SELECT-IR state.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the SHIFT-IR State.</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>Shift the first 9 bits of the JSHUTDOWN instruction, LSB first.</td>
<td>111001101</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Shift the MSB of the JSHUTDOWN instruction while exiting the SHIFT-IR state.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the UPDATE-IR state.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>Move into the RTI state; remain there for 12 TCK cycles.</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Move into the SELECT-IR state.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the SHIFT-IR State.</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>Shift the first 9 bits of the CFG_IN instruction, LSB first</td>
<td>111000101</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Shift the MSB of the CFG_IN instruction while exiting SHIFT-IR</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the SELECT-DR state</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the SHIFT-DR state</td>
<td>X</td>
<td>0</td>
</tr>
</tbody>
</table>
### Table 4-40: Shutdown Readback Command Sequence (JTAG), XC2VP2 through XC2VP7 Devices (Continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Set and Hold</th>
<th># of Clocks (TCK)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TDI</td>
<td>TMS</td>
</tr>
</tbody>
</table>
| 5    | Shift configuration packets into the CFG_IN data register, MSB first | a: 0xAA995566  
b: 0x30002001  
c: 0x00000000  
d: 0x30008001  
e: 0x00000007  
f: 0x28006000  
g: 0x480222FA  
h: 0x20000000 0x20000000 | 0  | 287 |
|      | Shift the LSB of the last configuration packet while exiting the SHIFT-DR state | 0 | 1  | 1 |
|      | Move into the SELECT-IR state | X | 1  | 3 |
|      | Move into the SHIFT-IR state | X | 0  | 2 |
| 6    | Shift the first 9 bits of the CFG_OUT instruction, LSB first | 111000100  
(CFG_OUT) 09 | 0  | 9  |
|      | Shift the MSB of the CFG_OUT instruction while exiting the SHIFT-IR state. | 1 | 1  | 1 |
|      | Move into the SELECT-DR state. | X | 1  | 2 |
|      | Move into the SHIFT-DR state. | X | 0  | 2 |
| 7    | Shift the contents of the FDRO register out of the CFG_OUT data register. | X | 0  | number of readback bits – 1 |
|      | Shift the last bit of the FDRO register out of the CFG_OUT data register while exiting the SHIFT-DR state. | X | 1  | 1 |
|      | Move into the SELECT-IR state. | X | 1  | 3 |
|      | Move into the SHIFT-IR state. | X | 0  | 2 |
| 8    | Shift the first 9 bits of the CFG_IN instruction, LSB first. | 111000101  
(CFG_IN) 09 | 0  | 9  |
|      | Shift the MSB of the CFG_IN instruction while exiting the SHIFT-IR state. | 1 | 1  | 1 |
|      | Move into the SELECT-DR state. | X | 1  | 2 |
|      | Move into the SHIFT-DR state. | X | 0  | 2 |
Table 4-40:  Shutdown Readback Command Sequence (JTAG), XC2VP2 through XC2VP7 Devices (Continued)

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Set and Hold</th>
<th># of Clocks (TCK)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>TDI</td>
<td>TMS</td>
</tr>
<tr>
<td>9</td>
<td>Shift configuration packets into the CFG_IN data register, MSB first.</td>
<td>a: 0x20000000</td>
<td>b: 0x30008001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>c: 0x00000005</td>
<td>d: 0x30008001</td>
</tr>
<tr>
<td></td>
<td>Shift the LSB of the last configuration packet while exiting the SHIFT-DR state.</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the UPDATE-DR state.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>Shift the first 9 bits of the JSTART instruction, LSB first.</td>
<td>111001100</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>(JSTART)</td>
<td>(JSTART)</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>Shift the MSB of the JSTART instruction while exiting the SHIFT-IR state.</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Move into the UPDATE-IR state.</td>
<td>X</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>Move into the RTI state; remain there for 12 TCK cycles.</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>12</td>
<td>End by placing the TAP controller in the TLR state.</td>
<td>X</td>
<td>1</td>
</tr>
</tbody>
</table>

Notes:
1. The JTAG instruction opcodes given in this table are for XC2VP2 through XC2VP7 devices. Other Virtex-II Pro devices have larger instruction registers, so the Instruction Opcodes for these devices must be modified. See “Boundary-Scan Instruction Set,” page 320 for more information, or refer to the appropriate device BSDL file.
Configuration Memory Read Procedure (1532 JTAG)

The IEEE 1532 JTAG readback procedure differs slightly from the IEEE 1149.1 JTAG readback procedure in that readback commands are not sent to the configuration logic via the CFG_IN JTAG register; rather, the ISC_READ JTAG register is used to directly read configuration memory.

At the end of 1532 JTAG readback, CRC Error status must be cleared by issuing a Reset CRC command or writing the correct CRC value to the CRC register. The 1532 JTAG readback procedure is illustrated in Figure 4-44.

![IEEE 1532 JTAG Readback Flow](image_url)

*Figure 4-44: IEEE 1532 JTAG Readback Flow*
Readback Files

The Xilinx Bitstream Generator can produce several files to assist the user in reading and comparing configuration memory. These files are described in Table 4-41.

Table 4-41:  Readback Files

<table>
<thead>
<tr>
<th>File Ext.</th>
<th>File Type</th>
<th>BitGen Setting</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.rba</td>
<td>ASCII</td>
<td>-b readback -g readback</td>
<td>An ASCII file that contains readback commands, rather than configuration commands, and expected readback data where the configuration data would normally be. This file must be used with the .msk file.</td>
</tr>
<tr>
<td>.rbb</td>
<td>Binary</td>
<td>-g readback</td>
<td>Binary version of .rba file. This file must be used with the .msk file.</td>
</tr>
<tr>
<td>.rbd</td>
<td>ASCII</td>
<td>-g readback</td>
<td>An ASCII file that contains only expected readback data, including the initial pad frame. No commands are included. This file must be used with the .msd file.</td>
</tr>
<tr>
<td>.msk</td>
<td>Binary</td>
<td>-m</td>
<td>A binary file that contains the same configuration commands as a .bit file, but replaces the contents of the FDRI write packet with mask data that indicate whether the corresponding bits in the .bit file should be compared. If a mask bit is 0, the corresponding bits in the readback data stream should be compared. If a mask bit is 1, the corresponding bit in the readback data stream should be ignored.</td>
</tr>
<tr>
<td>.msd</td>
<td>ASCII</td>
<td>-g readback</td>
<td>An ASCII file that contains only mask bits. The first bit in the .msd file corresponds to the first bit in the .rbd file. Pad data in the actual readback stream are accounted for in the .msd and .rbd files. If a mask bit is 0, that bit should be verified against the bit stream data. If a mask bit is 1, that bit should not be verified.</td>
</tr>
<tr>
<td>.ll</td>
<td>ASCII</td>
<td>-l</td>
<td>An ASCII file that contains information on each of the nodes in the design that can be captured for readback. The file contains the absolute bit position in the readback stream, frame address, frame offset, logic resource used, and name of the component in the design.</td>
</tr>
</tbody>
</table>

The design.rba and design.rbb files combine readback commands with expected readback data, whereas the .rbd file contains only expected readback data. Systems that use a .rbd file for readback must store readback commands elsewhere. The actual readback data must be masked against a .msk or .msd mask file, as certain bits within the expected readback stream in the .rba, .rbb, and .rbd files should be ignored.

The readback command set files do not indicate when users must change the SelectMAP or JTAG interface from write to read control; the user must handle this based on the Readback Command Sequences described above.

Verifying Readback Data

The readback data stream contains configuration frame data that are preceded by one frame of pad data, as described in the Configuration Memory Read procedure. The readback stream does not contain any of the commands or packet information found in the configuration bitstream, and no CRC calculation is performed during readback. The readback data stream is shown in Figure 4-45.
The readback data stream is verified by comparing it to the original configuration frame data that were programmed into the device. Certain bits within the readback data stream must not be compared, however, since these might correspond to user memory or null memory locations. The location of “don’t care” bits in the readback data stream is given by the mask files, the .msk and .msd files. These files have different formats although both convey essentially the same information. Once readback data have been obtained from the device, either of the following comparison procedures can be used:

1. **Compare readback data to the .rbd "golden" readback file, mask using the .msd file.**

The simplest way to verify the readback data stream is to compare it to the .rbd "golden" readback file, masking readback bits with the .msd file. This approach is simple because there is a 1:1 correspondence between the start of the readback data stream and the start of the .rbd and .msd files, making the task of aligning readback, mask, and expected data easier.
The .rbd and .msd files contain an ASCII representation of the readback and mask data along with a file header that lists the file name, etc. This header information should be ignored or deleted. The ASCII '1's and '0's in the .rbd and .msd files correspond to the binary readback data from the device. Take care to interpret these files as text, not binary sources. Users might wish to convert the .rbd and .msd files to a binary format using a script or text editor, as this can simplify the verify procedure for some systems and reduce the size of the files by a factor of eight.

The drawback to this approach is that in addition to storing the initial configuration bitstream and the .msd file, the golden .rbd file must be stored somewhere, increasing the overall storage requirement.

2. **Compare readback data to the configuration .bit file, mask using the .msk file.**

Another approach for verifying readback data is to compare the readback data stream to the frame data within the FDRI write in the original configuration bitstream, masking readback bits with the .msk file.

After sending readback commands to the device, comparison begins by aligning the beginning of the readback frame data to the beginning of the FDRI write in the .bit and .msk files. The comparison ends when the end of the FDRI write is reached.

This approach requires the least in-system storage space, since only the .bit, .msk, and readback commands must be stored.

![Diagram](Diagram.png)

*Figure 4-47: Comparing Readback Data Using the .msk and .bit Files*

The .rba and .rbb files contain expected readback data along with readback command sets. They are intended for use with the .msk file, although they are better suited to readback for Virtex devices (see Xilinx Application Note [XAPP138](#)) than for Virtex-II devices.
Readback Capture

The configuration memory readback command sequence is identical for both Readback Verify and Readback Capture. However, the Capture sequence requires an additional step to sample internal register values.

Users can sample CLB and IOB registers by instantiating the CAPTURE_VIRTEX2 primitive in their design (Figure 4-48) and asserting the CAP input on that primitive while the design is operating. On the next rising clock edge on the CAPTURE_VIRTEX2 CLK input, the internal GRDBK signal is asserted, storing all CLB and IOB register values into configuration memory cells. These values can then be read out of the device along with the IOB and CLB configuration columns by reading configuration memory through the readback process described above. The capture sequence causes the current register value to be stored to the same memory cell that programs the register's initial state. If the GRESTORE command is sent after the capture sequence has changed the state of this cell, the registers return to an unintended state.

The capture operation can also be performed by writing the GCAPTURE command to the CMD register.

![Figure 4-48: Virtex-II and Virtex-II Pro Library Primitive](http://www.xilinx.com/ug002_c4_35_091900)

<table>
<thead>
<tr>
<th>Signal</th>
<th>Description</th>
<th>Access</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCAPTURE</td>
<td>Captures the state of all slice and IOB registers. Complement of GRESTORE.</td>
<td>&quot;CAP&quot; input on capture block, user controlled. Also CMD register via GCAPTURE command.</td>
</tr>
<tr>
<td>GRESTORE</td>
<td>Initializes all registers as configured.</td>
<td>CMD register.</td>
</tr>
</tbody>
</table>

If the CAP signal is left asserted over multiple clock cycles, the Capture cell is updated with the new register value on each rising clock edge. To limit the capture operation to the first rising clock edge, the user can add the ONESHOT attribute to the CAPTURE_VIRTEX2 primitive. More information on the ONESHOT attribute can be found in the Constraints Guide.

Once the configuration memory frames have been read out of the device, the user can pick the captured register values out of the readback data stream. The capture bit locations are given in the logic allocation file (design.ll) as described below.
Figure 4-49 shows a snippet from a logic allocation file for the ISE "jc2_top" example design. The line from the header comments explaining the line format has been moved to the start of the bit offset data for clarity. The <offset> field gives the absolute bit offset from the beginning of configuration memory (frame address 0x00000000). The <frame address> field gives the frame address that the capture bit is located in, and the <frame offset> field gives the bit offset from the start of the frame. The <information> field gives the mapping between the bit and the user design—for example, the "DIR" register (Figure 4-49) that is located in Slice X8Y15 is located at bit offset 100790.

Note that captured DFF values, along with LUTRAM and SRL16 values, are stored in their inverted sense.

Using ChipScope ILA

The ChipScope ILA functional verification tool is currently sold separately through the Xilinx web site. This program uses a combination of PC software and instantiated soft cores to capture states of internal signals. This data is read out of the JTAG USER1 scan chain using the MultiLINX cable or a parallel cable. ChipScope ILA supports only the Virtex architecture and allows for the functional verification and debugging of an FPGA configured design.

ChipScope ILA supports the high-speed USB interface to the MultiLINX cable set on Windows 98/2000/XP platforms, and the RS232 connection on Windows 95/98/2000/NT/XP platforms. UNIX support is not available. More details are available under ChipScope ILA at: www.xilinx.com
Chapter 5

PCB Design Considerations

Summary

This chapter covers the following topics:

- Pinout Information
- Pinout Diagrams
- Package Specifications
- Flip-Chip Packages
- Thermal Data
- Printed Circuit Board Considerations
- Board Routability Guidelines
- XPower
- IBIS Models
- BSDL and Boundary Scan Models

Pinout Information

Introduction

Virtex-II Pro devices are available in the following package styles:

- FG256, FG456, and FG676: Wire-bond fine-pitch BGA of 1.00 mm pitch
- FGG256, FGG456, and FGG676: Pb-free wire-bond fine-pitch BGA of 1.00 mm pitch
- FF672, FF896, FF1152, FF1148, FF1517, FF1704, and FF1696: Flip-chip fine-pitch BGA of 1.00 mm pitch

Virtex-II Pro X devices are available in the following package styles:

- FF896 and FF1704: Flip-chip fine-pitch BGA of 1.00 mm pitch

Refer to the Virtex-II Pro and Virtex-II Pro X Platform FPGA Data Sheet for complete device specifications and all available pinout tables.

All devices available in a given package are pinout-compatible, and are listed in the same pinout table. Global pins, including JTAG, configuration, and power/ground pins, are listed at the end of each pinout table. Pins that are not available for the smaller devices are listed in the “No Connects” column. Each device is split into eight I/O banks to allow for flexibility in the choice of I/O standards.
Table 5-2, the FG256/FGG256 pinout table, is included in this volume as an example of the
pinout tables available in the Data Sheet and on the distribution CD-ROM. Always check
the online Data Sheet at www.xilinx.com for the most current technical information on this
and all other Xilinx devices.

Pin Definitions

Table 5-1 provides a description of each pin type listed in Virtex-II Pro pinout tables.

**Table 5-1: Virtex-II Pro Pin Definitions**

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User I/O Pins:</strong></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| IO_LXXY_#  | Input/Output | All user I/O pins are capable of differential signalling and can implement LVDS, ULVDS, BLVDS, LVPECL, or LDT pairs. Each user I/O is labeled “IO_LXXY_#”, where:
|            |           |            |
|            |           | IO indicates a user I/O pin. |
|            |           | LXXY indicates a differential pair, with XX a unique pair in the bank and Y = P/N for the positive and negative sides of the differential pair. |
|            |           | # indicates the bank number (0 through 7) |

| **Dual-Function Pins:** |           |             |
| IO_LXXY_/#/ZZZ          | Dual-function pins are labelled “IO_LXXY_#/ZZZ”, where “ZZZ” can be one of the following pins: Per Bank - VRP, VRN, or VREF
| Globally - GCLKX(S/P), BUSY/DOUT, INIT_B, D0/DIN – D7, RDWR_B, or CS_B |
| These dual functions are defined in the following section: |

| "ZZZ" (Dual Function) Definitions: |           |             |
| D0/DIN, D1, D2, D3, D4, D5, D6, D7 | Input/Output | In SelectMAP mode, D0 through D7 are configuration data pins. These pins become user I/Os after configuration, unless the SelectMAP port is retained.
|            |           | In bit-serial modes, DIN (D0) is the single-data input. This pin becomes a user I/O after configuration. |
| CS_B       | Input     | In SelectMAP mode, this is the active-low Chip Select signal. The pin becomes a user I/O after configuration, unless the SelectMAP port is retained. |
| RDWR_B     | Input     | In SelectMAP mode, this is the active-low Write Enable signal. The pin becomes a user I/O after configuration, unless the SelectMAP port is retained. |
| BUSY/DOUT  | Output    | In SelectMAP mode, BUSY controls the rate at which configuration data is loaded. The pin becomes a user I/O after configuration, unless the SelectMAP port is retained.
|            |           | In bit-serial modes, DOUT provides preamble and configuration data to downstream devices in a daisy-chain. The pin becomes a user I/O after configuration. |
| INIT_B     | Bidirectional (open-drain) | When Low, this pin indicates that the configuration memory is being cleared. When held Low, the start of configuration is delayed. During configuration, a Low on this output indicates that a configuration data error has occurred. The pin becomes a user I/O after configuration. |
## Pinout Information

**Table 5-1: Virtex-II Pro Pin Definitions (Continued)**

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GCLKx (S/P)</td>
<td>Input/Output</td>
<td>These are clock input pins that connect to Global Clock Buffers. These pins become regular user I/Os when not needed for clocks. These pins can be used to clock the RocketIO transceiver. See the <strong>RocketIO Transceiver User Guide</strong> for design guidelines and BREFCLK-specific pins, by device.</td>
</tr>
<tr>
<td>VRP</td>
<td>Input</td>
<td>This pin is for the DCI voltage reference resistor of P transistor (per bank).</td>
</tr>
<tr>
<td>VRN</td>
<td>Input</td>
<td>This pin is for the DCI voltage reference resistor of N transistor (per bank).</td>
</tr>
<tr>
<td>VREF</td>
<td>Input</td>
<td>These are input threshold voltage pins. They become user I/Os when an external threshold voltage is not needed (per bank).</td>
</tr>
</tbody>
</table>

**Dedicated Pins:**(1)

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCLK</td>
<td>Input/Output</td>
<td>Configuration clock. Output in Master mode or Input in Slave mode.</td>
</tr>
<tr>
<td>PROG_B</td>
<td>Input</td>
<td>Active Low asynchronous reset to configuration logic. This pin has a permanent weak pull-up resistor.</td>
</tr>
<tr>
<td>DONE</td>
<td>Input/Output</td>
<td>DONE is a bidirectional signal with an optional internal pull-up resistor. As an output, this pin indicates completion of the configuration process. As an input, a Low level on DONE can be configured to delay the start-up sequence.</td>
</tr>
<tr>
<td>M2, M1, M0</td>
<td>Input</td>
<td>Configuration mode selection. Pin is biased by VCCAUX (must be 2.5V). These pins should not connect to 3.3V unless 100Ω series resistors are used. The mode pins are not to be toggled (changed) while in operation during and after configuration.</td>
</tr>
<tr>
<td>HSWAP_EN</td>
<td>Input</td>
<td>Enable I/O pull-ups during configuration.</td>
</tr>
<tr>
<td>TCK</td>
<td>Input</td>
<td>Boundary Scan Clock. This pin is 3.3V compatible.</td>
</tr>
<tr>
<td>TDI</td>
<td>Input</td>
<td>Boundary Scan Data Input. This pin is 3.3V compatible.</td>
</tr>
<tr>
<td>TDO</td>
<td>Output (open-drain)</td>
<td>Boundary Scan Data Output. Pin is open-drain and can be pulled up to 3.3V. It is recommended that the external pull-up be greater than 200Ω. There is no internal pull-up.</td>
</tr>
<tr>
<td>TMS</td>
<td>Input</td>
<td>Boundary Scan Mode Select. This pin is 3.3V compatible.</td>
</tr>
<tr>
<td>PWRDWN_B</td>
<td>Input (unsupported)</td>
<td>Active Low power-down pin (unsupported). <em>Driving this pin Low can adversely affect device operation and configuration.</em> PWRDWN_B is internally pulled High, which is its default state. It does not require an external pull-up.</td>
</tr>
</tbody>
</table>

**Other Pins:**

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>DXN, DXP</td>
<td>N/A</td>
<td>Temperature-sensing diode pins (Anode: DXP, Cathode: DXN).</td>
</tr>
<tr>
<td>VBATT</td>
<td>Input</td>
<td>Decryptor key memory backup supply. Connect to VCCAUX or GND if battery is not used.</td>
</tr>
<tr>
<td>RSVD</td>
<td>N/A</td>
<td>Reserved pin - do not connect.</td>
</tr>
<tr>
<td>VCCO</td>
<td>Input</td>
<td>Power-supply pins for the output drivers (per bank).</td>
</tr>
<tr>
<td>VCCAUX</td>
<td>Input</td>
<td>Power-supply pins for auxiliary circuits.</td>
</tr>
<tr>
<td>VCCINT</td>
<td>Input</td>
<td>Power-supply pins for the internal core logic.</td>
</tr>
<tr>
<td>GND</td>
<td>Input</td>
<td>Ground.</td>
</tr>
</tbody>
</table>
**Table 5-1:  Virtex-II Pro Pin Definitions (Continued)**

<table>
<thead>
<tr>
<th>Pin Name</th>
<th>Direction</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AVCCAUXRX#</td>
<td>Input</td>
<td>Analog power supply for receive circuitry of the RocketIO MGT (2.5V).</td>
</tr>
<tr>
<td>AVCCAUXTX#</td>
<td>Input</td>
<td>Analog power supply for transmit circuitry of the RocketIO MGT (2.5V).</td>
</tr>
<tr>
<td>BREFCLKN,</td>
<td>Input</td>
<td>Differential clock input that clocks the RocketIO X MGTs populating the same</td>
</tr>
<tr>
<td>BREFCLKP(2)</td>
<td></td>
<td>side of the chip (top or bottom). Can also drive DCMs for RocketIO X MGT use.</td>
</tr>
<tr>
<td>VTRXPAD#</td>
<td>Input</td>
<td>Receive termination supply for the RocketIO MGT (1.8V - 2.8V).</td>
</tr>
<tr>
<td>VTTXPAD#</td>
<td>Input</td>
<td>Transmit termination supply for the RocketIO MGT (1.8V-2.8V).</td>
</tr>
<tr>
<td>GNDA#</td>
<td>Input</td>
<td>Ground for the analog circuitry of the RocketIO MGT.</td>
</tr>
<tr>
<td>RXPPAD#</td>
<td>Input</td>
<td>Positive differential receive port of the RocketIO MGT.</td>
</tr>
<tr>
<td>RXNPAD#</td>
<td>Input</td>
<td>Negative differential receive port of the RocketIO MGT.</td>
</tr>
<tr>
<td>TXPPAD#</td>
<td>Output</td>
<td>Positive differential transmit port of the RocketIO MGT.</td>
</tr>
<tr>
<td>TXNPAD#</td>
<td>Output</td>
<td>Negative differential transmit port of the RocketIO MGT.</td>
</tr>
</tbody>
</table>

**Notes:**
1. All dedicated pins (JTAG and configuration) are powered by VCCAUX (independent of the bank VCCO voltage).
2. Virtex-II Pro X devices XC2VPX20 and XC2VPX70 only. Each BREFCLKN(P) differential clock input pair takes the place of one regular Virtex-II Pro dual-function IO/GCLKx(S/P) pair on each side of the chip (top or bottom). For RocketIO BREFCLK, see section “BREFCLK Pin Definitions (RocketIO Only)” immediately following.

**BREFCLK Pin Definitions (RocketIO Only)**

These dedicated clocks use the same clock inputs for all packages:

<table>
<thead>
<tr>
<th>Top</th>
<th>BREFCLK</th>
<th>Direction</th>
<th>GCLK4S</th>
<th>Bottom</th>
<th>BREFCLK2</th>
<th>Direction</th>
<th>GCLK6P</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BREFCLK2</td>
<td>P</td>
<td>GCLK4S</td>
<td></td>
<td>BREFCLKK</td>
<td>P</td>
<td>GCLK6P</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>GCLK5P</td>
<td></td>
<td></td>
<td>N</td>
<td>GCLK7S</td>
</tr>
<tr>
<td></td>
<td>BREFCLK2</td>
<td>P</td>
<td>GCLK2S</td>
<td></td>
<td>BREFCLKK</td>
<td>P</td>
<td>GCLK0P</td>
</tr>
<tr>
<td></td>
<td></td>
<td>N</td>
<td>GCLK3P</td>
<td></td>
<td></td>
<td>N</td>
<td>GCLK1S</td>
</tr>
</tbody>
</table>

For detailed information about using BREFCLK/BREFCLK2, including routing considerations and pin numbers for all package types, refer to Chapter 2, "Digital Design Considerations,” in the RocketIO Transceiver User Guide.

**FG256/FGG256 Fine-Pitch BGA Package**

As shown in Table 5-2, XC2VP2 and XC2VP4 Virtex-II Pro devices are available in the FG256 and Pb-free FGG256 fine-pitch BGA packages. The pins in each of these devices are identical.

The FG256/FGG256 pinout information (Table 5-2) is included here as an example. All Virtex-II Pro pinout tables are available on the distribution CD-ROM. To be assured of getting the very latest updates and corrections, however, access the pinout tables in Module 4 of the online Data Sheet.
## Pinout Information

### Table 5-2: FG25/FGG2566 — XC2VP2 and XC2VP4

<table>
<thead>
<tr>
<th>Bank</th>
<th>Pin Description</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>IO_L01N_0/VRP_0</td>
<td>C2</td>
</tr>
<tr>
<td>0</td>
<td>IO_L01P_0/VRN_0</td>
<td>C3</td>
</tr>
<tr>
<td>0</td>
<td>IO_L02N_0</td>
<td>B3</td>
</tr>
<tr>
<td>0</td>
<td>IO_L02P_0</td>
<td>C4</td>
</tr>
<tr>
<td>0</td>
<td>IO_L03N_0</td>
<td>A2</td>
</tr>
<tr>
<td>0</td>
<td>IO_L03P_0/VREF_0</td>
<td>A3</td>
</tr>
<tr>
<td>0</td>
<td>IO_L06N_0</td>
<td>D5</td>
</tr>
<tr>
<td>0</td>
<td>IO_L06P_0</td>
<td>C5</td>
</tr>
<tr>
<td>0</td>
<td>IO_L07P_0</td>
<td>D6</td>
</tr>
<tr>
<td>0</td>
<td>IO_L09N_0</td>
<td>E6</td>
</tr>
<tr>
<td>0</td>
<td>IO_L09P_0/VREF_0</td>
<td>E7</td>
</tr>
<tr>
<td>0</td>
<td>IO_L09N_0</td>
<td>D7</td>
</tr>
<tr>
<td>0</td>
<td>IO_L09P_0/VREF_0</td>
<td>C7</td>
</tr>
<tr>
<td>0</td>
<td>IO_L74N_0/GCLK7P</td>
<td>D8</td>
</tr>
<tr>
<td>0</td>
<td>IO_L74P_0/GCLK6S</td>
<td>C8</td>
</tr>
<tr>
<td>0</td>
<td>IO_L75N_0/GCLK5P</td>
<td>B8</td>
</tr>
<tr>
<td>0</td>
<td>IO_L75P_0/GCLK4S</td>
<td>A8</td>
</tr>
<tr>
<td>1</td>
<td>IO_L75N_1/GCLK3P</td>
<td>A9</td>
</tr>
<tr>
<td>1</td>
<td>IO_L75P_1/GCLK2S</td>
<td>B9</td>
</tr>
<tr>
<td>1</td>
<td>IO_L74N_1/GCLK1P</td>
<td>C9</td>
</tr>
<tr>
<td>1</td>
<td>IO_L74P_1/GCLK0S</td>
<td>D9</td>
</tr>
<tr>
<td>1</td>
<td>IO_L69N_1/VREF_1</td>
<td>C10</td>
</tr>
<tr>
<td>1</td>
<td>IO_L69P_1</td>
<td>D10</td>
</tr>
<tr>
<td>1</td>
<td>IO_L09N_1/VREF_1</td>
<td>E10</td>
</tr>
<tr>
<td>1</td>
<td>IO_L09P_1</td>
<td>E11</td>
</tr>
<tr>
<td>1</td>
<td>IO_L07N_1</td>
<td>D11</td>
</tr>
<tr>
<td>1</td>
<td>IO_L06N_1</td>
<td>C12</td>
</tr>
<tr>
<td>1</td>
<td>IO_L06P_1</td>
<td>D12</td>
</tr>
<tr>
<td>1</td>
<td>IO_L03N_1/VREF_1</td>
<td>A14</td>
</tr>
<tr>
<td>1</td>
<td>IO_L03P_1</td>
<td>A15</td>
</tr>
<tr>
<td>1</td>
<td>IO_L02N_1</td>
<td>C13</td>
</tr>
<tr>
<td>1</td>
<td>IO_L02P_1</td>
<td>B14</td>
</tr>
<tr>
<td>1</td>
<td>IO_L01N_1/VRP_1</td>
<td>C14</td>
</tr>
<tr>
<td>1</td>
<td>IO_L01P_1/VRN_1</td>
<td>C15</td>
</tr>
</tbody>
</table>
### Table 5-2: FG25/FGG2566 — XC2VP2 and XC2VP4 (Continued)

<table>
<thead>
<tr>
<th>Bank</th>
<th>Pin Description</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>IO_L01N_2/VRP_2</td>
<td>E14</td>
</tr>
<tr>
<td>2</td>
<td>IO_L01P_2/VRN_2</td>
<td>E15</td>
</tr>
<tr>
<td>2</td>
<td>IO_L02N_2</td>
<td>E13</td>
</tr>
<tr>
<td>2</td>
<td>IO_L02P_2</td>
<td>F12</td>
</tr>
<tr>
<td>2</td>
<td>IO_L03N_2</td>
<td>F13</td>
</tr>
<tr>
<td>2</td>
<td>IO_L03P_2</td>
<td>F14</td>
</tr>
<tr>
<td>2</td>
<td>IO_L04N_2/VRF_2</td>
<td>F15</td>
</tr>
<tr>
<td>2</td>
<td>IO_L04P_2</td>
<td>F16</td>
</tr>
<tr>
<td>2</td>
<td>IO_L06N_2</td>
<td>G13</td>
</tr>
<tr>
<td>2</td>
<td>IO_L06P_2</td>
<td>G14</td>
</tr>
<tr>
<td>2</td>
<td>IO_L85N_2</td>
<td>G15</td>
</tr>
<tr>
<td>2</td>
<td>IO_L85P_2</td>
<td>G16</td>
</tr>
<tr>
<td>2</td>
<td>IO_L86N_2</td>
<td>G12</td>
</tr>
<tr>
<td>2</td>
<td>IO_L86P_2</td>
<td>H13</td>
</tr>
<tr>
<td>2</td>
<td>IO_L88N_2/VRF_2</td>
<td>H14</td>
</tr>
<tr>
<td>2</td>
<td>IO_L88P_2</td>
<td>H15</td>
</tr>
<tr>
<td>2</td>
<td>IO_L90N_2</td>
<td>H16</td>
</tr>
<tr>
<td>2</td>
<td>IO_L90P_2</td>
<td>J16</td>
</tr>
<tr>
<td>3</td>
<td>IO_L90N_3</td>
<td>J15</td>
</tr>
<tr>
<td>3</td>
<td>IO_L90P_3</td>
<td>J14</td>
</tr>
<tr>
<td>3</td>
<td>IO_L89N_3</td>
<td>J13</td>
</tr>
<tr>
<td>3</td>
<td>IO_L89P_3</td>
<td>K12</td>
</tr>
<tr>
<td>3</td>
<td>IO_L87N_3/VRF_3</td>
<td>K16</td>
</tr>
<tr>
<td>3</td>
<td>IO_L87P_3</td>
<td>K15</td>
</tr>
<tr>
<td>3</td>
<td>IO_L85N_3</td>
<td>K14</td>
</tr>
<tr>
<td>3</td>
<td>IO_L85P_3</td>
<td>K13</td>
</tr>
<tr>
<td>3</td>
<td>IO_L06N_3</td>
<td>L16</td>
</tr>
<tr>
<td>3</td>
<td>IO_L06P_3</td>
<td>L15</td>
</tr>
<tr>
<td>3</td>
<td>IO_L05N_3</td>
<td>L14</td>
</tr>
<tr>
<td>3</td>
<td>IO_L05P_3</td>
<td>L13</td>
</tr>
<tr>
<td>3</td>
<td>IO_L03N_3/VRF_3</td>
<td>L12</td>
</tr>
<tr>
<td>3</td>
<td>IO_L03P_3</td>
<td>M13</td>
</tr>
<tr>
<td>3</td>
<td>IO_L02N_3</td>
<td>M16</td>
</tr>
</tbody>
</table>
**Table 5-2: FG25/FGG2566 — XC2VP2 and XC2VP4 (Continued)**

<table>
<thead>
<tr>
<th>Bank</th>
<th>Pin Description</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>IO_L02P_3</td>
<td>N16</td>
</tr>
<tr>
<td>3</td>
<td>IO_L01N_3/VRP_3</td>
<td>M15</td>
</tr>
<tr>
<td>3</td>
<td>IO_L01P_3/VRN_3</td>
<td>M14</td>
</tr>
<tr>
<td>4</td>
<td>IO_L01N_4/DOUT</td>
<td>P15</td>
</tr>
<tr>
<td>4</td>
<td>IO_L01P_4/INIT_B</td>
<td>P14</td>
</tr>
<tr>
<td>4</td>
<td>IO_L02N_4/D0</td>
<td>R14</td>
</tr>
<tr>
<td>4</td>
<td>IO_L02P_4/D1</td>
<td>P13</td>
</tr>
<tr>
<td>4</td>
<td>IO_L03N_4/D2</td>
<td>T15</td>
</tr>
<tr>
<td>4</td>
<td>IO_L03P_4/D3</td>
<td>T14</td>
</tr>
<tr>
<td>4</td>
<td>IO_L06N_4/VRP_4</td>
<td>N12</td>
</tr>
<tr>
<td>4</td>
<td>IO_L06P_4/VRN_4</td>
<td>P12</td>
</tr>
<tr>
<td>4</td>
<td>IO_L07P_4/VREF_4</td>
<td>N11</td>
</tr>
<tr>
<td>4</td>
<td>IO_L09N_4</td>
<td>M11</td>
</tr>
<tr>
<td>4</td>
<td>IO_L09P_4/VREF_4</td>
<td>M10</td>
</tr>
<tr>
<td>4</td>
<td>IO_L69N_4</td>
<td>N10</td>
</tr>
<tr>
<td>4</td>
<td>IO_L69P_4/VREF_4</td>
<td>P10</td>
</tr>
<tr>
<td>4</td>
<td>IO_L74N_4/GCLK3S</td>
<td>N9</td>
</tr>
<tr>
<td>4</td>
<td>IO_L74P_4/GCLK2P</td>
<td>P9</td>
</tr>
<tr>
<td>4</td>
<td>IO_L75N_4/GCLK1S</td>
<td>R9</td>
</tr>
<tr>
<td>4</td>
<td>IO_L75P_4/GCLK0P</td>
<td>T9</td>
</tr>
<tr>
<td>5</td>
<td>IO_L75N_5/GCLK7S</td>
<td>T8</td>
</tr>
<tr>
<td>5</td>
<td>IO_L75P_5/GCLK6P</td>
<td>R8</td>
</tr>
<tr>
<td>5</td>
<td>IO_L74N_5/GCLK5S</td>
<td>P8</td>
</tr>
<tr>
<td>5</td>
<td>IO_L74P_5/GCLK4P</td>
<td>N8</td>
</tr>
<tr>
<td>5</td>
<td>IO_L69N_5/VREF_5</td>
<td>P7</td>
</tr>
<tr>
<td>5</td>
<td>IO_L69P_5</td>
<td>N7</td>
</tr>
<tr>
<td>5</td>
<td>IO_L09N_5/VREF_5</td>
<td>M7</td>
</tr>
<tr>
<td>5</td>
<td>IO_L09P_5</td>
<td>M6</td>
</tr>
<tr>
<td>5</td>
<td>IO_L07N_5/VREF_5</td>
<td>N6</td>
</tr>
<tr>
<td>5</td>
<td>IO_L06N_5/VRP_5</td>
<td>P5</td>
</tr>
<tr>
<td>5</td>
<td>IO_L06P_5/VRN_5</td>
<td>N5</td>
</tr>
<tr>
<td>5</td>
<td>IO_L03N_5/D4</td>
<td>T3</td>
</tr>
<tr>
<td>5</td>
<td>IO_L03P_5/D5</td>
<td>T2</td>
</tr>
</tbody>
</table>
### Table 5-2: FG25/FGG2566 — XC2VP2 and XC2VP4 (Continued)

<table>
<thead>
<tr>
<th>Bank</th>
<th>Pin Description</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>IO_L02N_5/D6</td>
<td>P4</td>
</tr>
<tr>
<td>5</td>
<td>IO_L02P_5/D7</td>
<td>R3</td>
</tr>
<tr>
<td>5</td>
<td>IO_L01N_5/RDWR_B</td>
<td>P3</td>
</tr>
<tr>
<td>5</td>
<td>IO_L01P_5/CS_B</td>
<td>P2</td>
</tr>
<tr>
<td>6</td>
<td>IO_L01P_6/VRN_6</td>
<td>M3</td>
</tr>
<tr>
<td>6</td>
<td>IO_L01N_6/VRP_6</td>
<td>M2</td>
</tr>
<tr>
<td>6</td>
<td>IO_L02P_6</td>
<td>N1</td>
</tr>
<tr>
<td>6</td>
<td>IO_L02N_6</td>
<td>M1</td>
</tr>
<tr>
<td>6</td>
<td>IO_L03P_6</td>
<td>M4</td>
</tr>
<tr>
<td>6</td>
<td>IO_L03N_6/VREF_6</td>
<td>L5</td>
</tr>
<tr>
<td>6</td>
<td>IO_L05P_6</td>
<td>L4</td>
</tr>
<tr>
<td>6</td>
<td>IO_L05N_6</td>
<td>L3</td>
</tr>
<tr>
<td>6</td>
<td>IO_L06P_6</td>
<td>L2</td>
</tr>
<tr>
<td>6</td>
<td>IO_L06N_6</td>
<td>L1</td>
</tr>
<tr>
<td>6</td>
<td>IO_L85P_6</td>
<td>K4</td>
</tr>
<tr>
<td>6</td>
<td>IO_L85N_6</td>
<td>K3</td>
</tr>
<tr>
<td>6</td>
<td>IO_L87P_6</td>
<td>K2</td>
</tr>
<tr>
<td>6</td>
<td>IO_L87N_6/VREF_6</td>
<td>K1</td>
</tr>
<tr>
<td>6</td>
<td>IO_L89P_6</td>
<td>K5</td>
</tr>
<tr>
<td>6</td>
<td>IO_L89N_6</td>
<td>J4</td>
</tr>
<tr>
<td>6</td>
<td>IO_L90P_6</td>
<td>J3</td>
</tr>
<tr>
<td>6</td>
<td>IO_L90N_6</td>
<td>J2</td>
</tr>
<tr>
<td>7</td>
<td>IO_L90P_7</td>
<td>J1</td>
</tr>
<tr>
<td>7</td>
<td>IO_L90N_7</td>
<td>H1</td>
</tr>
<tr>
<td>7</td>
<td>IO_L88P_7</td>
<td>H2</td>
</tr>
<tr>
<td>7</td>
<td>IO_L88N_7/VREF_7</td>
<td>H3</td>
</tr>
<tr>
<td>7</td>
<td>IO_L86P_7</td>
<td>H4</td>
</tr>
<tr>
<td>7</td>
<td>IO_L86N_7</td>
<td>G5</td>
</tr>
<tr>
<td>7</td>
<td>IO_L85P_7</td>
<td>G1</td>
</tr>
<tr>
<td>7</td>
<td>IO_L85N_7</td>
<td>G2</td>
</tr>
<tr>
<td>7</td>
<td>IO_L06P_7</td>
<td>G3</td>
</tr>
<tr>
<td>7</td>
<td>IO_L06N_7</td>
<td>G4</td>
</tr>
<tr>
<td>7</td>
<td>IO_L04P_7</td>
<td>F1</td>
</tr>
<tr>
<td>Bank</td>
<td>Pin Description</td>
<td>Pin Number</td>
</tr>
<tr>
<td>------</td>
<td>------------------------------</td>
<td>------------</td>
</tr>
<tr>
<td>7</td>
<td>IO_L04N_7/VREF_7</td>
<td>F2</td>
</tr>
<tr>
<td>7</td>
<td>IO_L03P_7</td>
<td>F3</td>
</tr>
<tr>
<td>7</td>
<td>IO_L03N_7</td>
<td>F4</td>
</tr>
<tr>
<td>7</td>
<td>IO_L02P_7</td>
<td>F5</td>
</tr>
<tr>
<td>7</td>
<td>IO_L02N_7</td>
<td>E4</td>
</tr>
<tr>
<td>7</td>
<td>IO_L01P_7/VRN_7</td>
<td>E2</td>
</tr>
<tr>
<td>7</td>
<td>IO_L01N_7/VRP_7</td>
<td>E3</td>
</tr>
<tr>
<td>0</td>
<td>VCCO_0</td>
<td>F8</td>
</tr>
<tr>
<td>0</td>
<td>VCCO_0</td>
<td>F7</td>
</tr>
<tr>
<td>0</td>
<td>VCCO_0</td>
<td>E8</td>
</tr>
<tr>
<td>1</td>
<td>VCCO_1</td>
<td>F9</td>
</tr>
<tr>
<td>1</td>
<td>VCCO_1</td>
<td>F10</td>
</tr>
<tr>
<td>1</td>
<td>VCCO_1</td>
<td>E9</td>
</tr>
<tr>
<td>2</td>
<td>VCCO_2</td>
<td>H12</td>
</tr>
<tr>
<td>2</td>
<td>VCCO_2</td>
<td>H11</td>
</tr>
<tr>
<td>2</td>
<td>VCCO_2</td>
<td>G11</td>
</tr>
<tr>
<td>3</td>
<td>VCCO_3</td>
<td>K11</td>
</tr>
<tr>
<td>3</td>
<td>VCCO_3</td>
<td>J12</td>
</tr>
<tr>
<td>3</td>
<td>VCCO_3</td>
<td>J11</td>
</tr>
<tr>
<td>4</td>
<td>VCCO_4</td>
<td>M9</td>
</tr>
<tr>
<td>4</td>
<td>VCCO_4</td>
<td>L9</td>
</tr>
<tr>
<td>4</td>
<td>VCCO_4</td>
<td>L10</td>
</tr>
<tr>
<td>5</td>
<td>VCCO_5</td>
<td>M8</td>
</tr>
<tr>
<td>5</td>
<td>VCCO_5</td>
<td>L8</td>
</tr>
<tr>
<td>5</td>
<td>VCCO_5</td>
<td>L7</td>
</tr>
<tr>
<td>6</td>
<td>VCCO_6</td>
<td>K6</td>
</tr>
<tr>
<td>6</td>
<td>VCCO_6</td>
<td>J6</td>
</tr>
<tr>
<td>6</td>
<td>VCCO_6</td>
<td>J5</td>
</tr>
<tr>
<td>7</td>
<td>VCCO_7</td>
<td>H6</td>
</tr>
<tr>
<td>7</td>
<td>VCCO_7</td>
<td>H5</td>
</tr>
<tr>
<td>7</td>
<td>VCCO_7</td>
<td>G6</td>
</tr>
<tr>
<td>N/A</td>
<td>CCLK</td>
<td>N15</td>
</tr>
<tr>
<td>N/A</td>
<td>PROG_B</td>
<td>D1</td>
</tr>
</tbody>
</table>
## Table 5-2: FG25/FGG2566 — XC2VP2 and XC2VP4 (Continued)

<table>
<thead>
<tr>
<th>Bank</th>
<th>Pin Description</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>DONE</td>
<td>P16</td>
</tr>
<tr>
<td>N/A</td>
<td>M0</td>
<td>N3</td>
</tr>
<tr>
<td>N/A</td>
<td>M1</td>
<td>N2</td>
</tr>
<tr>
<td>N/A</td>
<td>M2</td>
<td>P1</td>
</tr>
<tr>
<td>N/A</td>
<td>TCK</td>
<td>D16</td>
</tr>
<tr>
<td>N/A</td>
<td>TDI</td>
<td>E1</td>
</tr>
<tr>
<td>N/A</td>
<td>TDO</td>
<td>E16</td>
</tr>
<tr>
<td>N/A</td>
<td>TMS</td>
<td>C16</td>
</tr>
<tr>
<td>N/A</td>
<td>PWRDWN_B</td>
<td>N14</td>
</tr>
<tr>
<td>N/A</td>
<td>HSWAP_EN</td>
<td>C1</td>
</tr>
<tr>
<td>N/A</td>
<td>RSVD</td>
<td>D14</td>
</tr>
<tr>
<td>N/A</td>
<td>VBATT</td>
<td>D15</td>
</tr>
<tr>
<td>N/A</td>
<td>DXP</td>
<td>D2</td>
</tr>
<tr>
<td>N/A</td>
<td>DXN</td>
<td>D3</td>
</tr>
<tr>
<td>N/A</td>
<td>AVCCAUXTX6</td>
<td>B5</td>
</tr>
<tr>
<td>N/A</td>
<td>VTTXPAD6</td>
<td>B4</td>
</tr>
<tr>
<td>N/A</td>
<td>TXNPAD6</td>
<td>A4</td>
</tr>
<tr>
<td>N/A</td>
<td>TXPPAD6</td>
<td>A5</td>
</tr>
<tr>
<td>N/A</td>
<td>GNDA6</td>
<td>C6</td>
</tr>
<tr>
<td>N/A</td>
<td>GNDA6</td>
<td>C6</td>
</tr>
<tr>
<td>N/A</td>
<td>RXPPAD6</td>
<td>A6</td>
</tr>
<tr>
<td>N/A</td>
<td>RXNPAD6</td>
<td>A7</td>
</tr>
<tr>
<td>N/A</td>
<td>VTRXPAD6</td>
<td>B6</td>
</tr>
<tr>
<td>N/A</td>
<td>AVCCAUUXRX6</td>
<td>B7</td>
</tr>
<tr>
<td>N/A</td>
<td>AVCCAUXTX7</td>
<td>B11</td>
</tr>
<tr>
<td>N/A</td>
<td>VTTXPAD7</td>
<td>B10</td>
</tr>
<tr>
<td>N/A</td>
<td>TXNPAD7</td>
<td>A10</td>
</tr>
<tr>
<td>N/A</td>
<td>TXPPAD7</td>
<td>A11</td>
</tr>
<tr>
<td>N/A</td>
<td>GNDA7</td>
<td>C11</td>
</tr>
<tr>
<td>N/A</td>
<td>GNDA7</td>
<td>C11</td>
</tr>
<tr>
<td>N/A</td>
<td>RXPPAD7</td>
<td>A12</td>
</tr>
<tr>
<td>N/A</td>
<td>RXNPAD7</td>
<td>A13</td>
</tr>
<tr>
<td>N/A</td>
<td>VTRXPAD7</td>
<td>B12</td>
</tr>
<tr>
<td>N/A</td>
<td>AVCCAUUXRX7</td>
<td>B13</td>
</tr>
<tr>
<td>N/A</td>
<td>AVCCAUUXRX18</td>
<td>R13</td>
</tr>
</tbody>
</table>
### Table 5-2: FG25/FG2566 — XC2VP2 and XC2VP4 (Continued)

<table>
<thead>
<tr>
<th>Bank</th>
<th>Pin Description</th>
<th>Pin Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>VTRXPAD18</td>
<td>R12</td>
</tr>
<tr>
<td>N/A</td>
<td>RXNPAD18</td>
<td>T13</td>
</tr>
<tr>
<td>N/A</td>
<td>RXPPAD18</td>
<td>T12</td>
</tr>
<tr>
<td>N/A</td>
<td>GNDA18</td>
<td>P11</td>
</tr>
<tr>
<td>N/A</td>
<td>GNDA18</td>
<td>P11</td>
</tr>
<tr>
<td>N/A</td>
<td>TXPPAD18</td>
<td>T11</td>
</tr>
<tr>
<td>N/A</td>
<td>TXNpad18</td>
<td>T10</td>
</tr>
<tr>
<td>N/A</td>
<td>VTTXPAD18</td>
<td>R10</td>
</tr>
<tr>
<td>N/A</td>
<td>AVCCAUXTX18</td>
<td>R11</td>
</tr>
<tr>
<td>N/A</td>
<td>AVCCAUUXRX19</td>
<td>R7</td>
</tr>
<tr>
<td>N/A</td>
<td>VTRXPAD19</td>
<td>R6</td>
</tr>
<tr>
<td>N/A</td>
<td>RXNPAD19</td>
<td>T7</td>
</tr>
<tr>
<td>N/A</td>
<td>RXPPAD19</td>
<td>T6</td>
</tr>
<tr>
<td>N/A</td>
<td>GNDA19</td>
<td>P6</td>
</tr>
<tr>
<td>N/A</td>
<td>GNDA19</td>
<td>P6</td>
</tr>
<tr>
<td>N/A</td>
<td>TXPPAD19</td>
<td>T5</td>
</tr>
<tr>
<td>N/A</td>
<td>TXNpad19</td>
<td>T4</td>
</tr>
<tr>
<td>N/A</td>
<td>VTTXPAD19</td>
<td>R4</td>
</tr>
<tr>
<td>N/A</td>
<td>AVCCAUXTX19</td>
<td>R5</td>
</tr>
<tr>
<td>N/A</td>
<td>VCCINT</td>
<td>N4</td>
</tr>
<tr>
<td>N/A</td>
<td>VCCINT</td>
<td>N13</td>
</tr>
<tr>
<td>N/A</td>
<td>VCCINT</td>
<td>M5</td>
</tr>
<tr>
<td>N/A</td>
<td>VCCINT</td>
<td>M12</td>
</tr>
<tr>
<td>N/A</td>
<td>VCCINT</td>
<td>E5</td>
</tr>
<tr>
<td>N/A</td>
<td>VCCINT</td>
<td>E12</td>
</tr>
<tr>
<td>N/A</td>
<td>VCCINT</td>
<td>D4</td>
</tr>
<tr>
<td>N/A</td>
<td>VCCINT</td>
<td>D13</td>
</tr>
<tr>
<td>N/A</td>
<td>VCCAUX</td>
<td>R16</td>
</tr>
<tr>
<td>N/A</td>
<td>VCCAUX</td>
<td>R1</td>
</tr>
<tr>
<td>N/A</td>
<td>VCCAUX</td>
<td>B16</td>
</tr>
<tr>
<td>N/A</td>
<td>VCCAUX</td>
<td>B1</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>T16</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>T1</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>R2</td>
</tr>
<tr>
<td>Bank</td>
<td>Pin Description</td>
<td>Pin Number</td>
</tr>
<tr>
<td>------</td>
<td>----------------</td>
<td>------------</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>R15</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>L6</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>L11</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>K9</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>K8</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>K7</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>K10</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>J9</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>J8</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>J7</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>J10</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>H9</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>H8</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>H7</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>H10</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>G9</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>G8</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>G7</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>G10</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>F6</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>F11</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>B2</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>B15</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>A16</td>
</tr>
<tr>
<td>N/A</td>
<td>GND</td>
<td>A1</td>
</tr>
</tbody>
</table>
Pinout Diagrams

This section contains pinout diagrams for all Virtex-II Pro package/device combinations:

- **FG256/FGG256 Fine-Pitch BGA Pinout Diagrams**
  - FG256/FGG256 Composite Pinout Diagram (XC2VP2, XC2VP4), page 397
  - FG256/FGG256 Banked Pins (XC2VP2, XC2VP4), page 398
  - FG256/FGG256 Dedicated/Other Pins (XC2VP2, XC2VP4), page 399

- **FG456/FGG456 Fine-Pitch BGA Pinout Diagrams**
  - FG456/FGG456 Composite Pinout Diagram (XC2VP7), page 401
  - FG456/FGG456 Dedicated/Other Pins (XC2VP7), page 402
  - FG456/FGG456 Composite Pinout Diagram (XC2VP4), page 403
  - FG456/FGG456 Banked Pins (XC2VP4, XC2VP7), page 404
  - FG456/FGG456 Composite Pinout Diagram (XC2VP2), page 405
  - FG456/FGG456 Banked Pins (XC2VP2), page 406
  - FG456/FGG456 Dedicated/Other Pins (XC2VP2, XC2VP4, page 407

- **FG676/FGG676 Fine-Pitch BGA Pinout Diagrams**
  - FG676/FGG676 Composite Pinout Diagram (XC2VP30, XC2VP40), page 409
  - FG676/FGG676 Banked Pins (XC2VP30, XC2VP40), page 410
  - FG676/FGG676 Composite Pinout Diagram (XC2VP20), page 411
  - FG676/FGG676 Banked Pins (XC2VP20), page 412
  - FG676/FGG676 Dedicated/Other Pins (XC2VP20, XC2VP30, XC2VP40), page 413

- **FF672 Flip-Chip Fine-Pitch BGA Pinout Diagrams**
  - FF672 Composite Pinout Diagram (XC2VP7), page 415
  - FF672 Banked Pins (XC2VP7), page 416
  - FF672 Dedicated/Other Pins (XC2VP7), page 417
  - FF672 Composite Pinout Diagram (XC2VP4), page 418
  - FF672 Banked Pins (XC2VP4), page 419
  - FF672 Composite Pinout Diagram (XC2VP2), page 420
  - FF672 Banked Pins (XC2VP2), page 421
  - FF672 Dedicated/Other Pins (XC2VP2, XC2VP4), page 422

- **FF896 Flip-Chip Fine-Pitch BGA Pinout Diagrams**
  - FF896 Composite Pinout Diagram (XC2VP20, XC2VP30), page 423
  - FF896 Banked Pins (XC2VP20, XC2VP30), page 424
  - FF896 Composite Pinout Diagram (XC2VP7), page 425
  - FF896 Banked Pins (XC2VP7), page 426
  - FF896 Dedicated/Other Pins (XC2VP7, XC2VP20, XC2VP30), page 427
  - FF896 Composite Pinout Diagram (XC2VPX20), page 428
  - FF896 Banked Pins (XC2VPX20), page 429
  - FF896 Dedicated/Other Pins (XC2VPX20), page 430
- FF1152 Flip-Chip Fine-Pitch BGA Pinout Diagrams
  - FF1152 Composite Pinout Diagram (XC2VP50), page 431
  - FF1152 Dedicated/Other Pins (XC2VP50), page 432
  - FF1152 Composite Pinout Diagram (XC2VP40), page 433
  - FF1152 Banked Pins (XC2VP40, XC2VP50), page 434
  - FF1152 Dedicated/Other Pins (XC2VP40), page 435
  - FF1152 Composite Pinout Diagram (XC2VP30), page 436
  - FF1152 Banked Pins (XC2VP30), page 437
  - FF1152 Dedicated/Other Pins (XC2VP30), page 438
  - FF1152 Composite Pinout Diagram (XC2VP20), page 439
  - FF1152 Banked Pins (XC2VP20), page 440
  - FF1152 Dedicated/Other Pins (XC2VP20), page 441

- FF1148 Flip-Chip Fine-Pitch BGA Pinout Diagrams
  - FF1148 Composite Pinout Diagram (XC2VP50), page 443
  - FF1148 Banked Pins (XC2VP50), page 444
  - FF1148 Composite Pinout Diagram (XC2VP40), page 445
  - FF1148 Banked Pins (XC2VP40), page 446
  - FF1148 Dedicated/Other Pins (XC2VP40, XC2VP50), page 447

- FF1517 Flip-Chip Fine-Pitch BGA Pinout Diagrams
  - FF1517 Composite Pinout Diagram (XC2VP70), page 449
  - FF1517 Banked Pins (XC2VP70), page 450
  - FF1517 Composite Pinout Diagram (XC2VP50), page 451
  - FF1517 Banked Pins (XC2VP50), page 452
  - FF1517 Dedicated/Other Pins (XC2VP50, XC2VP70), page 453

- FF1704 Flip-Chip Fine-Pitch BGA Pinout Diagrams
  - FF1704 Composite Pinout Diagram (XC2VP100), page 455
  - FF1704 Banked Pins (XC2VP100), page 456
  - FF1704 Composite Pinout Diagram (XC2VP70), page 457
  - FF1704 Banked Pins (XC2VP70), page 458
  - FF1704 Dedicated/Other Pins (XC2VP70, XC2VP100), page 459
  - FF1704 Composite Pinout Diagram (XC2VPX70), page 460
  - FF1704 Banked Pins (XC2VPX70), page 461
  - FF1704 Dedicated/Other Pins (XC2VPX70), page 462

- FF1696 Flip-Chip Fine-Pitch BGA Pinout Diagrams
  - FF1696 Composite Pinout Diagram (XC2VP100), page 463
  - FF1696 Banked Pins (XC2VP100), page 464
  - FF1696 Dedicated/Other Pins (XC2VP100), page 465
FG256/FGG256 Fine-Pitch BGA Pinout Diagrams

FG256/FGG256 Composite Pinout Diagram (XC2VP2, XC2VP4)

**FG256/FGG256 (XC2VP2/4) - Top View**

Bank 0 - 17 Max User I/Os*

Bank 1 - 17 Max User I/Os*

Bank 2 - 18 Max User I/Os*

Bank 3 - 18 Max User I/Os*

Bank 4 - 17 Max User I/Os*

Bank 5 - 17 Max User I/Os*

Bank 6 - 18 Max User I/Os*

Bank 7 - 18 Max User I/Os*

---

**Figure 5-1:** FG256/FGG256 Composite Pinout Diagram (XC2VP2, XC2VP4)
Chapter 5: PCB Design Considerations

FG256/FGG256 Banked Pins (XC2VP2, XC2VP4)

**Figure 5-2:** FG256/FGG256 Banked Pins (XC2VP2, XC2VP4)
FG256/FGG256 Dedicated/Other Pins (XC2VP2, XC2VP4)

Figure 5-3: FG256/FGG256 Dedicated/Other Pins (XC2VP2, XC2VP4)
This page intentionally left blank.
FG456/FGG456 Fine-Pitch BGA Pinout Diagrams

FG456/FGG456 Composite Pinout Diagram (XC2VP7)

**Figure 5-4:** FG456/FGG456 Composite Pinout Diagram (XC2VP7)
FG456/FGG456 Dedicated/Other Pins (XC2VP7)

**Figure 5-5:** FG456/FGG456 Dedicated/Other Pins (XC2VP7)
FG456/FGG456 Composite Pinout Diagram (XC2VP4)

**Figure 5-6:** FG456/FGG456 Composite Pinout Diagram (XC2VP4)
Chapter 5: PCB Design Considerations

FG456/FGG456 Banked Pins (XC2VP4, XC2VP7)

* Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.

<table>
<thead>
<tr>
<th>User I/O Pins</th>
<th>Dedicated Pins</th>
<th>Other Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO_LXXY_#</td>
<td></td>
<td>VCCO</td>
</tr>
<tr>
<td>Dual-Purpose Pins:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIN/D0-D7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDWR_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUSY/DOUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INIT_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCLKx (P)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCLKx (S)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VREF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Pair</td>
<td></td>
<td>NO CONNECT</td>
</tr>
</tbody>
</table>

Figure 5-7: FG456/FGG456 Banked Pins (XC2VP4, XC2VP7)
Figure 5-8: FG456/FGG456 Composite Pinout Diagram (XC2VP2)
Chapter 5: PCB Design Considerations

FG456/FGG456 Banked Pins (XC2VP2)

**Figure 5-9: FG456/FGG456 Banked Pins (XC2VP2)**

* Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.
FG456/FGG456 Dedicated/Other Pins (XC2VP2, XC2VP4)

**Figure 5-10:** FG456/FGG456 Dedicated/Other Pins (XC2VP2, XC2VP4)
This page intentionally left blank.
FG676/FGG676 Fine-Pitch BGA Pinout Diagrams

FG676/FGG676 Composite Pinout Diagram (XC2VP30, XC2VP40)

**Bank 0 - 43 Max User I/Os**

**Bank 1 - 43 Max User I/Os**

**Bank 2 - 60 Max User I/Os**

**Bank 3 - 60 Max User I/Os**

**Bank 4 - 43 Max User I/Os**

**Bank 5 - 43 Max User I/Os**

**Bank 6 - 62 Max User I/Os**

**Bank 7 - 60 Max User I/Os**

* Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.

<table>
<thead>
<tr>
<th>User I/O Pins</th>
<th>Dedicated Pins</th>
<th>Other Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID</td>
<td>CCLK</td>
<td>DXN</td>
</tr>
<tr>
<td>ID</td>
<td>PROG_B</td>
<td>DXP</td>
</tr>
<tr>
<td>ID</td>
<td>DONE</td>
<td>VBATT</td>
</tr>
<tr>
<td>ID</td>
<td>M2, M1, M0</td>
<td>RSVD</td>
</tr>
<tr>
<td>ID</td>
<td>HSWAP_EN</td>
<td>VCCO</td>
</tr>
<tr>
<td>ID</td>
<td>TCK</td>
<td>VCCAUX</td>
</tr>
<tr>
<td>ID</td>
<td>TDI</td>
<td>VCCINT</td>
</tr>
<tr>
<td>ID</td>
<td>TDO</td>
<td>GND</td>
</tr>
<tr>
<td>ID</td>
<td>TMS</td>
<td>NO CONNECT</td>
</tr>
<tr>
<td>ID</td>
<td>PWRDWN_B</td>
<td>AVCCAUennon-RX</td>
</tr>
<tr>
<td>ID</td>
<td>RXPPAD</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>TXPPAD</td>
<td></td>
</tr>
<tr>
<td>ID</td>
<td>RXNPAD</td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5-11:** FG676/FGG676 Composite Pinout Diagram (XC2VP30, XC2VP40)
Chapter 5: PCB Design Considerations

FG676/FGG676 Banked Pins (XC2VP30, XC2VP40)

Figure 5-12: FG676/FGG676 Banked Pins (XC2VP30, XC2VP40)
Figure 5-13: FG676/FGG676 Composite Pinout Diagram (XC2VP20)
Chapter 5: PCB Design Considerations

FG676/FGG676 Banked Pins (XC2VP20)

Figure 5-14: FG676/FGG676 Banked Pins (XC2VP20)
FG676/FGG676 Dedicated/Other Pins (XC2VP20, XC2VP30, XC2VP40)

**Figure 5-15:** FG676/FGG676 Dedicated/Other Pins (XC2VP20, XC2VP30, XC2VP40)
This page intentionally left blank.
## FF672 Flip-Chip Fine-Pitch BGA Pinout Diagrams

### FF672 Composite Pinout Diagram (XC2VP7)

#### FF672 (XC2VP7) - Top View

**Bank 1 - 39 Max User I/Os**

**Bank 0 - 39 Max User I/Os**

**Bank 1** - 60 Max User I/Os

**Bank 2** - 60 Max User I/Os

**Bank 3** - 60 Max User I/Os

**Bank 4** - 39 Max User I/Os

**Bank 5** - 39 Max User I/Os

**Bank 6** - 60 Max User I/Os

**Bank 7** - 60 Max User I/Os

### Detailed Table

<table>
<thead>
<tr>
<th>User I/O Pins</th>
<th>Dedicated Pins</th>
<th>Other Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO_LXY#</td>
<td>CCLK</td>
<td>DXN</td>
</tr>
<tr>
<td>DIN/D0-D7</td>
<td>PROG_B</td>
<td>DXP</td>
</tr>
<tr>
<td>CS_B</td>
<td>DONE</td>
<td>VBATT</td>
</tr>
<tr>
<td>RDWR_B</td>
<td>M2, M1, M0</td>
<td>RSVD</td>
</tr>
<tr>
<td>BUSY/DOUT</td>
<td>HSWAP_EN</td>
<td>VCCD</td>
</tr>
<tr>
<td>INIT_B</td>
<td>TCK</td>
<td>VCAUX</td>
</tr>
<tr>
<td>GCLKx (P)</td>
<td>TDI</td>
<td>VCCINT</td>
</tr>
<tr>
<td>GCLKx (S)</td>
<td>TDO</td>
<td>GND</td>
</tr>
<tr>
<td>VRP</td>
<td>TMS</td>
<td>NO CONNECT</td>
</tr>
<tr>
<td>VRN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VREF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Pair</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.

---

**Figure 5-16:** FF672 Flip-Chip Fine-Pitch BGA Composite Pinout Diagram
FF672 Banked Pins (XC2VP7)

* Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.

**Figure 5-17:** FF672 Banked Pins (XC2VP7)
FF672 Dedicated/Other Pins (XC2VP7)

**Figure 5-18:** FF672 Dedicated Pins Diagram
Figure 5-19: FF672 Composite Pinout Diagram (XC2VP4)
FF672 Banked Pins (XC2VP4)

**Figure 5-20:** FF672 Banked Pins (XC2VP4)
FF672 Composite Pinout Diagram (XC2VP2)

**Figure 5-21:** FF672 Composite Pinout Diagram (XC2VP2)
Pinout Diagrams

FF672 Banked Pins (XC2VP2)

Figure 5-22: FF672 Banked Pins (XC2VP2)

* Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.

<table>
<thead>
<tr>
<th>User I/O Pins</th>
<th>Dedicated Pins</th>
<th>Other Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO_LXXY_#</td>
<td></td>
<td>VCCO</td>
</tr>
<tr>
<td>Dual-Purpose Pins:</td>
<td></td>
<td>NO CONNECT</td>
</tr>
<tr>
<td>DIN/DO-D7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDWR_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUSY/DOUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INIT_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCLKx (P)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCLKx (S)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VREF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Pair</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ug012_c4_110h_082502
Figure 5-23: FF672 Dedicated/Other Pins (XC2VP2)
**FF896 Flip-Chip Fine-Pitch BGA Pinout Diagrams**

**FF896 Composite Pinout Diagram (XC2VP20, XC2VP30)**

---

**Figure 5-24: FF896 Composite Pinout Diagram (XC2VP20, XC2VP30)**
FF896 Banked Pins (XC2VP20, XC2VP30)

**Figure 5-25:** FF896 Banked Pins (XC2VP20, XC2VP30)
FF896 Composite Pinout Diagram (XC2VP7)

Figure 5-26: FF896 Composite Pinout Diagram (XC2VP7)
FF896 Banked Pins (XC2VP7)

```
<table>
<thead>
<tr>
<th>User I/O Pins</th>
<th>Dedicated Pins</th>
<th>Other Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>● IO_LXY_#</td>
<td></td>
<td>VCCO</td>
</tr>
<tr>
<td>○ DIN/D0-D7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ CS_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ RDWR_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ BUSY/DOUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ INIT_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ GCLKx (P)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ GCLKx (S)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ VRP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ VRN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>○ VREF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>● No Pair</td>
<td></td>
<td>NO CONNECT</td>
</tr>
</tbody>
</table>
```

Figure 5-27: FF896 Banked Pins (XC2VP7)
FF896 (XC2VP7/20/30) - Top View

FF896 Dedicated/Other Pins (XC2VP7, XC2VP20, XC2VP30)

<table>
<thead>
<tr>
<th>Dedicated Pins</th>
<th>Other Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>CCLK</td>
<td>DXN</td>
</tr>
<tr>
<td>PROG_B</td>
<td>DXP</td>
</tr>
<tr>
<td>DONE</td>
<td>VBATT</td>
</tr>
<tr>
<td>M2, M1, M0</td>
<td>RSVD</td>
</tr>
<tr>
<td>HSWAP_EN</td>
<td>VCCAUX</td>
</tr>
<tr>
<td>TCK</td>
<td>VCCIINT</td>
</tr>
<tr>
<td>TDI</td>
<td>GND</td>
</tr>
<tr>
<td>TDO</td>
<td>NO CONNECT</td>
</tr>
<tr>
<td>TMS</td>
<td></td>
</tr>
<tr>
<td>PWRDWN_B</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AVCCAUXXRX</td>
</tr>
<tr>
<td></td>
<td>VTRXPAD</td>
</tr>
<tr>
<td></td>
<td>AVCCAUXTX</td>
</tr>
<tr>
<td></td>
<td>VTTXPAD</td>
</tr>
<tr>
<td></td>
<td>GNDA</td>
</tr>
<tr>
<td></td>
<td>RXNXPAD</td>
</tr>
<tr>
<td></td>
<td>RXPPAD</td>
</tr>
<tr>
<td></td>
<td>TXPPAD</td>
</tr>
<tr>
<td></td>
<td>TXNXPAD</td>
</tr>
</tbody>
</table>

Figure 5-28: FF896 Dedicated/Other Pins (XC2VP7, XC2VP20, XC2VP30)
**Chapter 5: PCB Design Considerations**

**Figure 5-29:** FF896 Composite Pinout Diagram (XC2VPX20)
FF896 Banked Pins (XC2VPX20)

Figure 5-30: FF896 Banked Pins (XC2VPX20)
Figure 5-31: FF896 Dedicated/Other Pins (XC2VPX20)
FF1152 Flip-Chip Fine-Pitch BGA Pinout Diagrams

FF1152 Composite Pinout Diagram (XC2VP50)

**Figure 5-32:** FF1152 Composite Pinout Diagram (XC2VP50)
FF1152 Dedicated/Other Pins (XC2VP50)

Figure 5-33: FF1152 Dedicated/Other Pins (XC2VP50)
FF1152 Composite Pinout Diagram (XC2VP40)

Figure 5-34: FF1152 Composite Pinout Diagram (XC2VP40)
FF1152 Banked Pins (XC2VP40, XC2VP50)

Figure 5-35: FF1152 Banked Pins (XC2VP40, XC2VP50)
Figure 5-36: FF1152 Dedicated/Other Pins (XC2VP40)
Figure 5-37: FF1152 Composite Pinout Diagram (XC2VP30)
Figure 5-38: FF1152 Banked Pins (XC2VP30)
FF1152 Dedicated/Other Pins (XC2VP30)

**Figure 5-39:** FF1152 Dedicated/Other Pins (XC2VP30)
Figure 5-40: FF1152 Composite Pinout Diagram (XC2VP20)
Figure 5-41: FF1152 Banked Pins (XC2VP20)
FF1152 Dedicated/Other Pins (XC2VP20)

**Figure 5-42:** FF1152 Dedicated/Other Pins (XC2VP20)
This page intentionally left blank.
FF1148 Flip-Chip Fine-Pitch BGA Pinout Diagrams

FF1148 Composite Pinout Diagram (XC2VP50)

Figure 5-43: FF1148 Composite Pinout Diagram (XC2VP50)
FF1148 Banked Pins (XC2VP50)

Figure 5-44: FF1148 Banked Pins (XC2VP50)
FF1148 Composite Pinout Diagram (XC2VP40)

**Figure 5-45:** FF1148 Composite Pinout Diagram (XC2VP40)
Chapter 5: PCB Design Considerations

Figure 5-46: FF1148 Banked Pins (XC2VP40)

* Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.

<table>
<thead>
<tr>
<th>User I/O Pins</th>
<th>Dedicated Pins</th>
<th>Other Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO_LXXY_#</td>
<td>VCCO</td>
<td></td>
</tr>
<tr>
<td>Dual-Purpose Pins:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DIN/D0-D7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CS_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>RDWR_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>BUSY/DOUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INIT_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCLKx (P)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>GCLKx (S)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>V/R</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VRN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VREF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Pair</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
FF1148 Dedicated/Other Pins (XC2VP40, XC2VP50)

Figure 5-47: FF1148 Dedicated/Other Pins (XC2VP40, XC2VP50)
This page intentionally left blank.
FF1517 Flip-Chip Fine-Pitch BGA Pinout Diagrams

FF1517 Composite Pinout Diagram (XC2VP70)

* Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.

<table>
<thead>
<tr>
<th>User I/O Pins</th>
<th>Dedicated Pins</th>
<th>Other Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO_LXXY_#</td>
<td>CCLK</td>
<td>DXN</td>
</tr>
<tr>
<td>Dual-Purpose Pins:</td>
<td>PROG_B</td>
<td>DXP</td>
</tr>
<tr>
<td>DIN/D0-D7</td>
<td>DONE</td>
<td>VBATT</td>
</tr>
<tr>
<td>CS_B</td>
<td>M2, M1, M0</td>
<td>RSVD</td>
</tr>
<tr>
<td>RDWR_B</td>
<td>HSWAP_EN</td>
<td>VCCO</td>
</tr>
<tr>
<td>BUSY/DOUT</td>
<td>TCK</td>
<td>VCCAUX</td>
</tr>
<tr>
<td>INIT_B</td>
<td>TDI</td>
<td>VCCINT</td>
</tr>
<tr>
<td></td>
<td>TDO</td>
<td>GND</td>
</tr>
<tr>
<td></td>
<td>TMS</td>
<td>NO CONNECT</td>
</tr>
<tr>
<td></td>
<td>PWRDWN_B</td>
<td></td>
</tr>
</tbody>
</table>

ug012_c4_107a_082202

Figure 5-48: FF1517 Composite Pinout Diagram (XC2VP70)
FF1517 Banked Pins (XC2VP70)

**User I/O Pins**
- IO_LXXY_#

**Dual-Purpose Pins:**
- DIN/D0-D7
- CS_B
- RDWR_B
- BUSY/DOUT
- INIT_B
- GCLKx (P)
- GCLKx (S)
- VRP
- VRN
- VREF
- No Pair

**Dedicated Pins**
- VCCO

**Other Pins**
- NO CONNECT

*Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.*

**Figure 5-49:** FF1517 Banked Pins (XC2VP70)
Figure 5-50: FF1517 Composite Pinout Diagram (XC2VP50)
**Chapter 5: PCB Design Considerations**

**Figure 5-51: FF1517 Banked Pins (XC2VP50)**

- **User I/O Pins**
  - Dual-Purpose Pins:
    - IO_LXXY_#
    - DIN/D0-D7
    - CS_B
    - RDWR_B
    - BUSY/DOUT
    - INIT_B
    - GCLKx (P)
    - GCLKx (S)
    - VRP
    - VRN
    - VREF
    - No Pair

- **Dedicated Pins**
  - VCCO

- **Other Pins**
  - NO CONNECT

---

Bank 1 - 81 Max User I/Os*

Bank 2 - 132 Max User I/Os*

Bank 3 - 132 Max User I/Os*

Bank 4 - 81 Max User I/Os*

Bank 5 - 81 Max User I/Os*

Bank 6 - 81 Max User I/Os*

Bank 7 - 132 Max User I/Os*

* Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.
FF1517 Dedicated/Other Pins (XC2VP50, XC2VP70)

Figure 5-52: FF1517 Dedicated/Other Pins (XC2VP50, XC2VP70)
This page intentionally left blank.
FF1704 Composite Pinout Diagram (XC2VP100)

Bank 1 - 104 Max User I/Os*
Bank 2 - 156 Max User I/Os*
Bank 3 - 156 Max User I/Os*
Bank 4 - 104 Max User I/Os*
Bank 5 - 104 Max User I/Os*
Bank 6 - 156 Max User I/Os*
Bank 7 - 156 Max User I/Os*

* Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.

<table>
<thead>
<tr>
<th>User I/O Pins</th>
<th>Dedicated Pins</th>
<th>Other Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>IO_LXXX_#</td>
<td>CCLK</td>
<td>DXN</td>
</tr>
<tr>
<td>Dual-Purpose Pins:</td>
<td>PROG_B</td>
<td>DXP</td>
</tr>
<tr>
<td>DIN/D0-D7</td>
<td>DONE</td>
<td>VBATT</td>
</tr>
<tr>
<td>CS_B</td>
<td>M2, M1, M0</td>
<td>RSVD</td>
</tr>
<tr>
<td>RDWR_B</td>
<td>HSWAP_EN</td>
<td>VCCO</td>
</tr>
<tr>
<td>BUSY/DOUT</td>
<td>TCK</td>
<td>VCCAUX</td>
</tr>
<tr>
<td>INIT_B</td>
<td>TDI</td>
<td>VCCINT</td>
</tr>
<tr>
<td>GCLKx (P)</td>
<td>TDO</td>
<td>GND</td>
</tr>
<tr>
<td>GCLKx (S)</td>
<td>TMS</td>
<td>NO CONNECT</td>
</tr>
<tr>
<td>VRP</td>
<td>PWRDWN_B</td>
<td></td>
</tr>
<tr>
<td>VRN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VREF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No Pair</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Figure 5-53:** FF1704 Composite Pinout Diagram (XC2VP100)
**Figure 5-54: FF1704 Banked Pins (XC2VP100)**

<table>
<thead>
<tr>
<th>User I/O Pins</th>
<th>Dedicated Pins</th>
<th>Other Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>• IO_LXXY_#</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• DIN/D0-D7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• CS_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• RDWR_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• BUSY/DOUT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• INIT_B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• GCLKx (P)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• GCLKx (S)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• VRP</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• VRN</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• VREF</td>
<td></td>
<td></td>
</tr>
<tr>
<td>• No Pair</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.

---

**FF1704 (XC2VP100/125) - Top View**

- Bank 1 - 104 Max User I/Os*
- Bank 0 - 104 Max User I/Os*
- Bank 2 - 156 Max User I/Os*
- Bank 7 - 156 Max User I/Os*
- Bank 3 - 156 Max User I/Os*
- Bank 6 - 156 Max User I/Os*
- Bank 5 - 104 Max User I/Os*
- Bank 4 - 104 Max User I/Os*
FF1704 Composite Pinout Diagram (XC2VP70)

* Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.

** User I/O Pins **
- IO_LXXY_
- DIN/D0-D7
- CS_B
- RDWR_B
- BUSY/DOUT
- INIT_B
- GCLKx (P)
- GCLKx (S)
- VRP
- VRN
- VREF
- No Pair

** Dedicated Pins **
- CCLK
- PROG_B
- DONE
- HSWAP_EN
- TCK
- TDI
- TDO
- TMS
- PWRDWN_B

** Other Pins **
- DXN
- DXP
- VBATT
- RSVD
- VCCO
- VCCAUX
- VCCINT
- GND
- NO CONNECT

---

** Figure 5-55: FF1704 Composite Pinout Diagram (XC2VP70) **
FF1704 Banked Pins (XC2VP70)

Bank 1 - 93 Max User I/Os*

Bank 2 - 156 Max User I/Os*

Bank 3 - 156 Max User I/Os*

Bank 4 - 93 Max User I/Os*

Bank 5 - 93 Max User I/Os*

Bank 6 - 156 Max User I/Os*

Bank 7 - 156 Max User I/Os*

* Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.

Figure 5-56: FF1704 Banked Pins (XC2VP70)
FF1704 Dedicated/Other Pins (XC2VP70, XC2VP100)

Figure 5-57: FF1704 Dedicated/Other Pins (XC2VP70, XC2VP100)
FF1704 Composite Pinout Diagram (XC2VPX70)

**Figure 5-58: FF1704 Composite Pinout Diagram (XC2VPX70)**
**FF1704 Banked Pins (XC2VPX70)**

**FF1704 (XC2VPX70) - Top View**

- **Bank 1** - 93 Max User I/Os*
- **Bank 2** - 156 Max User I/Os*
- **Bank 3** - 156 Max User I/Os*
- **Bank 4** - 93 Max User I/Os*
- **Bank 5** - 91 Max User I/Os*
- **Bank 6** - 156 Max User I/Os*
- **Bank 7** - 156 Max User I/Os*

**User I/O Pins**
- IO_LXXY_
- Dual-Purpose Pins:
  - DIN/D0-D7
  - CS_B
  - RDWR_B
  - BUSY/DOUT
  - INIT_B
  - GCLKx (P)
  - GCLKx (S)
  - VRP
  - VRN
  - VREF
  - No Pair

**Dedicated Pins**
- VCCO

**Other Pins**
- NO CONNECT

* Callouts for max number of I/Os per bank include dual-purpose pins shown in table, right.

**Figure 5-59: FF1704 Composite Pinout Diagram (XC2VPX70)**
Figure 5-60: FF1704 Composite Pinout Diagram (XC2VPX70)
Figure 5-61: FF1696 Composite Pinout Diagram (XC2VP100)
Figure 5-62: FF1696 Banked Pins (XC2VP100)
FF1696 Dedicated/Other Pins (XC2VP100)

FF1696 (XC2VP100/125) - Top View

Dedicated Pins:
- CCLK
- PROG_B
- DONE
- M2, M1, M0
- HSWAP_EN
- TCK
- TDI
- TDO
- TMS
- PWRDWN_B

Other Pins:
- DXN
- DXP
- VBATT
- VCCINT
- GND
- NO CONNECT
- AVCCAUXXRX
- VTRXPAD
- AVCCAUXTX
- VTTXPAD
- GNDA
- RXNPD
- RXPPAD
- TXPPAD
- TXNPD

Figure 5-63: FF1696 Dedicated/Other Pins (XC2VP100)
Package Specifications

This section contains specifications for the following Virtex-II Pro packages:

- FG256/FGG256 Fine-Pitch BGA Package (1.00 mm Pitch), page 468
- FG456/FGG456 Fine-Pitch BGA Package (1.00 mm Pitch), page 469
- FG676/FGG676 Fine-Pitch BGA Package (1.00 mm pitch), page 470
- FF672 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch), page 471
- FF896 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch), page 472
- FF1152 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch), page 473
- FF1148 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch), page 474
- FF1517 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch), page 475
- FF1704 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch), page 476
- FF1696 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch), page 477
FG256/FGG256 Fine-Pitch BGA Package (1.00 mm Pitch)

Figure 5-64: FG256/FGG256 Fine-Pitch BGA Package
FG456/FGG456 Fine-Pitch BGA Package (1.00 mm Pitch)

Figure 5-65: FG456/FGG456 Fine-Pitch BGA Package

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MILLIMETERS</th>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>A₁</td>
<td>0.35 0.50 0.60</td>
<td></td>
</tr>
<tr>
<td>D/E</td>
<td>23.00 BSC</td>
<td></td>
</tr>
<tr>
<td>D/E₁</td>
<td>21.00 REF</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>1.00 BSC</td>
<td></td>
</tr>
<tr>
<td>φb</td>
<td>0.50 0.60 0.70</td>
<td></td>
</tr>
<tr>
<td>aqq</td>
<td></td>
<td>22</td>
</tr>
<tr>
<td>ccc</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ddd</td>
<td>0.30 0.30 0.30</td>
<td></td>
</tr>
<tr>
<td>eee</td>
<td>0.10</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:

2. SYMBOL 'M' IS THE BALL MATRIX SIZE.
3. NOMINAL 'A' DIMENSION FOR 2-LAYER IS 2.03mm AND FOR 4-LAYER IS 2.20mm.
4. CONFORMS TO JEDEC MS-034-AAJ-1 (DEPOPULATED).
Chapter 5: PCB Design Considerations

FG676/FGG676 Fine-Pitch BGA Package (1.00mm pitch)

Figure 5-66: FG676/FGG676 Fine-Pitch BGA Package Specifications
**FF672 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch)**

*Figure 5-67: FF672 Flip-Chip Fine-Pitch BGA Package*

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MILLIMETERS</th>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>2.45</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>2.65</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>0.40</td>
<td>0.50</td>
</tr>
<tr>
<td>A2</td>
<td>2.15</td>
<td></td>
</tr>
<tr>
<td>D/E</td>
<td>27.00 BASIC</td>
<td></td>
</tr>
<tr>
<td>D/Ei</td>
<td>25.00 REF</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>1.00 BASIC</td>
<td></td>
</tr>
<tr>
<td>Øb</td>
<td>0.50</td>
<td>0.60</td>
</tr>
<tr>
<td>aaa</td>
<td>0.20</td>
<td></td>
</tr>
<tr>
<td>ccc</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>add</td>
<td>0.30</td>
<td></td>
</tr>
<tr>
<td>eee</td>
<td>0.10</td>
<td></td>
</tr>
<tr>
<td>M</td>
<td>26</td>
<td></td>
</tr>
</tbody>
</table>

NOTES:
1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M-1994
2. SYMBOL 'M' IS THE BALL MATRIX SIZE.
3. CONFORMS TO JEDEC MS-034-AAL-1 (DEPOPULATED)
FF896 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch)

Figure 5-68: FF896 Flip-Chip Fine-Pitch BGA Package

NOTES:
1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M-1994
2. SYMBOL 'M' IS THE BALL MATRIX SIZE.
3. CONFORMS TO JEDEC MS-034-AAN-1 (DEPOPULATED)
## FF1152 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch)

### Package Specifications

#### FF1152 Flip-Chip Fine-Pitch BGA Package

![FF1152 Flip-Chip Fine-Pitch BGA Package](image.png)

#### Table: FF1152 Flip-Chip Fine-Pitch BGA Package Dimensions

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Minimum</th>
<th>Nominal</th>
<th>Maximum</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3.20</td>
<td>3.40</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>0.40</td>
<td>0.50</td>
<td>0.60</td>
</tr>
<tr>
<td>A2</td>
<td></td>
<td></td>
<td>2.80</td>
</tr>
<tr>
<td>D/E</td>
<td>35.00</td>
<td>BASIC</td>
<td></td>
</tr>
<tr>
<td>D1/E1</td>
<td>33.00</td>
<td>REF</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>1.00</td>
<td>BASIC</td>
<td></td>
</tr>
<tr>
<td>φb</td>
<td>0.50</td>
<td>0.60</td>
<td>0.70</td>
</tr>
<tr>
<td>αa</td>
<td></td>
<td></td>
<td>0.20</td>
</tr>
<tr>
<td>βcc</td>
<td></td>
<td></td>
<td>0.35</td>
</tr>
<tr>
<td>βdd</td>
<td></td>
<td></td>
<td>0.30</td>
</tr>
<tr>
<td>βee</td>
<td></td>
<td></td>
<td>0.10</td>
</tr>
<tr>
<td>M</td>
<td>34</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

#### Notes:

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M–1994
2. SYMBOL ‘M’ IS THE BALL MATRIX SIZE.
3. CONFORMS TO JEDEC MS–034–AAR–1 (DEPOPULATED)

---

**Figure 5-69:** FF1152 Flip-Chip Fine-Pitch BGA Package
FF1148 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch)

Figure 5-70: FF1148 Flip-Chip Fine-Pitch BGA Package

NOTES:
1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M-1994
2. SYMBOL 'M' IS THE BALL MATRIX SIZE.
3. CONFORMS TO JEDEC MS-034-AAR-1 (DEPOPULATED)
FF1517 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch)

**Figure 5-71:** FF1517 Flip-Chip Fine-Pitch BGA Package

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MILLIMETERS</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3.20 3.40</td>
</tr>
<tr>
<td>A1</td>
<td>0.40 0.50 0.60</td>
</tr>
<tr>
<td>A2</td>
<td>2.80</td>
</tr>
<tr>
<td>D/E</td>
<td>40.00 BASIC</td>
</tr>
<tr>
<td>D/E1</td>
<td>38.00 REF</td>
</tr>
<tr>
<td>e</td>
<td>1.00 BASIC</td>
</tr>
<tr>
<td>øb</td>
<td>0.50 0.60 0.70</td>
</tr>
<tr>
<td>lobb</td>
<td>0.20</td>
</tr>
<tr>
<td>lodd</td>
<td>0.25</td>
</tr>
<tr>
<td>eee</td>
<td>0.25</td>
</tr>
<tr>
<td>M</td>
<td>39 2</td>
</tr>
</tbody>
</table>

**NOTES:**

1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ASME Y14.5M-1994
2. SYMBOL "M" IS THE PIN MATRIX SIZE.
3. CONFORMS TO JEDEC MS-034-AAU-1 (DEPOPULATED)
FF1704 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch)

Figure 5-72: FF1704 Flip-Chip Fine-Pitch BGA Package

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MILLIMETERS</th>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>MIN. 3.20</td>
<td>NOM. 3.45</td>
</tr>
<tr>
<td>A1</td>
<td>0.40</td>
<td>0.50</td>
</tr>
<tr>
<td>A2</td>
<td>2.85</td>
<td></td>
</tr>
<tr>
<td>D/E</td>
<td>42.50 BASIC</td>
<td></td>
</tr>
<tr>
<td>D2/E1</td>
<td>41.00 REF</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>100 BASIC</td>
<td></td>
</tr>
<tr>
<td>φ0</td>
<td>0.50</td>
<td>0.60</td>
</tr>
<tr>
<td>φ1</td>
<td></td>
<td>0.70</td>
</tr>
<tr>
<td>oo</td>
<td></td>
<td>0.20</td>
</tr>
<tr>
<td>bbbb</td>
<td></td>
<td>0.25</td>
</tr>
<tr>
<td>ddd</td>
<td></td>
<td>0.25</td>
</tr>
<tr>
<td>eee</td>
<td></td>
<td>0.10</td>
</tr>
<tr>
<td>M</td>
<td>42</td>
<td>2</td>
</tr>
</tbody>
</table>

NOTES:
1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ASME Y14.5M-1994
2. SYMBOL "M" IS THE PIN MATRIX SIZE.
3. CONFORMS TO JEDEC MS-034-AAV-1 (DEPOPULATED)
FF1696 Flip-Chip Fine-Pitch BGA Package (1.00 mm Pitch)

Figure 5-73: FF1696 Flip-Chip Fine-Pitch BGA Package
Flip-Chip Packages

As silicon devices become more integrated with smaller feature sizes as well as increased functionality and performance, packaging technology is also evolving to take advantage of these silicon advancements. Flip-chip packaging is the latest packaging option introduced by Xilinx to meet the demand for high I/O count and high performance required by today's advanced applications.

Flip-chip packaging interconnect technology replaces peripheral bond pads of traditional wire-bond interconnect technology with area array interconnect at the die/substrate interface.

The area array pads contain wettable metallization for solders (either eutectic or high-lead), where a controlled amount of solder is deposited either by plating or screen-printing. These parts are then reflowed to yield bumped dies with relatively uniform solder bumps spread over the surface of the device. Unlike traditional packaging in which the die is attached to the substrate face up and the connection is made by using wire, the bumped die in a flip-chip package is flipped over and placed face down, with the conductive bumps connecting directly to the matching metal pads on the ceramic or organic laminate substrate. The solder material at molten stage is self-aligning and produces good joints even if the chip is placed offset on the substrate.

Flip-chip packages are assembled on high-density, multi-layer ceramic or organic laminate substrates. Since flip-chip bump pads are in area array configuration, very fine lines and geometry on the substrates are required to be able to successfully route the signals from the die to the periphery of the substrates. Multi-layer build-up structures offer this layout flexibility on flip-chip packages, and they provide improvements in power distribution and signal transmission characteristics.

Advantages of Flip-Chip Technology

Flip-chip interconnections in combination with the advanced multi-layer laminated substrates provide superior performance over traditional wire-bond packaging. Benefits include:

- Easy access to core power/ground and shorter interconnects, resulting in better electrical performance
- Better noise control since the inductance of flip-chip interconnect is lower
- Excellent thermal performance due to direct heatsinking to backside of the die
- Higher I/O density since bond pads are in area array format
- Smaller size

Thermal Data

Thermal Considerations

The Virtex-II Pro device is a feature-rich FPGA product based on the high-performance Virtex-II architecture. The product incorporates numerous features such as multiple RocketIO™ Multi-Gigabit Transceivers (MGTs), one or more embedded IBM PowerPC processors, high-speed SelectI/O™ technology supporting a variety of I/O standards, on-board digitally controlled impedance (DCI) technology, and much more. In fully configured designs that engage all these features at high clock rates, power consumption can add up quickly.
Unlike the features of ASICs or even of microprocessors, the combination of Virtex-II Pro features that will be utilized in an application are not known ahead of time. Therefore, as in previous FPGA devices, it remains challenging to predict the power requirements and resulting thermal management needs of a Virtex-II Pro device in a given package. These devices, therefore, do not come with a preset thermal solution. The end user’s conditions will determine what solutions will be most appropriate. In consideration of the high heat-generating potential of the Virtex-II Pro devices, package offerings are tailored to include medium and high power capable options that allow external thermal management to meet the application’s requirements.

Table 5-3 shows thermal resistance parameters for Virtex-II Pro packages. Estimated power consumption capability is given, as well. These values were derived using some typical thermal management assumptions as stated in the table. Table 5-3 provides only an example, and is not intended as a maximum power dissipation specification.

### Table 5-3: Thermal Data for Virtex-II Pro Packages

<table>
<thead>
<tr>
<th>Package</th>
<th>Lead Pitch (mm)</th>
<th>Junction-to-Ambient θJA, in Air (°C/Watt)</th>
<th>Junction-to-Case θJC, Cold Plate (°C/Watt)</th>
<th>Junction-to-Board ΨJB (°C/Watt)</th>
<th>Max Power, Bare Pkg (Watts)</th>
<th>Power With Heatsink (Watts)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FG256/FGG256</td>
<td>2.4L PCB, 17x17</td>
<td>24 - 28</td>
<td>3.0 - 5.0</td>
<td>14.0</td>
<td>1.7</td>
<td>N/A (1)</td>
</tr>
<tr>
<td>FG456/FGG456</td>
<td>4L PCB, 23x23</td>
<td>18 - 22</td>
<td>2.0 - 2.5</td>
<td>10.0</td>
<td>2.5</td>
<td>N/A (1)</td>
</tr>
<tr>
<td>FG676/FGG676</td>
<td>4L PCB, 26x26</td>
<td>16 - 18</td>
<td>2.0 - 2.5</td>
<td>7.6</td>
<td>3.0</td>
<td>N/A (1)</td>
</tr>
<tr>
<td>FF672, 27x27 Flip-Chip</td>
<td>1.0</td>
<td>14 - 16</td>
<td>0.5</td>
<td>6.5</td>
<td>3.3</td>
<td>16</td>
</tr>
<tr>
<td>FF896, 31x31 Flip-Chip</td>
<td>1.0</td>
<td>10 - 12</td>
<td>0.5</td>
<td>3.7</td>
<td>4.5</td>
<td>21</td>
</tr>
<tr>
<td>FF1152, 35x35 Flip-Chip</td>
<td>1.0</td>
<td>10 - 12</td>
<td>0.5</td>
<td>3.7</td>
<td>4.5</td>
<td>21</td>
</tr>
<tr>
<td>FF1148, 35x35 Flip-Chip</td>
<td>1.0</td>
<td>10 - 12</td>
<td>0.5</td>
<td>3.7</td>
<td>4.5</td>
<td>21</td>
</tr>
<tr>
<td>FF1517, 40x40 Flip-Chip</td>
<td>1.0</td>
<td>10 - 12</td>
<td>0.5</td>
<td>3.7</td>
<td>4.5</td>
<td>21</td>
</tr>
<tr>
<td>FF1704, 42.5x42.5 Flip-Chip</td>
<td>1.0</td>
<td>10 - 12</td>
<td>0.5</td>
<td>3.7</td>
<td>4.5</td>
<td>21</td>
</tr>
<tr>
<td>FF1696, 42.5x42.5 Flip-Chip</td>
<td>1.0</td>
<td>10 - 12</td>
<td>0.5</td>
<td>3.7</td>
<td>4.5</td>
<td>21</td>
</tr>
</tbody>
</table>

θJA (Theta-JA): Thermal resistance, junction-to-ambient (Xilinx-supplied)

θJC (Theta-JC): Thermal resistance, junction-to-case (use with ratio of heat through the top)

ΨJB (Psi-JB, or "Theta-JB"): Thermal resistance in still air, junction-to-board

θSA (Theta-SA): Thermal resistance, heatsink (manufacturer-supplied)

θCS (Theta-CS): Thermal resistance, heatsink adhesive (manufacturer-supplied)

Notes:

1. The heat sink used in this example is not mechanically compatible with the FG256/FGG256, FG456/FGG456, and FG676/FGG676 packages.

Virtex-II Pro packages can be grouped into three broad performance categories: low, medium, and high, based on their power handling capabilities. All of the packages can use external thermal enhancements, which can range from simple airflow to schemes that can include passive as well as active heatsinks. This is particularly true for high-performance flip-chip packages where system designers have the option to further enhance the packages to handle in excess of 20 watts, with arrangements that take system physical
constraints into consideration. Table 5-4 shows simple but incremental power management schemes that can be brought to bear on flip-chip packages.

Table 5-4: Virtex-II Pro Flip-Chip Thermal Management

<table>
<thead>
<tr>
<th>Power</th>
<th>Technique</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low End (1 - 6 watts)</td>
<td>Bare package with moderate air 8 - 12 °C/Watt</td>
<td>Bare package. Package can be used with moderate airflow within a system.</td>
</tr>
<tr>
<td>Mid Range (4 - 10 watts)</td>
<td>Passive heatsink with air 5 - 10 °C/Watt</td>
<td>Package is used with various forms of passive heatsinks and heat spreader techniques.</td>
</tr>
<tr>
<td>High End (8 - 25 watts)</td>
<td>Active heatsink 2 - 3 °C/Watt or better</td>
<td>Package is used with active heatsinks, TEC, and board-level heat spreader techniques.</td>
</tr>
</tbody>
</table>

Thermal Management Options

The following are thermal management options to consider:

- For moderate power dissipation (2 to 6 watts), the use of passive heatsinks and heatspreaders attached with thermally conductive double-sided tapes or retainers can offer quick thermal solutions.

- The use of lightweight finned external passive heatsinks can be effective for dissipating up to 10 watts. The more efficient external heatsinks tend to be tall and heavy. To help protect component joints from bulky heatsink-induced stresses, the use of spring loaded pins or clips that transfer the mounting stress to a circuit board is advisable. The diagonals of some of these heatsinks can be designed with extensions to allow direct connections to the board.

- Flip-chip packages: All flip-chip packages are thermally enhanced BGAs with die facing down. They are offered with exposed metal heatsink at the top. These high-end thermal packages lend themselves to the application of external heatsinks (passive or active) for further heat removal efficiency. Again, precaution should be taken to prevent component damage when a bulky heatsink is attached.

- Active heatsinks can include a simple heatsink incorporating a mini fan or even a Peltier Thermoelectric Cooler (TECs) with a fan to blow away any heat generated. Any considerations to apply TEC in heat management should require consultation with experts in using the device, since these devices can be reversed and cause damage to the components. Also, condensation can be an issue.

- Outside the package itself, the board on which the package sits can have a significant impact on thermal performance. Board designs can be implemented to take advantage of a board's ability to spread heat. The effect of the board is dependent on its size and how it conducts heat. Board size, the level of copper traces on it, and the number of buried copper planes all lower the junction-to-ambient thermal resistance for packages mounted on the board.

The junction-to-board thermal resistance for Virtex-II Pro packages are given in Table 5-3. A standard JEDEC type board was used for obtaining the data. Users need to be aware that a direct heat path to the board from a component also exposes the component to the effect of other heat sources - particularly if the board is not cooled effectively. An otherwise cooler component might be heated by other heat contributing components on the board.
Printed Circuit Board Considerations

Layout Considerations

The PC board is no longer just a means to hold ICs in place. At today’s high clock rates and fast signal transitions, the PC board performs a vital function in feeding stable supply voltages to the IC and in maintaining signal integrity between devices.

VCC and Ground Planes

Since CMOS power consumption is dynamic, it is a non-trivial task to assure stable supply voltages at the device pins and to minimize ground differentials. A multi-layer PC board is a must, with four layers for the simplest circuits, 6 to 12 layers for typical boards. Ground and VCC must each be distributed in complete layers with few holes. Slots in these layers would cause an unacceptable inductive voltage drop, when the supply current changes at a rate of 1 A/ns, or even faster. Besides an uninterrupted ground plane, Virtex-II Pro devices require one plane for VCCINT (1.5V) plus one plane for VCCAUX (2.5V). VCCO can be distributed on wide sections of split plane layers. Note that signal traces on adjacent layers should not be routed across these plane splits.

Beyond low resistance and inductance, ground and VCC planes combined can also provide a small degree of VCC decoupling. The capacitance between two planes is ~180 pF/inch² or ~28 pF/cm², assuming 5mil (0.125 mm) spacing with FR4 epoxy.

VCC Decoupling

Fast changing ICC transitions must be supplied by local decoupling capacitors, placed very closely to the VCC device pins or balls. These capacitors must have sufficient capacitance to supply ICC for a few nanoseconds, and must have low intrinsic resistance and inductance. X7R or NPO ceramic surface-mounted capacitors of 0.01 to 0.1 µF, one per VCC device pin, are appropriate. 0.1 µF can supply 1A for 2 ns with a 20 mV voltage droop.

\[ 1A \times 2 \text{ ns} = 2 \text{ nanocoulomb} = 100 \text{ nF} \times 0.02V \]

Low impedance at >100 MHz is important. Some capacitance variation with temperature is acceptable. A series of values of decoupling capacitors are needed in order to supply transient current to the device over all frequency ranges. The highest frequency capacitors should be in the range of 300 pF to 1 nF. The smallest capacitors are the first-line source for ICC and they must be placed very close to the VCC pins. An inch of copper plane represents an inductance of several nanohenries, defeating the purpose of the decoupling capacitor. Therefore, the highest frequency (smallest value) capacitors should be mounted within 0.4 inches (1 cm) of the VCC and ground pin pair that it is decoupling. On large full-array BGA packages like the FF1152 and FF1517, following the decoupling guidelines presents implementation challenges. Note that the power and ground balls on these packages are grouped in banks. A subset of the initial line of high-frequency decoupling capacitors should be mounted directly opposite the component on the reverse side of the board and within the apparent boundary of the component. At least one and typically two 0.1 µF chipcaps per bank should be used. The balance of the high-frequency bypass requirements can be implemented outside the periphery on these full-array packages. Backing up this local decoupling is one tantalum capacitor of 10 µF to 100 µF, able to supply multiple amperes for about 100 ns.

Finally, each board needs a power-supply decoupling electrolytic capacitor of 1000 µF to 10,000 µF, able to supply even more current for a portion of the supply switching period. As described below, larger capacitors inevitably have higher series resistance and
inductance, which is the reason for the above-mentioned hierarchy of supply decoupling. As a general rule, multiple capacitors in parallel always offer lower resistance and inductance than any single capacitor.

Bypass capacitors must have vias (at least two per pad) abutting the pads, or vias in the pads, to provide the lowest inductance connection.

**Decoupling Capacitors**

The ideal decoupling capacitor would present a short circuit to ground for all AC signals. A real capacitor combines a given amount of capacitance with unavoidable parasitics, a small series resistance, and inductance. At low frequencies, the composite impedance is capacitive: that is, it decreases with increasing frequency. At high frequencies, it is inductive and increases with frequency, making the decoupling ineffective. In between, there is the LC resonant frequency, where the capacitor looks like a small resistor. This is the range where it is most effective.

Different technologies provide different trade-offs between desirable features like small size and high capacitance, and undesirable features like series resistance and inductance. Electrolytic and tantalum capacitors offer the largest capacitance in a given physical size, but also have the highest inductance. This makes them useful for decoupling low frequencies and storing large amounts of charge, but useless for high frequency decoupling. Surface-mount ceramic capacitors, on the other hand, offer the lowest inductance and the best high-frequency performance, but offer only a small amount of capacitance, usually less than a microfarad.

**Figure 5-74** shows the frequency-dependent impedance and resistance of a typical electrolytic capacitor of 1500 µF, while **Figure 5-75** and **Figure 5-76** show the equivalent data for ceramic bypass capacitors of 33,000 pF and 3,300 pF, respectively. Note that the resonant frequency for the small ceramic bypass capacitor at 100 MHz is 10,000 times higher than the resonance frequency of the large electrolytic capacitor at 10 KHz. For more technical information on decoupling capacitors, see the manufacturers’ websites.

![Impedance (Ohms)](image-url)

**Figure 5-74:** **1500 µF Electrolytic Capacitor Frequency Response Curve**
Transmission Line Reflections and Terminations

A PC board trace must be analyzed as a transmission line. Its series resistance and parallel conductance can generally be ignored, but series inductance and parallel capacitance per unit length are important parameters. Any signal transition (rising or falling edge) travels along the trace at a speed determined by the incremental inductance and capacitance.

Figure 5-75: 33000 pF X7R Component Frequency Response Curve

Figure 5-76: 3300 pF X7R Component Frequency Response Curve
For an outer-layer trace (air on one side), the propagation delay is 140 ps/inch, or 55 ps/cm. For an inner-layer trace (FR4 with \( \varepsilon = 4.5 \) on both sides), the propagation delay is 180 ps/inch, or 70 ps/cm.

The voltage-to-current ratio at any point along the transmission line is called the characteristic impedance \( Z_0 \). A field solver, which is available in most signal integrity simulation software, may be used to determine the impedance of a given trace geometry. As a first order approximation, it is determined by \( w/d \), the ratio of trace width \( w \) to the distance \( d \) above the ground or \( V_{CC} \) plane.

For an outer layer trace (microstrip),
\[
Z_0 = 50 \Omega \quad \text{when} \quad w = 2d \quad (e.g., \quad w = 12 \text{ mil}, \quad d = 6 \text{ mil}),
\]
\[
Z_0 = 75 \Omega \quad \text{when} \quad w = d \quad (e.g., \quad \text{both} \quad 6 \text{ mil} = 0.15 \text{ mm}).
\]

For an inner layer trace between two ground or \( V_{CC} \) planes (strip line),
\[
Z_0 = 50 \Omega \quad \text{when} \quad w = 0.6 \cdot d \quad (e.g., \quad w = 5 \text{ mil}, \quad d = 8 \text{ mil}),
\]
\[
Z_0 = 75 \Omega \quad \text{when} \quad w = 0.25 \cdot d \quad \text{(impractical)}.
\]

Most signal traces fall into the range of 40\( \Omega \) to 80\( \Omega \).

A slow transition treats a short narrow trace as a lumped capacitance of about 2 pF/inch (0.8 pF/cm). However, if the trace is so long or the signal transition so fast that the potential echo from the far end arrives after the end of the transition, then the trace must be analyzed as a transmission line.

In this case, the driver sees the trace not as a lumped capacitance, but rather as a pure resistance of \( Z_0 \). The signal transition then travels along the trace at the speed mentioned above. At any trace-impedance discontinuity all or part of the signal is reflected back to the origin. If the far end is resistively terminated with \( R = Z_0 \), then there is no reflection. If, however, the end is open, or loaded with only a CMOS input, then the transition doubles in amplitude, and this new wave travels back to the driver, where it may be reflected again, resulting in ringing. Such ringing has a serious impact on signal integrity, reduces noise margins, and can lead to malfunction, especially if an asynchronous signal or a clock signal crosses the input threshold voltage unpredictably. Two alternate ways to avoid reflections and ensure signal integrity are parallel termination and series termination.

**Parallel Termination**

Reflections from the far end of the transmission line are avoided if the far end is loaded with a resistor equal to \( Z_0 \). A popular variation uses two resistors, one to \( V_{CC} \), one to ground, as the Thevenin equivalent of \( Z_0 \). This reduces the load current for one signal level, while increasing it for the other. Parallel termination inherently has some dc power consumption. DCI on-chip termination may be used to realize parallel termination schemes.

See Figure 5-77.
Series Termination

While parallel termination eliminates reflections, series termination relies on the reflection from the far end to achieve a full-amplitude signal. For series termination, the driver impedance is adjusted to equal $Z_0$, thus driving a half-amplitude signal onto the transmission line. At the unterminated far end, the reflection creates a full-amplitude signal, which then travels back to the driver where it is absorbed, since the output impedance equals $Z_0$. See Figure 5-78.

Series termination dissipates no dc power, but the half-amplitude round-trip delay signal means that there must be no additional loads along the line. Series termination is ideal (and only meaningful) for single-source-single-destination interconnects. DCI on-chip termination may be used to realize series termination schemes through the use of controlled impedance drivers.

DCI On-Chip Termination

Virtex-II Pro devices offer digitally controlled output impedance drivers and digitally-controlled input termination, thus eliminating the need for any external termination resistors. This feature is extremely valuable with high pin-count, high density packages. These PC board considerations apply to all modern systems with fast current and voltage transitions, irrespective of the actual clock frequency. The designer of relatively slow systems is more likely caught off-guard by the inherent speed of modern CMOS ICs, where $di/dt$ is measured in $A/ns$, $dV/dt$ is measured in $V/ns$, and input flip-flops can react to 1 ns pulses, that are invisible on mid-range oscilloscopes. Powerful tools like HyperLynx and other signal integrity analysis tools can be used to analyze signal integrity on the PC board and can often be amortized by one eliminated board respin.
JTAG Configuration and Test Signals

Poor signal integrity and limitations of devices in a JTAG scan chain can reduce the maximum JTAG test clock (TCK) rate and reliability of JTAG-based configuration and test procedures. The JTAG TCK and test mode (TMS) signals must be buffered, distributed, and routed with the same care as any clock signal especially for long JTAG scan chains. The devices in a JTAG scan chain should be ordered such that the connections from the TDO of one device to the TDI of the next device are minimized. When high-speed JTAG-based configuration for the Virtex-II Pro devices is required, devices with lower-specified maximum TCK rates should be placed in a separate JTAG scan chain.

Crosstalk

Crosstalk can happen when two signals are routed closely together. Current through one of the traces creates a magnetic field that induces current on the neighboring trace, or the voltage on the trace couples capacitively to its neighbor. Crosstalk can be accurately modeled with signal integrity software. Two easy-to-remember rules of thumb are:

- Crosstalk falls off with the square of increasing distance between the traces.
- Crosstalk also falls off with the square of decreasing distance to a ground plane.

\[
\text{Peak Crosstalk Voltage} = \frac{\Delta V}{1 + (D/H)^2}
\]

where

- \(\Delta V\) is the voltage swing
- \(D\) is the distance between traces (center to center)
- \(H\) is the spacing above the ground plane

Example:

3.3V swing, and two stripline traces 5 mils apart and 5 mils above the ground plane.

\[
\text{Peak Crosstalk Voltage} = \frac{(3.3V)}{(1 + (0.005/0.005)^2)} = 1.65V
\]

This can cause a false transition on the neighboring trace. Separating the trace by an additional 5 mils is significantly better:

\[
\text{Peak Crosstalk Voltage} = \frac{(3.3V)}{(1 + (0.01/0.005)^2)} = 0.66V
\]

Decoupling of V\(_{REF}\) Pins

V\(_{REF}\) pins must each have a bypass capacitor to insure that no noise is coupled onto the reference. Care should be taken to minimize crosstalk between signal traces and V\(_{REF}\) traces by using adequate inter-trace spacing and avoiding long parallel runs. Routing V\(_{REF}\) traces close to noisy buses (e.g., PCI) should be avoided if possible.

Signal Routing to and from Package Pins

Signal escaping (traces leaving the pin/ball area) can be quite difficult for the large FG and flip-chip packages. The number of signal layers required to escape all the pins depends on the PCB design rules. The thinner the traces, the more signals per layer can be routed, and the fewer layers are needed. The thinner traces have higher characteristic impedance, so choose an impedance plan that makes sense, and then be consistent. Traces from 40\(\Omega\) to 80\(\Omega\) are common.
If only one signal can be escaped between two pads, only two rows of pins can be escaped per layer. For FG packages (1.0mm pitch) one signal of width 5 mils (0.13mm) can be escaped between two pads, assuming a space constraint equal to the trace width. For a discussion of signal routing specific to Virtex-II Pro devices, see [www.xilinx.com](http://www.xilinx.com) for currently available application notes.

As packages are able to handle more I/Os with a minimum increase in size, the signal integrity of those signals must be considered, regardless of clock frequency. Especially with the largest packages, precise PCB layer stackup is required. Parameters such as board material, trace width, pad type, and stackup must be defined based on simulation, and the fabrication drawings must be marked “Precise Layer Stackup,” and the stackup must be specified. A number of board-level signal integrity simulators exist. Careful attention to PCB design rules creates a robust design with low EMI and high signal reliability.

**Board Routability Guidelines**

**Board-Level BGA Routing Challenges**

Xilinx ball grid array (BGA) wire-bond and flip-chip packages contain a matrix of solder balls (see Figure 5-79). These packages are made of multilayer BT substrates. Signal balls are in a perimeter format. Power and ground pins are grouped together appropriately.

![Figure 5-79: Fine-Pitch BGA Pin Assignments](image)

The number of layers required for effective routing of these packages is dictated by the layout of balls in each package. If several other technologies and components are already present on the board, the system cost is factored with every added board layer. The intent of a board designer is to optimize the number of layers required to route these packages, considering both cost and performance. This section provides guidelines for minimizing required board layers for routing BGA products using standard PCB technologies (5 mil wide lines and spaces or 6 mil wide lines and spaces).

For high performance and other system needs, designers can use premium technologies with finer lines/spaces on the board. The pin assignment and pin grouping scheme in BGA packages enables efficient routing of the board with an optimum number of required board layers.
Board Routing Strategy

The diameter of a land pad on the component side is provided by Xilinx. This information is required prior to the start of board layout when designing the board pads to match component-side land geometry. Typical values for these land pads are described in Figure 5-80 and summarized in Table 5-5.

![Figure 5-80: Suggested Board Layout of Soldered Pads for BGA Packages](x157_02_120500)

Non-Solder-Mask-Defined Land Patterns or Land-Defined Land Patterns are recommended for all packages.

Table 5-5: Summary of Typical Land Pad Values (mm)

<table>
<thead>
<tr>
<th>Land Pad Characteristics</th>
<th>FG256/FGG256</th>
<th>FG456/FGG456</th>
<th>FG676/FGG676</th>
<th>FF672</th>
<th>FF896</th>
<th>FF1152</th>
<th>FF1148</th>
<th>FF1517</th>
<th>FF1704</th>
<th>FF1696</th>
</tr>
</thead>
<tbody>
<tr>
<td>Component Land Pad Diameter (SMD)(4)</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
<td>0.48</td>
<td>0.48</td>
<td>0.48</td>
<td>0.48</td>
<td>0.48</td>
<td>0.48</td>
<td>0.48</td>
</tr>
<tr>
<td>Solder Land (L) Diameter</td>
<td>0.40</td>
<td>0.40</td>
<td>0.40</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
<td>0.45</td>
</tr>
<tr>
<td>Opening in Solder Mask (M) Diameter</td>
<td>0.50</td>
<td>0.50</td>
<td>0.50</td>
<td>0.55</td>
<td>0.55</td>
<td>0.55</td>
<td>0.55</td>
<td>0.55</td>
<td>0.55</td>
<td>0.55</td>
</tr>
<tr>
<td>Solder (Ball) Land Pitch (e)</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Line Width Between Via and Land (w)</td>
<td>0.130</td>
<td>0.130</td>
<td>0.130</td>
<td>0.130</td>
<td>0.130</td>
<td>0.130</td>
<td>0.130</td>
<td>0.130</td>
<td>0.130</td>
<td>0.130</td>
</tr>
<tr>
<td>Distance Between Via and Land (D)</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
<td>0.70</td>
</tr>
<tr>
<td>Via Land (VL) Diameter</td>
<td>0.61</td>
<td>0.61</td>
<td>0.61</td>
<td>0.61</td>
<td>0.61</td>
<td>0.61</td>
<td>0.61</td>
<td>0.61</td>
<td>0.61</td>
<td>0.61</td>
</tr>
<tr>
<td>Through Hole (VH), Diameter</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
<td>0.300</td>
</tr>
<tr>
<td>Pad Array</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
<td>Full</td>
</tr>
<tr>
<td>Matrix or External Row</td>
<td>16 x 16</td>
<td>22 x 22</td>
<td>26 x 26</td>
<td>26 x 26</td>
<td>30 x 30</td>
<td>34 x 34</td>
<td>34 x 34</td>
<td>39 x 39</td>
<td>42 x 42</td>
<td>42 x 42</td>
</tr>
<tr>
<td>Periphery Rows</td>
<td>-</td>
<td>7(3)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Notes:
1. Dimensions in millimeters.
2. 3 x 3 matrix for illustration only, one land pad shown with via connection.
3. FG456 package has solder balls in the center in addition to the periphery rows of balls.
4. Component land pad diameter refers to the pad opening on the component side (solder-mask defined).
For Xilinx BGA packages, NSMD (Non Solder Mask Defined) pads on the board are suggested. This allows a clearance between the land metal (diameter L) and the solder mask opening (diameter M) as shown in Figure 5-80. The space between the NSMD pad and the solder mask, and the actual signal trace widths depends on the capability of the PCB vendor. The cost of the PCB is higher when the line width and spaces are smaller.

Selection of the pad types and pad sizes determines the available space between adjacent balls for signal escape. Based on PCB capability, the number of lines that can share the available space is described in Figure 5-81. Based on geometrical considerations, if one signal escapes between adjacent balls, then two signal rows can be routed on a single metal layer. This is illustrated in Figure 5-81, as routing with one line/channel, either at 6 mil lines and spaces or 5 mil lines and spaces. The blocked nature of multi-gigabit transceiver (MGT) at the top edge prevents a direct implementation of one line/channel arrangement in layer 1. Using this suggested routing scheme, a minimum of eight PCB layers are required to route up to 10 signal rows in a package.

These Virtex-II Pro BGA packages may incorporate up to 16 MGT channels per package. The balls that make up these channels are grouped within the two outer rows of top and bottom edges. This outer row arrangement allows easy escape and pairing of signals. Accommodation of any filtering schemes off the chip can be accomplished in close proximity as well.

A slightly lower trace width than that employed on the top and bottom external or exposed traces can be used by the inner signal rows routed in internal layers. Depending on the signal being handled, the practice of “necking down” a trace in the critical space between the BGA balls is allowable. Changes in width over very short distances can cause small impedance changes. Validate these issues with the board vendor and signal integrity engineers responsible for design. It is also suggested to implement the MGT signals in stripline arrangement.

Figure 5-81 describes a board-level layout strategy for a Xilinx 1.0 mm pitch FF672 package, which incorporates four MGT channels at the top edge. Detail A in Figure 5-81 describes the opening geometry for the land pad and the solder mask. Routing with 5 mil lines/trace allows one signal per channel (between the balls). For successful routing, eight-row-deep signal traces require six to eight PCB layers. Figure 5-82 shows the suggested schematic of layers for the eight-layer routing scheme. By using a premium board technology such as Microvia Technology—allowing up to 4 mil lines and spaces—efficient routing with a reduced number of board layers is made possible. A grouping scheme for power, ground, control, and I/O pins may also enable efficient routing.

Figure 5-83 and Figure 5-84 show examples of suggested layer-by-layer board escape routing strategy used to implement MGT and LVDS pairs for two of the Virtex-II Pro packages, the wire-bond FG456 and the flip-chip FF672. These drawings assume a standard PCB technology of 5 mil wide lines and spaces.

More details are contained in XAPP157, which is available on the web at www.xilinx.com/bvdocs/appnotes/xapp157.pdf, as is a full-color (PDF) version of this document.
Chapter 5: PCB Design Considerations

Figure 5-81: **FF672 PC Board Layout/Land Pattern**

![Diagram showing FF672 PC Board Layout/Land Pattern]

- **Legend:**
  - I/O
  - GND
  - PWR
  - VCCINT
  - VINT

- **Dimensions:**
  - 27 x 27 mm, 1.0 mm Fine Pitch BGA

Figure 5-82: **Eight-Layer Routing Scheme**

<table>
<thead>
<tr>
<th>Signal</th>
<th>Layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>L-1</td>
</tr>
<tr>
<td>S2</td>
<td>L-2</td>
</tr>
<tr>
<td>S3 – MGT</td>
<td>L-3</td>
</tr>
<tr>
<td>S4</td>
<td>L-4</td>
</tr>
<tr>
<td>S5</td>
<td>L-5</td>
</tr>
<tr>
<td>Power/Gnd</td>
<td>L-6</td>
</tr>
<tr>
<td>Power/Gnd</td>
<td>L-7</td>
</tr>
<tr>
<td>S5</td>
<td>L-8</td>
</tr>
</tbody>
</table>

![Diagram showing Eight-Layer Routing Scheme]
Figure 5-83: FG456 Routing with LVDS Pairs
Figure 5-84: FF672 Routing with LVDS Pairs
XPower

XPower is the first graphic power-analysis software available for programmable logic design. Earlier than ever in the design flow you can analyze total device power, power per net, routed, or partially routed or unrouted designs. You can also receive graphical or ASCII-based reports, all driven from a comprehensive graphic interface, or use a command-line driven batch-mode execution.

XPower supports Virtex-II Pro and Virtex-II advanced FPGA devices and the XPLA3 family of CPLD devices, offering the broadest device support available in programmable power estimation.

XPower also supports the importing of simulation data from ModelSim™ VCD files, greatly reducing the time spent setting up net activity, and increasing overall estimation accuracy.

XPower uses device knowledge and design data to project device power and by-net power utilization. This is a significant advance from the static estimation pages most logic providers offer their customers, and it is a leap forward in providing logic designers with accurate power dissipation information.

Features include:

- Support for Virtex-II Pro, Virtex-II, Virtex-E, Virtex, and Spartan-II FPGAs in ISE 4.1i
- XPLA3 device support through ISE WebPACK 4.1i
- VCD simulation file import for ModelSim™
- Windows 98 and NT, Windows ME and Windows 2000, and Solaris support
- Graphic and/or ASCII report formats
- Menu-driven or batch-mode execution
- Device data read directly from Xilinx layout files
- Save and recall setup data
- Recognize small voltage variations on VCC

More details are available under the Xpower link on the www.xilinx.com website.

IBIS Models

The need for higher system performance leads to faster output transitions. Signals with fast transitions cannot be considered purely digital. It is therefore important to understand their analog behavior by signal integrity analysis.

To simulate the signal integrity of printed circuit boards (PCB) accurately and solve design problems before the PCB is fabricated, models of the I/O characteristics are required. SPICE models were traditionally used for this purpose, however, a manufacturer's SPICE models contain proprietary circuit-level information. Therefore, there was a need for a model type which does not give details of the circuit topology, or process parameters. One such standard is the I/O Buffer Information Specification (IBIS) format originally suggested by Intel.

In the early 1990's, the IBIS Open Forum was formed and the first IBIS specification was written to promote tool independent I/O models for system signal integrity analysis. IBIS is now the ANSI/EIA-656 and IEC 62014-1 standard. IBIS accurately describes the signal behavior of the interconnections without disclosing the actual technology and circuitry used to implement the I/O. The standard is basically a black-box approach to protect proprietary information.
Using IBIS Models

IBIS models can be used by designers for system-level analysis of signal integrity issues, such as ringing, ground bounce, cross talk, and predicting RFI/EMI. Complete designs can be simulated and evaluated before going through the expensive and time consuming process of producing prototype PCBs. This type of pre-layout simulation can considerably reduce the development cost and time to market, while increasing the reliability of the I/O operation.

IBIS models consist of look-up tables that predict the I/V characteristics and dV/dt of integrated circuit inputs and outputs when combined with PCB traces, cables and passive components. The data is extracted for the typical case, minimum case (weak transistors, low VCC, hot temperatures) and maximum case (strong transistors, high VCC, cold temperatures).

IBIS models have a limitation in that they do not contain internal delays. IBIS models contain package parasitic information for simulation of ground bounce. However, not all simulators are able to use this data to simulate ground bounce. Simulation results may not agree with measurement results due to package, die, PCB trace and ground plane modeling inaccuracies. Similarly, because simultaneous-switching outputs (SSOs) are also difficult to model, only a first approximation is provided to the designer.

IBIS Generation

IBIS models are generated either from SPICE simulations or from actual device measurements. A SPICE netlist of the I/O buffer is required to produce V/I and dV/dt simulation curve data. The SPICE simulation data is then converted to an IBIS format/syntax file.

IBIS models that are derived from measurement data do not have process corner information, unlike IBIS models that are derived from SPICE simulation data. It is only practical to measure a few parts, and it is therefore impossible to represent the extremes of production by such a method.

Advantages of IBIS

Using IBIS models has a great advantage to the user in that simulation speed is significantly increased over SPICE, while accuracy is only slightly decreased. Non-convergence, which can be a problem with SPICE models and simulators, is eliminated in IBIS simulation. Virtually all EDA vendors presently support IBIS models and ease of use of these IBIS simulators is generally very good. IBIS models for most devices are freely available over the Internet, making it easy to simulate several different manufacturers devices on the same board.

IBIS File Structure

An IBIS file contains two sections, the header and the model data for each component. One IBIS file can describe several devices. The following is the contents list in a typical IBIS file:

- IBIS Version
- File Name
- File Revision
- Component
- Package R/L/C
- Pin name, model, R/L/C
IBIS Models

- Model (i.e., 3-state)
- Temperature Range (typical, minimum, and maximum)
- Voltage Range (typical, minimum, and maximum)
- Pull-Up Reference
- Pull-Down Reference
- Power Clamp Reference
- Ground Clamp Reference
- I/V Tables for:
  - Pull-Up
  - Pull-Down
  - Power Clamp
  - Ground Clamp
- Rise and Fall dV/dt for minimum, typical, and maximum conditions (driving 50Ω)
- Package Model (optional) <package_number>.pkg with RLC sections.

IBIS I/V and dV/dt Curves

A digital buffer can be measured in receive (3-state) mode and drive mode. IBIS I/V curves are based on the data of both these modes. The transition between modes is achieved by phasing in/out the difference between the driver and the receiver models, while keeping the receiver model constantly in the circuit.

The I/V curve range required by the IBIS specification is \(-V_{CC}\) to \((2xV_{CC})\). This wide voltage range exists because the theoretical maximum overshoot due to a full reflection is twice the signal swing. The ground clamp I/V curve must be specified over the range \(-V_{CC}\) to \(V_{CC}\), and the power clamp I/V curve must be specified from \(V_{CC}\) to \((2xV_{CC})\).

The three supported conditions for the IBIS buffer models are typical values (required), minimum values (optional), and maximum values (optional). For CMOS buffers, the minimum condition is defined as high temperature and low supply voltage, and the maximum condition is defined as low temperature and high supply voltage.

An IBIS model of a digital buffer has four I/V curves:

- The pull-down I/V curve contains the mode data for the driver driving low. The origin of the curve is at 0V for CMOS buffers.
- The pull-up I/V curve contains the mode data for the driver driving high. The origin of the curve is at the supply voltage \(V_{CC}\) or \(V_{DD}\).
- The ground clamp I/V curve contains receive (3-state) mode data, with the origin of the curve at 0V for CMOS buffers.
- The power clamp I/V curve contains receive (3-state) mode data, with the origin of the curve at the supply voltage \(V_{CC}\) or \(V_{DD}\). For 3.3V buffers that are 5V tolerant, the power clamp is referenced to 5V while the pull-up is referenced to 3.3V.

Ramp and dV/dt Curves

The Ramp keyword contains information on how fast the pull-up and pull-down transistors turn on/off. The dV/dt curves give the same information, while including the effects of die capacitance \(C_{\text{comp}}\). \(C_{\text{comp}}\) is the total die capacitance as seen at the die pad, excluding the package capacitance.
dV/dt curves describe the transient characteristics of a buffer more accurately than ramps. A minimum of four dV/dt curves are required to describe a CMOS buffer: pull-down ON, pull-up OFF, pull-down OFF, and pull-up ON. dV/dt curves incorporate the clock-to-out delay, and the length of the dV/dt curve corresponds to the clock speed at which the buffer is used. Each dV/dt curve has t = 0, where the pulse crosses the input threshold.

Xilinx IBIS Package Parasitic Modelling

Xilinx IBIS modeling previously used a simple RCL model for the pin and bond wire parasitics. Due to the fast rise and fall times of many of the supported I/O standards, it was deemed necessary to improve the package parasitic modeling. The latest IBIS 3.2 specification has a complex parasitic package model, which incorporates a transmission line and lumped RCL model. Unfortunately, IBIS 3.2 is still not widely supported by simulators.

For these reasons, the old lumped package parasitic parameters have been removed from the latest models, and the user should now manually add an external transmission line. A 65Ω ideal transmission line, with the delay set at 25 ps to 100 ps, is recommended. This works in conjunction with a revised lumped model (included inside the IBIS model). For critical applications, both extremes (25 ps and 100 ps) should be checked. However, for most I/O applications this difference is very small.

IBIS Simulations

The circuit shown in Figure 5-85 models a Virtex-II Pro LVCMOS_18F driver and receiver, connected by a 6-inch 50Ω circuit board trace. The rise and fall simulation results show that there is a large amount of overshoot and undershoot, which is actually limited by the IOB clamp diodes.

By adding a series termination resistor to the driver output, the overshoot and undershoot can be effectively controlled. The simulation results in Figure 5-86 show that by adding a 25Ω termination there is minimal overshoot and undershoot.
An alternative is to use a DCI source impedance controlled driver. The results shown in Figure 5-87 show the waveforms from an LVDCI_18 driver, which has virtually no overshoot or undershoot.

**Figure 5-86: Series Termination Example**

IBIS Simulators

Several different IBIS simulators are available today, and each simulator provides different results. An overshoot or undershoot of ±10% of the measured result is tolerable. Differences between the model and measurements occur because not all parameters are modeled. Simulators for IBIS models are provided by Cadence, Avanti Corporation, Hyperlynx, Mentor, Microsim, Intusoft, Veribest, and Viewlogic.

**Figure 5-87: DCI Driver Example**
Chapter 5: PCB Design Considerations

Xilinx IBIS Advantages

Xilinx provides preliminary IBIS files before working silicon has been verified (before tape out), as well as updated versions of IBIS files after the ICs are verified. Preliminary IBIS files are generated from SPICE models before working silicon has been verified. After the IC (device) is verified, appropriate changes are made to the existing IBIS files. These IBIS files are available at the web site http://www.xilinx.com/support/sw_ibus.htm

ANSI/EIA IBIS Official Website

http://www.eigroup.org/ibis/ibus.htm

BSDL and Boundary Scan Models

Boundary scan is a technique that is used to improve the testability of ICs. With Virtex-II Pro devices, registers are placed on I/Os that are connected together as a long shift register. Each register can be used to either save or force the state of the I/O. There are additional registers for accessing test modes.

The most common application for boundary scan is testing for continuity of the IC to the board. Some packages make visual inspection of solder joints impossible, e.g. BGA. The large number of I/Os available requires the use of such packages, and also increases the importance of testing. A large number of I/Os also means a long scan chain.

Test software is available to support testing with boundary scan. The software requires a description of the boundary scan implementation of the IC. The IEEE 1149.1 specification provides a language description for Boundary Scan Description Language (BSDL). Boundary scan test software accepts BSDL descriptions.

The IEEE 1149.1 spec also defines a 4 to 5 pin interface known as the JTAG interface. IEEE 1532 is a capability extension of IEEE 1149.1.

BSDL Files

Preliminary BSDL files are provided from the IC Design Process. Final BSDL files have been verified by an external third party test and verification vendor. Table 5-6 lists the Virtex-II Pro BSDL file names.

Table 5-6: Virtex-II Pro BSDL File Names

<table>
<thead>
<tr>
<th>XC2VP2_FG256.BSD</th>
<th>XC2VP20_FG696.BSD</th>
<th>XC2VP50_FF1152.BSD</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC2VP2_FG456.BSD</td>
<td>XC2VP20_FF896.BSD</td>
<td>XC2VP50_FF1148.BSD</td>
</tr>
<tr>
<td>XC2VP2_FF672.BSD</td>
<td>XC2VP20_FF1152.BSD</td>
<td>XC2VP50_FF1517.BSD</td>
</tr>
<tr>
<td>XC2VP4_FG256.BSD</td>
<td>XC2VP30_FG696.BSD</td>
<td>XC2VP70_FF1517.BSD</td>
</tr>
<tr>
<td>XC2VP4_FG456.BSD</td>
<td>XC2VP30_FF896.BSD</td>
<td>XC2VP70_FF1704.BSD</td>
</tr>
<tr>
<td>XC2VP4_FF672.BSD</td>
<td>XC2VP30_FF1152.BSD</td>
<td>XC2VP100_FF1704.BSD</td>
</tr>
<tr>
<td>XC2VP7_FG456.BSD</td>
<td>XC2VP40_FG696.BSD</td>
<td>XC2VP100_FF1696.BSD</td>
</tr>
<tr>
<td>XC2VP7_FF672.BSD</td>
<td>XC2VP40_FF1152.BSD</td>
<td></td>
</tr>
<tr>
<td>XC2VP7_FF896.BSD</td>
<td>XC2VP40_FF1148.BSD</td>
<td></td>
</tr>
</tbody>
</table>
Appendix A

BitGen and PROMGen Switches and Options

Using BitGen

BitGen produces a bitstream for Xilinx device configuration. After the design has been completely routed, it is necessary to configure the device so that it can execute the desired function. The Xilinx bitstream necessary to configure the device is generated with BitGen. BitGen takes a fully routed NCD (Circuit Description) file as its input and produces a configuration bitstream—a binary file with a .bit extension.

The BIT file contains all of the configuration information from the NCD file defining the internal logic and interconnections of the FPGA, plus device-specific information from other files associated with the target device. The binary data in the BIT file can then be downloaded into the FPGA memory cells, or it can be used to create a PROM file (see Figure A-1).
Appendix A: BitGen and PROMGen Switches and Options

BitGen Syntax

The following syntax creates a bitstream from your NCD file.

```
bitgen [options] <infile>.ncd> [outfile] [pcf_file]
```

*options* is one or more of the options listed in “BitGen Options,” page 501.

*<infile>* is the name of the NCD design for which you want to create the bitstream. You can specify only one design file, and it must be the first file specified on the command line.

You do not have to use an extension. If you do not, .ncd is assumed. If you do use an extension, it must be .ncd.

*<outfile>* is the name of the output file. If you do not specify an output file name, BitGen creates one in the same directory as the input file. If you specify -1 on the command line, the extension is .ll (see -l command line option). If you specify -m (see -m command line option), the extension is .msk. If you specify -b, the extension is .rbt. Otherwise, the extension is .bit. If you do not specify an extension, BitGen appends one according to the aforementioned rules. If you do include an extension, it must also conform to the rules.

*<pcf_file>* is the name of a physical constraints (.pcf) file. BitGen uses this file to determine which nets in the design are critical for tiedown, which is not available for Virtex families. BitGen automatically reads the .pcf file by default. If the physical constraints file is the second file specified on the command line, it must have a .pcf extension. If it is the third file specified, the extension is optional; .pcf is assumed. If a .pcf file name is specified, it must exist, otherwise the input design name with a .pcf extension is read if that file exists.

A report file containing all BitGen’s output is automatically created under the same directory as the output file. The report file has the same root name as the output file with a .bgn extension.

BitGen Files

This section describes input files that BitGen requires and output files that BitGen generates.

Input Files

Input to BitGen consists of the following files.

- **NCD file**—a physical description of the design mapped, placed and routed in the target device. The NCD file must be fully routed.
- **PCF file**—an optional user-modifiable ASCII Physical Constraints File. If you specify a PCF file on the BitGen command line, BitGen uses this file to determine which nets in the design are critical for tiedown (not used for Virtex families).

Output Files

Output from BitGen consists of the following files.

- **BIT file**—a binary file with a .bit extension. The BIT file contains all of the configuration information from the NCD file defining the internal logic and interconnections of the FPGA, plus device-specific information from other files associated with the target device. The binary data in the BIT file can then be downloaded into the FPGA memory cells, or it can be used to create a PROM file (see “Using PROMGen,” page 507).
Using BitGen

- **RBT file**—an optional “rawbits” file with a .rbt extension. The rawbits file is ASCII ones and zeros representing the data in the bitstream file. If you enter a -b option on the BitGen command line, an RBT file is produced in addition to the binary BIT file (see “-b (Create Rawbits File),” page 501).

- **LL file**—an optional ASCII logic allocation file with a .ll extension. The logic allocation file indicates the bitstream position of latches, flip-flops, and IOB inputs and outputs. A .ll file is produced if you enter a -l option on the BitGen command line (“-l (Create a Logic Allocation File),” page 507).

- **MSK file**—an optional mask file with a .msk extension. This file is used to compare relevant bit locations for executing a readback of configuration data contained in an operating FPGA. A MSK file is produced if you enter a -m option on the BitGen command line (see “-m (Generate a Mask File),” page 507).

- **BGN file**—a report file containing information about the BitGen run.

- **DRC file**—a Design Rule Check (DRC) file for the design. A DRC runs and the DRC file is produced unless you enter a -d option on the BitGen command line (see “-d (Do Not Run DRC),” page 501).

**BitGen Options**

Following is a description of command line options and how they affect BitGen behavior.

- **-b (Create Rawbits File)**

  Create a “rawbits” <file_name>.rbt file. The rawbits file consists of ASCII ones and zeros representing the data in the bitstream file.

  If you are using a microprocessor to configure a single FPGA, you can include the rawbits file in the source code as a text file to represent the configuration data. The sequence of characters in the rawbits file is the same as the sequence of bits written into the FPGA.

- **-d (Do Not Run DRC)**

  Do not run DRC (Design Rule Check). Without the -d option, BitGen runs a DRC and saves the DRC results in two output files: the BitGen report file <file_name>.bgn and the DRC file (<file_name>.drc). If you enter the -d option, no DRC information appears in the report file and no DRC file is produced.

  Running DRC before a bitstream is produced detects any errors that could cause the FPGA to malfunction. If DRC does not detect any errors, BitGen produces a bitstream file (unless you use the -j option described in the “-j (No BIT File),” page 507).

- **-f (Execute Commands File)**

  -f <command_file>

  The -f option executes the command line arguments in the specified <command_file>.

- **-g (Set Configuration)**

  -g <option>:<setting>

  The -g option specifies the startup timing and other bitstream options for Xilinx FPGAs. The settings for the -g option depend on the design’s architecture. These options have the following syntax:
Appendix A: BitGen and PROMGen Switches and Options

Binary
Creates a binary file with programming data only. Use this option to extract and view programming data. Any changes to the header will not affect the extraction process.

Settings: No, Yes
Default: No

CclkPin
Adds an internal pull-up to the Cclk pin. The Pullnone setting disables the pullup.

Settings: Pullnone, Pullup
Default: Pullup

Compress
This option uses the multiple frame write feature in the bitstream to reduce the size of the bitstream, not just the .bit file. Using the Compress option does not guarantee that the size of the bitstream will shrink.

ConfigRate
Virtex-II Pro uses an internal oscillator to generate the configuration clock, CCLK, when configuring in a master mode. Use the configuration rate option to select the rate for this clock.

Settings: 4, 5, 7, 8, 9, 10, 13, 15, 20, 26, 30, 34, 41, 45, 51, 55, 60
Default: 4

DCMShutdown
When DCMShutdown is enabled, the DCM (Digital Clock Manager) resets if the SHUTDOWN and AGHIGH commands are loaded into the configuration logic.

Settings: Disable, Enable
Default: Disable

DebugBitstream
If the device does not configure correctly, you can debug the bitstream using the DebugBitstream option. A debug bitstream is significantly larger than a standard bitstream. The values allowed for the DebugBitstream option are No and Yes.

Note: You should use this option only if your device is configured to use slave or master serial mode.

Values: No, Yes

In addition to a standard bitstream, a debug bitstream offers the following features:

- Writes 32 0s to the LOUT register after the synchronization word
- Loads each frame individually
- Performs a cyclical redundancy check (CRC) after each frame
- Writes the frame address to the LOUT register after each frame

DisableBandgap
Disables bandgap generator for DCMs to save power.

Settings: No, Yes
Default: No
DONE_cycle
Selects the Startup phase that activates the FPGA Done signal. Done is delayed when DonePipe=Yes.

Settings: 1, 2, 3, 4, 5, 6
Default: 4

DonePin
Adds an internal pull-up to the DONE Pin pin. The Pullnone setting disables the pullup.
Use this option only if you are planning to connect an external pull-up resistor to this pin. The internal pull-up resistor is automatically connected if you do not use this option.

Settings: Pullup, Pullnone
Default: Pullup

DonePipe
This option is intended for use with FPGAs being set up in a high-speed daisy chain configuration. When set to Yes, the FPGA waits on the CFG_DONE (DONE) pin to go High and then waits for the first clock edge before moving to the Done state.

Settings: No, Yes
Default: No

DriveDone
This option actively drives the DONE Pin high as opposed to using a pullup.

Settings: No, Yes
Default: No

Encrypt
Encrypts the bitstream. (For more information, see “Bitstream Encryption” in Chapter 3.)

Settings: No, Yes
Default: No

GTS_cycle
Selects the Startup phase that releases the internal 3-state control to the I/O buffers. The Done setting releases GTS when the DoneIn signal is High. DoneIn is either the value of the Done pin or a delayed version if DonePipe=Yes.

Settings: Done, 1, 2, 3, 4, 5, 6, Keep
Default: 5

GWE_cycle
Selects the Startup phase that asserts the internal write enable to flip-flops, LUT RAMs, and shift registers. It also enables the BRAMs. Before the Startup phase both BRAM writing and reading are disabled. The Done setting asserts GWE when the DoneIn signal is High. DoneIn is either the value of the Done pin or a delayed version if DonePipe=Yes. The Keep setting is used to keep the current value of the GWE signal.

Settings: Done, 1, 2, 3, 4, 5, 6, Keep
Default: 6
Appendix A: BitGen and PROMGen Switches and Options

Key0, Key1, Key2, Key3, Key4, Key5

Sets Key\(x\) for bitstream encryption. The Pick option causes BitGen to select a random number for the value. (For more information, see “Bitstream Encryption” in Chapter 3.)

- **Settings:** Pick, \(<\text{hex\_string}>\)
- **Default:** Pick

KeyFile

Specifies the name of the input encryption file.

- **Settings:** \(<\text{string}>\)
- **Default:** Pick

Keyseq0, Keyseq1, Keyseq2, Keyseq3, Keyseq4, Keyseq5

Sets the key sequence for key\(x\). The settings are equal to the following:

- **S** = single
- **F** = first
- **M** = middle
- **L** = last

- **Settings:** S, F, M, L
- **Default:** S

LCK_cycle

Selects the Startup phase to wait until DLLs/DCMs lock. If NoWait is selected, the Startup sequence does not wait for DLLs/DCMs.

- **Settings:** 0, 1, 2, 3, 4, 5, 6, NoWait
- **Default:** NoWait

M0Pin

The M0 pin is used to determine the configuration mode. Adds an internal pull-up, pull-down or neither to the M0 pin. Select Pullnone to disable both the pull-up resistor and pull-down resistor on the M0 pin.

- **Settings:** Pullup, Pulldown, Pullnone
- **Default:** Pullup

M1Pin

The M1 pin is used to determine the configuration mode. Adds an internal pull-up, pull-down or neither to the M1 pin. Select Pullnone to disable both the pull-up resistor and pull-down resistor on the M1 pin.

- **Settings:** Pullup, Pulldown, Pullnone
- **Default:** Pullup

M2Pin

The M2 pin is used to determine the configuration mode. Adds an internal pull-up, pull-down or neither to the M2 pin. Select Pullnone to disable both the pull-up resistor and pull-down resistor on the M2 pin.

- **Settings:** Pullup, Pulldown, Pullnone
- **Default:** Pullup
Match_cycle
Specifies a stall in this Startup cycle until DCI (Digitally Controlled Impedance) match signals are asserted.

Settings: NoWait, 0, 1, 2, 3, 4, 5, 6
Default: NoWait

Persist
This option is needed for Readback and Partial Reconfiguration using the SelectMAP configuration pins. If Persist is set to Yes, the pins used for SelectMAP mode are prohibited for use as user IO. Refer to the data sheet for a description of SelectMAP mode and the associated pins.

Settings: No, Yes
Default: No

ProgPin
Adds an internal pull-up to the ProgPin pin. The Pullnone setting disables the pull-up. The pull-up affects the pin after configuration.

Settings: Pullup, Pullnone
Default: Pullnone

ReadBack
This option allows you to perform Readback by the creating the necessary bitstream. When specifying the -g Readback option, the .rba, .rbb, .rbd, and .msd file are created.

Security
Selecting Level1 disables Readback. Selecting Level2 disables Readback and Partial Reconfiguration.

Settings: None, Level1, Level2
Default: None

StartCBC
Sets the starting CBC (Cipher Block Chaining) value. The pick option causes BitGen to select a random number for the value.

Settings: Pick, <hex_string>
Default: Pick

StartKey
Sets the starting key number.

Settings: 0, 3
Default: 0

StartupClk
The startup sequence following the configuration of a device can be synchronized to either Cclk, a User Clock, or the JTAG Clock. The default is Cclk.

- Cclk
  Enter Cclk to synchronize to an internal clock provided in the FPGA device.
Appendix A: BitGen and PROMGen Switches and Options

- **JTAG Clock**
  Enter JtagClk to synchronize to the clock provided by JTAG. This clock sequences the TAP controller which provides the control logic for JTAG.

- **UserClk**
  Enter UserClk to synchronize to a user-defined signal connected to the CLK pin of the STARTUP symbol.
  
  Settings: Cclk (pin—see Note), UserClk (user-supplied), JtagCLK
  Default: Cclk

  **Note:** In modes where Cclk is an output, the pin is driven by an internal oscillator.

**TckPin**
Adds a pull-up, a pull-down or neither to the TCK pin, the JTAG test clock. Selecting one setting enables it and disables the others. The Pullnone setting indicates there is no connection to either the pull-up or the pull-down.

  Settings: Pullup, Pulldown, Pullnone
  Default: Pullup

**TdiPin**
Adds a pull-up, a pull-down, or neither to the TDI pin, the serial data input to all JTAG instructions and JTAG registers. Selecting one setting enables it and disables the others. The Pullnone setting indicates there is no connection to either the pull-up or the pull-down.

  Settings: Pullup, Pulldown, Pullnone
  Default: Pullup

**TdoPin**
Adds a pull-up, a pull-down, or neither to the TdoPin pin, the serial data output for all JTAG instruction and data registers. Selecting one setting enables it and disables the others. The Pullnone setting indicates there is no connection to either the pull-up or the pull-down.

  Settings: Pullup, Pulldown, Pullnone
  Default: Pullup

**TmsPin**
This option selects an internal pullup or pulldown on the TMS (JTAG Mode Select) pin.

  Settings: Pullnone, Pullup, Pulldown
  Default: Pullup

**UnusedPin**
Adds a pull-up, a pull-down, or neither to the UnusedPin, the serial data output for all JTAG instruction and data registers. Selecting one setting enables it and disables the others. The Pullnone setting indicates there is no connection to either the pull-up or the pull-down.

The following settings are available. The default is PullDown.

  Settings: Pullup, Pulldown, Pullnone
  Default: Pulldown
UserID
You can enter up to an 8-digit hexadecimal code in the User ID register. You can use the
register to identify implementation revisions.

-h or -help (Command Usage)

-h <architecture>
Displays a usage message for BitGen. The usage message displays all available options for
BitGen operating on the specified <architecture>.

-j (No BIT File)
Do not create a bitstream file (.bit file). This option is generally used when you want to
generate a report without producing a bitstream. For example, if you wanted to run DRC
without producing a bitstream file, you would use the -j option.

Note: The .msk or .rbt files might still be created.

-l (Create a Logic Allocation File)
This option creates an ASCII logic allocation file (<design>.ll) for the selected design.
The logic allocation file indicates the bitstream position of latches, flip-flops, and IOB
inputs and outputs.

In some applications, you may want to observe the contents of the FPGA internal registers
at different times. The file created by the -l option helps you identify which bits in the
current bitstream represent outputs of flip-flops and latches. Bits are referenced by frame
and bit number within the frame.

The Hardware Debugger uses the <design>.ll file to locate signal values inside a
readback bitstream.

-m (Generate a Mask File)
Creates a mask file. This file is used to compare relevant bit locations for executing a
readback of configuration data contained in an operating FPGA.

-w (Overwrite Existing Output File)
Enables you to overwrite an existing .bit, .ll, .msk, or .rbt output file.

Using PROMGen

The PROMGen program is compatible with the following families.

• Virtex/Virtex-E/Virtex-II/Virtex-II Pro

PROMGen formats a BitGen-generated configuration bitstream (BIT) file into a PROM
format file (Figure A-2).

The PROM file contains configuration data for the FPGA device. PROMGen converts a BIT
file into one of three PROM formats: MCS-86 (Intel), EXORMAX (Motorola), or TEKHEX
(Tektronix). It can also generate a Hex file format.
Appendix A: BitGen and PROMGen Switches and Options

Figure A-2: PROMGen

There are two functionally equivalent versions of PROMGen. There is a stand-alone version you can access from an operating system prompt. You can also access an interactive version, called the PROM File Formatter, from inside the Design Manager for Alliance or the Project Manager in Foundation. This chapter describes the stand-alone version; the interactive version is described in the PROM File Formatter Guide.

You can also use PROMGen to concatenate bitstream files to daisy-chain FPGAs.

Note: If the destination PROM is one of the Xilinx Serial PROMs, you are using a Xilinx PROM Programmer, and the FPGAs are not being daisy-chained, it is not necessary to make a PROM file. See the Hardware User Guide for more information about daisy-chained designs.

PROMGen Syntax

Use the following syntax to start PROMGen from the operating system prompt:

```
promgen [options]
```

<wbr />

[options] can be any number of the options listed in “PROMGen Options,” page 509. Separate multiple options with spaces.

PROMGen Files

This section describes the PROMGen input and output files.

Input Files

The input to PROMGEN consists of BIT files— one or more bitstream files. BIT files contain configuration data for an FPGA design.

Output Files

Output from PROMGEN consists of the following files.
• **PROM files**—The file or files containing the PROM configuration information. Depending on the PROM file format used by the PROM programmer, you can output a TEK, MCS, or EXO file. If you are using a microprocessor to configure your devices, you can output a HEX file, containing a hexadecimal representation of the bitstream.

• **PRM file**—The PRM file is a PROM image file. It contains a memory map of the output PROM file. The file has a `.prm` extension.

**Bit Swapping in PROM Files**

PROMGen produces a PROM file in which the bits within a byte are swapped compared to the bits in the input BIT file. Bit swapping (also called “bit mirroring”) reverses the bits within each byte, as shown in Figure A-3.

In a bitstream contained in a BIT file, the Least Significant Bit (LSB) is always on the left side of a byte. But when a PROM programmer or a microprocessor reads a data byte, it identifies the LSB on the right side of the byte. In order for the PROM programmer or microprocessor to read the bitstream correctly, the bits in each byte must first be swapped so they are read in the correct order.

In this release of the Xilinx Development System, the bits are swapped for all of the PROM formats: MCS, EXO, and TEK. For a HEX file output, bit swapping is on by default, but it can be turned off by entering a `-b` PROMGen option that is available only for HEX file format.

**PROMGen Options**

This section describes the options that are available for the PROMGen command.

- `-b` (Disable Bit Swapping—HEX Format Only)
  
  This option only applies if the `-p` option specifies a HEX file for the output of PROMGen. By default (no `-b` option), bits in the HEX file are swapped compared to bits in the input BIT files. If you enter a `-b` option, the bits are not swapped. Bit swapping is described in “Bit Swapping in PROM Files,” page 509.

- `-c` (Checksum)
  
  ```
  promgen -c
  ```
The `-c` option generates a checksum value appearing in the `.prm` file. This value should match the checksum in the prom programmer. Use this option to verify that correct data was programmed into the prom.

**-d (Load Downward)**

```
  promgen -d <hexaddress0> <filename> <filename> ...
```

This option loads one or more BIT files from the starting address in a downward direction. Specifying several files after this option causes the files to be concatenated in a daisy-chain. You can specify multiple `-d` options to load files at different addresses. You must specify this option immediately before the input bitstream file.

The multiple file syntax is as follows:

```
  promgen -d <hexaddress0> <filename> <filename> ...
```

The multiple `-d` options syntax is as follows:

```
  promgen -d <hexaddress1> <filename> -d <hexaddress2> <filename> ...
```

**-f (Execute Commands File)**

```
  -f <command_file>
```

The `-f` option executes the command line arguments in the specified `<command_file>`.

**-help (Command Help)**

This option displays help that describes the PROMGen options.

**-l option (Disable Length Count)**

```
  promgen -l
```

The `-l` option disables the length counter in the FPGA bitstream. It is valid only for 4000EX, 4000XL, 4000XLA, 4000XV, and SpartanXL Devices. Use this option when chaining together bitstreams exceeding the 24-bit limit imposed by the length counter.

**-n (Add BIT Files)**

```
  -n <file1[.bit]> <file2[.bit]> ...
```

This option loads one or more BIT files up or down from the next available address following the previous load. The first `-n` option must follow a `-u` or `-d` option because `-n` does not establish a direction. Files specified with this option are not daisy-chained to previous files. Files are loaded in the direction established by the nearest prior `-u`, `-d`, or `-n` option.

The following syntax shows how to specify multiple files. When you specify multiple files, PROMGen daisy-chains the files.

```
  promgen -d <hexaddress> <file0> -n <file1> <file2> ...
```

The following syntax when using multiple `-n` options prevents the files from being daisy-chained:

```
  promgen -d <hexaddress> <file0> -n <file1> -n <file2> ...
```

**-o (Output File Name)**

```
  -o <file1>[.<ext>] <file2>[.<ext>] ...
```
This option specifies the output file name of a PROM if it is different from the default. If you do not specify an output file name, the PROM file has the same name as the first BIT file loaded.

_.ext_ is the extension for the applicable PROM format.

Multiple file names may be specified to split the information into multiple files. If only one name is supplied for split PROM files (by you or by default), the output PROM files are named <file>_.<#>_.ext>, where <file> is the base name, <#> is 0, 1, etc., and <ext> is the extension for the applicable PROM format.

```
promgen -d <hexaddress> <file0> -o <filename>
```

-p (PROM Format)

```
-p (mcs | exo | tek | hex)
```

This option sets the PROM format to one of the following: MCS (Intel MCS86), EXO (Motorola EXORMAX), TEK (Tektronix TEKHEX). The option may also produce a HEX file, which is a hexadecimal representation of the configuration bitstream used for microprocessor downloads. If specified, the -p option must precede any -u, -d, or -n options. The default format is MCS.

-r (Load PROM File)

```
r <promfile>
```

This option reads an existing PROM file as input instead of a BIT file. All of the PROMGen output options may be used, so the -r option can be used for splitting an existing PROM file into multiple PROM files or for converting an existing PROM file to another format.

-s (PROM Size)

```
s <promsize1> <promsize2> ...
```

This option sets the PROM size in kilobytes. The PROM size must be a power of 2. The default value is 64 kilobytes. The -s option must precede any -u, -d, or -n options.

Multiple <promsize> entries for the -s option indicates the PROM will be split into multiple PROM files.

**Note:** PROMGen PROM sizes are specified in bytes. _The Programmable Logic Data Book_ specifies PROM sizes in bits for Xilinx serial PROMs (see -x option).

-u (Load Upward)

```
u <hexaddress0> <filenamel> <filename2> ...
```

This option loads one or more BIT files from the starting address in an upward direction. When you specify several files after this option, PROMGen concatenates the files in a daisy-chain. You can load files at different addresses by specifying multiple -u options.

This option must be specified immediately before the input bitstream file.

-x (Specify Xilinx PROM)

```
x <xilinx_prom1> <xilinx_prom2> ...
```

The -x option specifies one or more Xilinx serial PROMs for which the PROM files are targeted. Use this option instead of the -s option if you know the Xilinx PROMs to use.
Appendix A: BitGen and PROMGen Switches and Options

Multiple `<xilinx_prom>` entries for the `-x` option indicates the PROM will be split into multiple PROM files.

Examples

To load the file `test.bit` up from address `0x0000` in MCS format, enter the following information at the command line:

```
promgen -u 0 test
```

To daisy-chain the files `test1.bit` and `test2.bit` up from address `0x0000`, and the files `test3.bit` and `test4.bit` from address `0x4000`, while using a 32K PROM and the Motorola EXORmax format, enter the following information at the command line:

```
promgen -s 32 -p exo -u 00 test1 test2 -u 4000 test3 test4
```

To load the file `test.bit` into the PROM programmer in a downward direction starting at address `0x400`, using a Xilinx XC1718D PROM, enter the following information at the command line:

```
promgen -x xc1718d -d 0x400 test
```

To specify a PROM file name that is different from the default file name, enter the following information at the command line:

```
promgen [options] <filename> -o <newfilename>
```
Appendix B

Platform Flash Family PROMs

This appendix contains package specifications for the Platform Flash Family of In-System Programmable Configuration PROMs. The latest versions of these drawings are available online at www.xilinx.com.

The data sheet for the Platform Flash Family of In-System Programmable Configuration PROMs can be accessed at www.xilinx.com/bvdocs/publications/ds123.pdf.

PROM Package Specifications

This section contains specifications for the following Platform Flash Family PROM packages:

- FS48 Package Specification
- VO48 Package Specification
- VO20 Package Specification
FS48 Package Specification

Figure B-1: FS48 Package Specification

NOTES:
1. ALL DIMENSIONS AND TOLERANCES CONFORM TO ANSI Y14.5M-1994
2. SYMBOL "M" IS THE PIN MATRIX SIZE.
VO48 Package Specification

**NOTES:**


2. DATUM [A–B] AND [D] TO BE DETERMINED WHERE CENTERLINE BETWEEN LEADS EXIST PLASTIC BODY AT DATUM PLANE [H].

3. "D1" & "E" DO NOT INCLUDE MOLD FLASH OR PROTRUSIONS, AND ARE MEASURED AT THE PARTING LINE, MOLD FLASH OR PROTRUSIONS SHALL NOT EXCEED 0.15mm ON "E" AND 0.25 ON "D1" PER SIDE.

4. THE LEAD WIDTH DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE 0.08mm TOTAL IN EXCESS OF THE LEAD WIDTH DIMENSION AT MAXIMUM MATERIAL CONDITION. DAMBAR CANNOT BE LOCATED ON THE LOWER RADIUS OR THE FOOT. MINIMUM SPACE BETWEEN PROTRUSIONS AND AN ADJACENT LEAD TO BE 0.07mm. SEE SECTION "C–C".

5. LEAD FINISH: VO48 – 85/15 (Sn/Pb)
   VO48C – 100% Matte Sn

6. CONTROLLING DIMENSION: MILLIMETERS.

7. THIS PART IS COMPLIANT WITH JEDEC SPECIFICATION MO–142 VARIATION DD.

**Figure B-2:** VO48 Package Specification
Appendix B: Platform Flash Family PROMs

VO20 Package Specification

**TOP VIEW**

PIN-1 ID.

**BOTTOM VIEW**

EJECTOR PIN. (OPTIONAL)

**SIDE VIEW**

SEATING PLANE

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN.</th>
<th>NOM.</th>
<th>MAX.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>--</td>
<td>--</td>
<td>.047</td>
</tr>
<tr>
<td>A1</td>
<td>.002</td>
<td>.004</td>
<td>.006</td>
</tr>
<tr>
<td>A2</td>
<td>.031</td>
<td>--</td>
<td>.041</td>
</tr>
<tr>
<td>B</td>
<td>.007</td>
<td>--</td>
<td>.012</td>
</tr>
<tr>
<td>C</td>
<td>.004</td>
<td>--</td>
<td>.0079</td>
</tr>
<tr>
<td>D</td>
<td>.252</td>
<td>.256</td>
<td>.260</td>
</tr>
<tr>
<td>E</td>
<td>.169</td>
<td>.173</td>
<td>.176</td>
</tr>
<tr>
<td>E1</td>
<td>.0256</td>
<td></td>
<td></td>
</tr>
<tr>
<td>H</td>
<td>.244</td>
<td>.252</td>
<td>.260</td>
</tr>
<tr>
<td>L</td>
<td>.0197</td>
<td>.0236</td>
<td>.0295</td>
</tr>
<tr>
<td>α</td>
<td>0°</td>
<td>5°</td>
<td>8°</td>
</tr>
</tbody>
</table>

**NOTES:**

2. DIMENSION "D" DOES NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION SHALL NOT EXCEED .006" PER SIDE.
3. DIMENSION "E" DOES NOT INCLUDE MOLD PROTRUSION. ALLOWABLE MOLD PROTRUSION SHALL NOT EXCEED .010" PER SIDE.
4. LEAD FINISH: VO20 - 85/15 (Sn/Pb) VO20G - 100% Matte Sn
5. CONFORMS TO JEDEC MO-153-AC

**Figure B-3: VO20 Package Specification**
Appendix C

Choosing the Battery for $V_{BATT}$

As noted in section “$V_{BATT}$” in Chapter 3, the battery backed-up RAM (BBRAM) array in the FPGA requires either $V_{CCAUX}$ or $V_{BATT}$ to keep the key programmed. Please reference DC and Switching Characteristics datasheet for $V_{BATT}$ and $I_{BATT}$ specifications. Once the Virtex-II Pro $V_{CCAUX}$ power is applied, the BBRAM is no longer powered from the $V_{BATT}$ pin. However, the leakage specification ($I_{BATT}$) still applies, as leakage results primarily from secondary structures rather than from the BBRAM itself.

Depending on the application, the BBRAM may need to be powered from time of shipment until used by a customer. This time period may be long, and involve storage at temperature extremes that shorten battery life. In the case of a primary (non-rechargeable) battery, it is safe to assume is that the battery must supply current to the system for the entire product life.

Battery Types and Chemistry Choices

In the engineering of a battery back-up system, designers need to decide if the battery system is to be primary (non-rechargeable) or secondary (rechargeable), and what cell voltage and chemistry are appropriate.

In the primary battery category, there are alkaline, lithium, and silver-oxide batteries. Their cell voltages are 1.5V, 3.0V, and 1.55V DC respectively at the beginning of service. Alkaline batteries are a good choice for operation below 54°C (130°F), and above –4°C (–20°F). Alkaline cells have a lifetime of less than one or two years without a load. Lithium batteries that are designed for high and low temperature operation may be used up to 125°C and down to –40°C. Lithium cells have a 15-year shelf or storage life, and if sized correctly may exhibit the same life in a system. Silver oxide cells are designed to operate from –10°C to +60°C, and have a less than 5% per year self-discharge rate at 21°C.

In the secondary battery category, there are nickel-cadmium, nickel metal hydride, lithium, and lead-acid batteries. Their cell voltages are 1.2V, 1.2V, 3.6V, and 2.0V DC respectively after charge under nominal load. The self-discharge rates of all of these batteries are such that they will typically need to be recharged within 30 days. The maximum number of deep (> 80%) charge/recharge cycles is usually less than 200 for these types of batteries.

Primary or Secondary

Primary batteries simplify the design as they cannot be recharged, and therefore require no extra circuitry. Secondary batteries require a method of charging, and can be damaged if charging currents are not controlled. Primary batteries can last up to 15 years (shelf life) which means that a battery could be placed in circuit, and forgotten about for the life of the product. Secondary batteries can provide significant current for a month or more, but then require recharging.
If the design goal is to configure the keys at the factory, and then ship the product, lithium primary cells are the first choice, followed by silver oxide as a second choice. These two examples will be detailed below.

If the design goal is to install keys in the field, through a system level secure key management system, then rechargeable batteries should be considered as they may outlast the product life as long as the product remains operational. A rechargeable example is also given.

Battery Cost and Care

Lithium batteries cost from USD1.00 to USD2.00 for 50 mAh to 1000 mAh capacity. For the rechargeable option, the battery cost is similar, but the extra cost of the charge controller IC must be added (USD2.50 to USD5.00).

Nickel cadmium batteries must be disposed of properly due to the cadmium metal. Lead acid batteries also have to be disposed of properly and silver is also considered a hazardous waste. Most batteries contain highly corrosive base or acid electrolytes, and will seriously damage or destroy electrical components if they leak. Batteries will leak if they are overcharged, if a primary battery type is charged, or if the battery suffers physical damage (dented or punctured).

Not all battery types are available throughout the world. Standard nickel cadmium batteries are generally available, as are alkaline cells. If operation anywhere in the world is needed, the standard AA alkaline cell may be obtained just about everywhere.

All batteries may have a risk of explosion or fire if they are shorted. The use of a small 1/8-watt surface mount resistor as a fuse is a necessary precaution.

Battery Summary

Table C-1: Primary Cell Types and Characteristics

<table>
<thead>
<tr>
<th></th>
<th>Lithium</th>
<th>Silver Oxide</th>
<th>Alkaline</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal Voltage:</td>
<td>3.0V</td>
<td>1.55V</td>
<td>1.5V</td>
</tr>
<tr>
<td>Temperature Range:</td>
<td>−40°C to +150°C</td>
<td>−10°C to +60°C</td>
<td>−20°C to +54°C</td>
</tr>
<tr>
<td>Lifetime:</td>
<td>&gt; 15 years</td>
<td>&gt; 10 years</td>
<td>&gt; 1 year</td>
</tr>
</tbody>
</table>
| Comments:        | • Best choice for long life  
                   • Industrial grade  
                   • Safely disposable | • Used in medical systems  
                   • Hazardous waste | • Inexpensive  
                   • Safely disposable (except Minnesota) |

Table C-2: Secondary Cell Types and Characteristics

<table>
<thead>
<tr>
<th></th>
<th>Lead Acid</th>
<th>Nickel Cadmium</th>
<th>Nickel Metal Hydride</th>
<th>Lithium Ion (Polymer)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nominal Voltage:</td>
<td>2.0V</td>
<td>1.2V</td>
<td>1.2V</td>
<td>3.6V</td>
</tr>
<tr>
<td>Temperature Range:</td>
<td>0°C to +40°C</td>
<td>0°C to +45°C</td>
<td>0°C to +40°C</td>
<td>0°C to +45°C</td>
</tr>
<tr>
<td>Lifetime:</td>
<td>&gt; 5 years</td>
<td>&gt; 5 years</td>
<td>&gt; 5 years</td>
<td>&gt; 2 years</td>
</tr>
</tbody>
</table>
| Comments:        | • Heavy  
                   • High current  
                   • Hazardous waste | • Low self-discharge rate  
                   • Hazardous waste | • High self-discharge rate  
                   • Safely disposable | • Complex charger circuit  
                   • Lightweight  
                   • Safely disposable |
Design Examples

Case #1:

A 15-year operational life is desired, using a primary extended-temperature-range lithium button or coin cell. The current required is for ten Virtex-II or Virtex-II Pro FPGAs, or 10 x 100 nA = 1 uA, continuous. Leakage currents from the printed circuit board (PCB) are about 1 uA. Thus, total current = 2 µA.

The smallest coin cell from Panasonic is the BR1225A 48 mAh (milliampere hour), size 0.49” diameter by 0.1” height. The capacity in mAh divided by the current is the number of hours to end-of-life voltage. In this case, the number of hours available is:

\[
\frac{48\text{mAh}}{0.002\text{mA}} = 24,000\text{ hours (~2.7 years)}
\]

To obtain 15 years, we need a larger-capacity battery.

Notice in this case that the PCB leakage is limiting battery life. If the PCB is specified with lower leakage (which typically requires de-ionized water wash cycles), the smallest coin cell is adequate for about 5 years of life.

If a 15-year life is required, a larger cell may be used. The cost of the larger-capacity battery might be offset by eliminating the de-ionized PCB water wash.

End-of-life battery voltage is 2.0V DC. Start-of-life battery voltage is 3.0V DC, nominal. There are no restrictions on disposal of this type of battery, so there is no need for hazard labels.

Adding a series diode is a good way to isolate the battery from the device. It allows for current testing by placing a current meter across the diode.

Case #2:

A Rayovac 675G is primary silver oxide cell, size 0.5” diameter by 0.2” thick. It provides 190 mAh starting at 1.55V and ending at 1.0V. In the previous case, where 1 uA operating current plus 1 uA PCB leakage yielded a total current of 2 µA, a single cell would reach only about 11 years of life rather than the 15 years required. Limiting the PCB leakage to less current is one option to achieve longer battery life. Another option is to use two cells in parallel to achieve the 15-year life requirement.

Because silver is considered a hazardous waste, there are restrictions on the disposal of a silver oxide battery. Customer instructions and labels regarding battery disposal will add to the cost of the system.

Case #3:

A system is designed to utilize a lithium polymer rechargeable battery. The batteries require a sophisticated charge controller IC to prevent damage and optimize performance. A National LM3621 charger IC is chosen to keep the cell charged, and requires only 1 µA from the cell to function. The total requirement is now 3 µA. There is a 5% charge loss per month, with no load utilizing a Battery Engineering 152-102-6.3 4000 mAh Polymer Rechargeable Lithium Battery, so the shelf life after first charge is ~20 months. The life with 3 µA load is 21 years, much longer than the self discharge of 5% per month. If no other use of the current is required, a smaller battery may be used to balance out the self-discharge life. If a goal of one year is desired, the smallest cell from the catalog (86-54-3.5, 700 mAh) may be used.
Appendix C: Choosing the Battery for $V_{BATT}$

Start-of-cycle voltage may be as high as 4.6V, so it must be dropped to 3.6V for the Virtex-II and Virtex-II Pro BBRAM power pin. Use of 1N914 or 1N4148 diodes in series is suggested to drop the voltage. End-of-cycle voltage will not go below 3.0V. The voltage drop from the diodes reduces this to a lower number, but this is still well above the 1.0V required by the $V_{BATT}$ pin.

In general, a 10 nA current is enough to drop the voltage 0.4 volts using two 1N914/1N4148 diodes. If the total current is less than the minimum forward current specified for the diode, this guideline can be used to estimate how many diodes are required in series to achieve a certain voltage drop. Double-check with the diode vendor and with lab measurements to confirm.

There is no disposal restriction on the lithium polymer battery, but the charging restrictions require a special integrated circuit, and there are many rules to be followed in the proper use of a rechargeable lithium cell.

Case #4:

The cost and care required for the lithium battery in Case #3 were cause for rejection in the design review process. The alternative is nickel metal hydride, which has no disposal concerns, and can be charged with a simple series resistor. A Panasonic HHR55AAA/FT (requiring a AAA cell holder) will provide 560 mAh capacity, more than enough for our application. For charging, the self-discharge rate of the battery needs only to be offset. Nickel Metal Hydride batteries lose ~1% capacity per day, so charging at $C/100$ (where $C =$ capacity in mAh) would just offset the self-discharge. Since it is safe to charge a nickel metal hydride battery at $C/20$ forever, a good compromise choice would be $C/50$, or 11 mA.

If the local power supply is 3.3V and the terminal battery voltage when fully charged is 1.45V, a series diode is required to prevent the battery from discharging when the power is turned off. Using a 1N914 diode, the drop is 0.7V, and $3.3 - 1.45 - 0.7 = 1.15V$. For 11 mA, the required resistor is therefore about $105 \Omega$. A 110 $\Omega$ 1/8-watt 5% resistor is all that is needed.

In order to replace the battery (if needed), a capacitor is added in parallel to the battery to hold the voltage for the time it takes to replace a battery.

For 2 $\mu$A, we solve for capacitance in $I = C \times dV/dt$, or $C = I \times dt/dV$. For a dV of 0.4 volts, a dT of 100 seconds, and a current of 1 $\mu$A, the capacitance required is 250 $\mu$F. The next safe standard value would be 470 $\mu$F. A high-quality, low-leakage aluminum electrolytic or tantalum would be recommended.
Glossary

Click on a letter, or scroll down to view the entire glossary.

A B C D E F G H I J K L M N O P Q R S T U V W X Z

A

AC Coupling

Method of interfacing drivers and receivers through a series capacitor. Often used when the differential swing between drivers and receivers is compatible, but common mode voltages of driver and receiver are not. Requires that a minimum data frequency be established based on the RC time constant, necessitating a run length limit.

Addressing Modes

Techniques used by software or hardware in calculating an address.

ALU (Arithmetic and Logic Unit)

The part of a processor that performs integer addition, subtraction, multiplication, division and other Boolean logic instructions.

ASIC (Application-Specific Integrated Circuit)

An integrated circuit designed to perform a particular function by defining the interconnection of a set of basic circuit building blocks drawn from a library provided by the circuit manufacturer.

Asynchronous

Logic that is not synchronized by a clock. Asynchronous designs can be faster than synchronous ones, but are more sensitive to parametric changes, and are thus less robust.

ATM (Asynchronous Transfer Mode)

A very-high-speed (megahertz to gigahertz) connection-oriented bit-serial protocol for transmitting data and real-time voice and video in fixed-length packets (48-byte payload, 5-byte header).
Attenuation

Reduction in amplitude of a signal.

B

Back Annotation

Automatically attaching timing values to the entered design format after the design has been placed and routed in a field-programmable gate array (FPGA).

BER (Bit Error Rate)

A measurement of the number of errors detected at a receiver in a given length of time, sometimes specified as a percentage of received bits; sometimes specified in exponential form (10E-8 to indicate 1 bit error in 10E-8 bits).

BERT (Bit Error Rate Test, Bit Error Rate Tester)

A test or an instrument used to determine the Bit Error Rate (BER) of a device or system under test. It is generally made up of a test pattern generator, receiver, and analyzer.

Big Endian

A representation of a multi-byte value that has the most significant byte of any multi-byte data field stored at the lowest memory address. Also see “Little Endian.”

BIST (Built-In Self Test)

The technique of designing circuits with additional logic that can be used to test proper operation of the primary (functional) logic.

Bitstream

The bitstream is a binary representation of an implemented FPGA design. The bitstream is generated by Xilinx bit generation tools (BitGen and Makebits) and is denoted with the .bit extension. For information on creating BIT files, refer to the Hardware Debugger Reference/User Guide.

Block RAM

An 18 Kb block of random access memory (RAM) inside the Virtex-II Pro device. Dual-port and synchronous operation are desirable.

Block SelectRAM

Fully-synchronous, dual-port memories in the Virtex-II Pro FPGAs. Each of these memories contain 18 x 1024 (18,432) bits. The organization of each memory is configurable. Block SelectRAM
resources complement smaller, distributed, LUT-based SelectRAM resources.

**Book “E”**

A Motorola and IBM jointly written architectural definition and instruction set for embedded PowerPC implementations.

**Boot ROMs**

The program used to bring up a computer or system that is stored in Read Only Memory

**Boundary Scan Interface**

One of the configuration interfaces on the Virtex device. This is a bit-serial interface. The Boundary Scan interface is also known as the JTAG port. Also see “SelectMAP Interface.”

**Breakpoint**

A location in a program at which execution is to be stopped and control of the processor switched to the debugger. Mechanisms for creating and removing breakpoints are provided by most debugging tools.

**BSP (Board Support Package)**

The purpose of the BSP is to isolate the user design from hardware making it easier to write new applications and to port applications from other environments.

BSP consists of a set of software modules that offer interface to peripheral devices and low-level processor core functions for user software or C library. For example, these functions include hardware initialization, boot monitor access, drivers for all the available resources and features on board.

**C**

**Cache**

A small block of high-speed memory located between the processor and main memory. Cache stores frequently used data and instructions. Cache improves system performance by reducing the need to access the system’s slower main memory.

**Cache Block**

See “Cache Line”

**Cache Line**

A portion of a cache array that contains a copy of contiguous system-memory addresses. Cache lines in PPC405 are 32 bytes long and aligned on a 32 byte address.
Cache Set

See “Congruence Class.”

Capture Data

The flip-flop and pad data saved from the logic cells and I/O blocks into the bitstream for readback. Use the CAPTURE_VIRTEX primitive in your HDL code to specify the trigger and clock for the capture operation.

CCLK (Configuration Clock)

During configuration, the configuration clock (CCLK) is an output in Master modes or in the Asynchronous Peripheral mode but is an input in Slave, Synchronous Peripheral, Express, and SelectMAP/Slave Serial modes. After configuration, CCLK has a weak pull-up and can be selected as the readback clock.

CDR (Clock/Data Recovery)

Feature of most high-speed serial transceivers. At the receiver, a clock is generated based on the timing of data transitions. In this way, a clock signal is derived from the data.

Channel Bonding

Feature of multi-channel high-speed transceivers. Allows multiple channels to be used together, offering a greater aggregate bandwidth.

Chirp

Bit sequence which is transmitted by a high-speed transceiver when it is not in use. The chirp is usually a repeating pattern of IDLE characters. The purpose of the chirp is to keep clock recovery circuits aligned and active while the link is not transmitting data.

CISC (Complex Instruction Set Computer)

The architecture of a processor family. CISC processors generally feature variable-length instructions, multiple addressing formats, and contain only a small number of general-purpose registers. Intel’s 80x86 family is the quintessential example of CISC.

CLB (Configurable Logic Block)

Xilinx-specific name for a block of logic surrounded by routing resources. The functional elements for constructing logic circuits. The Virtex-II Pro CLB is made up of four slices, and each slice contains two Logic Cells.

CML (Current Mode Logic)

A differential I/O standard used in high-speed serial channels. Voltage swing is typically from 450 mV to 1200 mV.
Coherency

Coherency describes the ordering of reads from and writes to a single memory location. A memory system is coherent when the value read from a memory address is always the last value written to the address. In a system where all devices read and write from a single, shared system memory, memory is always coherent.

Comma

A comma is a “K” character used by the transceiver to align the serial data on a byte/half-word boundary (depending on the protocol used), so that the serial data is correctly decoded into parallel data.

Common Mode

The DC component of a signal. In differential channels, it is the average voltage of the differential pair.

Compiler

Software that converts a higher-language description into a lower-level representation. For FPGAs, the complete partition, place, and process.

Configuration Bitstream

Configuration commands with configuration data.

Configuration Commands

Instructions for the Virtex-II Pro device. There are two classes of Configuration Command — Major and Minor. The Major Commands read and write data to configuration registers in the Virtex-II Pro device. The Minor commands instruct the Virtex-II Pro configuration logic to perform specific functions.

Configuration Data

Bits that directly define the state of programmable logic. These are written to a Virtex-II Pro device in a configuration bitstream, and read as readback data from a Virtex-II Pro device.

Configuration File

The internally stored file that controls the FPGA so that it performs the desired logic function. Also, the act of loading an FPGA with that file. That is, the process of programming Xilinx SRAM-based FPGAs with a bitstream.

Configuration Frame

The configuration bits in a Virtex-II Pro device are organized in columns. A column of CLBs with the I/O blocks above and below the CLBs contain 48 frames of configuration bits. The smallest number of bits that can be read or written through the configuration interfaces is one frame.
Configuration Interface
A logical interface on the Virtex-II Pro device through which configuration commands and data can be read and written. An interface consists of one or more physical device pins.

Configuration Readback
The operation of reading configuration data (also known as readback data) from a Virtex-II Pro device.

Congruence Class
A collection of cache lines with the same index.

Constraints
Performance requirements imposed on the design, usually in the form of maximum allowable delay, or the required operating frequency.

Context Switch
The process of switching from one task to another in a multitasking operating system. A context switch involves saving the context of the running task and restoring the previously-saved context of the other.

CoreConnect™ Bus Architecture
An interconnection internal bus developed by IBM. It eases the integration and reuse of processor, system, and peripheral cores. Elements of CoreConnect architecture include the processor local bus (PLB), the on-chip peripheral bus (OPB), a bus bridge, and a device control register (DCR) bus.

Cross Compiler
A compiler that runs on a particular processor architecture and produces code for a different architecture.

Crosstalk
Undesirable signal coupling from noisy aggressor nets to victim nets. May be eliminated by increasing the spacing between the nets or reducing signal amplitude of the aggressor net.

CS Pin
The CS pin is the Chip Enable pin for Virtex-II Pro devices. It is used only in SelectMAP mode. When CS is asserted (Low) the device examines data on the Data bus. When CS is de-asserted (High), all CCLK transitions are ignored.
DataFrame

A DataFrame is a block of configuration data. A configuration bit-stream contains many such frames, each with a start bit and stop bits. Also see “Configuration Frame.”

DC Balanced

A channel is said to be DC Balanced if it has an equal number of 1’s and 0’s transmitted across it. Encoding schemes like 8B10B are designed to ensure this.

DC Coupling

Method of interfacing drivers and receivers without the use of series capacitors. A direct connection (through PCB trace) from driver to receiver.

DCR (Device Control Register)

A CoreConnect bus. DCR is designed to transfer data between the CPU’s general purpose registers (GPRs) and the DCR slave logic’s device control registers (DCRs).

Debug Monitor

A piece of embedded software that has been designed specifically for use as a debugging tool. It usually resides in ROM and communicates with a debugger via a serial port or network connection. The debug monitor provides a set of primitive commands to view and modify memory locations and registers, create and remove breakpoints, and execute your program. The debugger combines these primitives to fulfill higher-level requests like program download and single-step.

Debugger

A software development tool used to test and debug embedded software. The debugger runs on a host computer and connects to the target through a serial port or network connection. Using a debugger you can download software to the target for immediate execution. You can also set breakpoints and examine the contents of specific memory locations and registers.

Deterministic Jitter

The component of jitter attributable to the data pattern in the channel. Different digital patterns have different spectral contents. These differing spectral contents give rise to varying amounts of signal jitter.

Dhrystone MIPS

Dhrystone is a benchmark program for testing a system’s integer performance. The objective is to compare the performance of a machine against the performance of a reference machine. The industry
has adopted the VAX 11/780 as the reference 1 MIPS (Million Instruction Per Second) machine.

**Differential Signaling**

A signaling scheme which uses two complementary signals to transmit data. Differential signaling offers faster data rates at reduced signal swing with higher signal-to-noise ratio.

**DIN Pin**

During serial configuration, the DIN pin is the serial configuration data input receiving data on the rising edge of CCLK. During parallel configuration, DIN is the D0 input. After configuration, DIN is a user-programmable I/O pin.

**Dirty Bit**

A bit in a memory cache or virtual memory page that has been modified by the CPU but not yet written back to storage.

**Dispersion**

“Smearing” of a signal or waveform as a result of transmission through a non-ideal transmission line. Through a non-ideal medium, signals travel at different velocities according to their frequency. Dispersion of the signal is the result. All cables and PCB transmission lines are non-ideal.

**DONE Pin**

The DONE pin on a Xilinx FPGA is a bidirectional signal with an optional internal pull-up resistor. As an output, it indicates the completion of the configuration process. As an input, a low level on DONE can be configured to delay the global logic initialization and the enabling of outputs.

**Double Word**

Eight bytes or 64 bits.

**DOUT Pin**

During configuration in any mode except Express and SelectMAP, the DOUT pin is the serial configuration data output that can drive the DIN pin of daisy-chained slave FPGAs. DOUT data changes on the falling edge of CCLK, one-and-a-half CCLK periods after it is received at the DIN pin (in Master Serial Mode only).

**DOUT/BUSY Pin**

For Virtex-II Pro devices, the DOUT/BUSY pin has a dual purpose, depending on device mode. When the device is in Serial mode, this pin functions as DOUT. When the device is in SelectMAP/Slave Parallel mode, this pin functions as a handshaking signal. If BUSY is asserted (High) on a rising edge of CCLK, the data is not seen on the data bus, and should be held until the data is accepted.
DRAM (Dynamic Random Access Memory)
A low-cost read-write memory where data is stored on capacitors and must be refreshed periodically. DRAMs are usually addressed by a sequence of two addresses, row address, and column address, which makes them slower and more difficult to use than SRAMs. Also see “SRAM.”

DSOCM (Data-Side On Chip Memory)
See “OCM (On-Chip Memory).”

DSP (Digital Signal Processing)
The manipulation of analog data that has been sampled and converted into a digital representation. Examples are filtering, convolution, Fast-Fourier-Transform, and so on.

EDIF (Electronic Data Interchange Format)
Industry standard for specifying a logic design in text (ASCII) form.

EEMBC (Embedded Microprocessor Benchmark Consortium)
It develops and certifies real-world benchmarks and benchmark scores to help designers select embedded processors.

Effective Address
The un-translated memory address as seen by a program.

Emulator
Short for “ICE (In-Circuit Emulator).”

Endianness
See “Big Endian” and “Little Endian.”

Equalization
Amplification or attenuation of certain frequency components of a signal. Used to counteract the effects of a non-ideal transmission medium.

ESD (Electrostatic Discharge)
High-voltage discharge can rupture the input transistor gate oxide. ESD-protection diodes divert the current to the supply leads.

Exception
An abnormal event or condition that requires the processor’s attention. They can be caused by instruction execution or an external
device. The processor records the occurrence of an exception and they often cause an interrupt to occur.

**Eye Diagram**

An eye diagram of a signal overlays the signal’s waveform over many cycles. Each cycle’s waveform is aligned to a common timing reference, typically a clock. An eye diagram provides a visual indication of the voltage and timing uncertainty associated with the signal. It can be generated by synchronizing an oscilloscope to a timing reference.

The vertical thickness of the line bunches in an eye diagram indicate the magnitude of AC voltage noise, whereas the horizontal thickness of the bunches where they cross over is an indication of the AC timing noise or jitter. Fixed DC voltage and timing offsets are indicated by the position of the eye on the screen.

**Eye Mask**

The size of the eye opening in the center of an eye diagram indicates the amount of voltage and timing margin available to sample this signal. Thus, for a particular electrical interface, a fixed reticule or window could be placed over the eye diagram showing how the actual signal compares to minimum criteria window, know as the eye mask. If a margin rectangle with width equal to the required timing margin and height equal to the required voltage margin fits into the opening, then the signal has adequate margins. Voltage margin can often be traded off for timing margin.

**F**

**Fall Time**

The time it takes for a waveform to transition from the high logic state to the low logic state. Fall time is usually measured from 90% of the total signal swing to 10% of the signal swing.

**FIFO (First-In First-Out)**

FIFO memory where data is stored in the incoming sequence and is read out in the same sequence. Input and output can be asynchronous to each other. A FIFO needs no external addresses, although all modern FIFOs are implemented internally with RAMs driven by circular read and write counters.

**FIT (Failure In Time)**

Describes the number of device failures statistically expected for a certain number of device-hours. Expressed as failures per one billion (10^9) device hours. Device temperature must be specified. Mean time between failure (MTBF) can be calculated from FIT. 10 FITs are good; 100 FITs are bad.
FIT (Fixed Interval Timer)
One of several user-accessible timers available in the Virtex-II Pro FPGA’s PowerPC 405 core. The FIT provides timer interrupts having a repeatable period. The FIT is functionally similar to an auto-reload Programmable Interval Timer (PIT), except that only a smaller fixed selection of interrupt periods is available.

Flash
Non-volatile programmable technology, and alternative to electrically-erasable programmable read-only memory (EEPROM) technology. The memory content can be erased by an electrical signal. This allows in-system programmability and eliminates the need for ultraviolet light and quartz windows in the package.

Flip-Flop
Single-bit storage cell that samples its data input at the active (rising or falling) clock edge, and then presents the new state on its Q output after that clock edge, holding it there until after the next active clock edge.

Flush
A cache or TLB operation that involves writing back a modified entry to memory, followed by an invalidation of the entry.

FPGA (Field Programmable Gate Array)
An integrated circuit that contains configurable (programmable) logic blocks and configurable interconnect between these blocks. Xilinx FPGAs are SRAM-based programmable logic devices (PLDs).

FPU (Floating Point Unit)
Floating-point operations include any operations that involve fractional numbers.

Frame
See “Configuration Frame.”

Function Generator
Also called a look-up table (LUT), with N inputs and one output. Can implement any logic function of its N inputs. N can be between 3 and 6; 4-input function generators are most popular.

Gate
Smallest logic element with several inputs and one output. The AND gate output is High when all inputs are High. The OR gate output is High when at least one input is High. The NAND gate output is Low.
when all inputs are High. A 2-input NAND gate is used as the measurement unit for gate array complexity.

**Gate Array**

An ASIC where transistors are predefined, and only the interconnect pattern is customized for the individual application.

**GNU**

A recursive acronym “GNU Not Unix” (pronounced “guh-NEW”). The Free Software Foundation’s GNU project was launched in 1984 to develop a complete Unix-like operating system that is freely distributed.

**GUI (Graphical User Interface )**

The way of representing the computer output on the screen as graphics, pictures, icons, and windows. Pioneered by Xerox and the Apple Macintosh, now universally adopted, e.g., by Windows95 and others.

**H**

**Halfword**

Two bytes, or 16 bits.

**HardWire**

Xilinx name for a low-cost derivative of an FPGA, where the configuration is fixed, but functionality and footprint are identical with the original FPGA-based design.

**Harvard Architecture**

Harvard architecture has separate data bus and an instruction bus. This allows instruction and data access in parallel making faster execution than a Von-Neuman architecture possible. PPC405 core is built on Harvard architecture.

**HDL (Hardware Description Language)**

A kind of language used for the conceptual design of integrated circuits. Examples are VHDL and Verilog.

**Hierarchical Design**

Design description in multiple layers, from the highest (overview) to the lowest (circuit details). An alternative is flat design, where everything is described at the same level of detail.
Hit

An indication that requested information exists in the accessed cache array, the associated fill buffer, or on the corresponding OCM interface.

HyperTransport™

A high-performance bus solution developed by Advanced Micro Devices and several partners to break the I/O bottleneck in 32- and 64-bit systems. HyperTransport provides a scalable architecture that provides better than an order of magnitude increase in bus transaction throughput over existing I/O bus architectures such as PCI, PCI-X and AGP. Formerly called Lightning Data Transport (LDT).

ICE (In-Circuit Emulator)

A debugging tool that takes the place of (emulates) the processor on the target board. Emulators frequently incorporate a special "bond-out" version of the target processor that allows the user to observe and record its internal state as the program is executing.

Idle Pattern

A data sequence transmitted by a high-speed transciever as a placeholder or for link maintenance. The particular sequence of an IDLE pattern is determined by the communication protocol, and is usually a control character like K28.5.

Impedance (Characteristic Impedance)

Electrical characteristic of a transmission line, derived from the capacitance and inductance per unit length.

Index Register

A special purpose register used by a processor when performing indexed addressing. The value in the index register is usually the reference location to which a displacement will be added.

INIT Pin

The INIT pin is a quadruple function signal. Before and during configuration, INIT is a bidirectional signal. A 1 - 10 kΩ external pull-up resistor is recommended. As an active-Low open-drain output, INIT is held Low during power stabilization and internal clearing of the configuration memory. As an active-Low input, it can be used to hold the FPGA in the internal WAIT state before the start of configuration. During configuration, a Low on this output indicates that a configuration data error has occurred. After the I/O become active in the Startup sequence, INIT becomes a user-programmable I/O.
Instruction Set

Is referred to the set of instructions that the microprocessor can execute. The instruction set specifies the types of instructions (such as load/store, integer arithmetic, and branch instructions), the specific instructions, and the encoding used for the instructions. The instruction set definition also specifies the addressing modes used for accessing memory.

Interrupt Latency

The amount of time between the assertion of an interrupt and the start of the associated interrupt service routine.

Interrupt Service Routine

A section of code written to handle the tasks associated with an interrupt request.

Interrupt Vector

A special code that identifies the circuit requesting an interrupt.

IP (Intellectual Property)

In the legal sense, patents, copyrights, and trade secrets. In integrated circuits (ICs), predefined large functions, called “cores,” that help the user complete a large design faster.

ISA (Instruction Set Architecture)

A term referring to a family of microprocessors with similar basic design, for example the PowerPC architecture includes Motorola’s PowerQUICC, PPC7440, 7410, IBM’s PPC 405, 440, and Xilinx Virtex-II Pro.

ISI (Inter-Symbol Interference)

A form of data corruption or noise due to the effect that data has on data-dependent channel characteristics.

ISOCM (Instruction-Side On Chip Memory)

See “OCM (On-Chip Memory).”

Jitter

The jitter of a periodic signal is the delay between the expected transition of the signal and the actual transition. Jitter is a zero mean random variable. When worst case analysis is undertaken the maximum value of this random variable is used.
Jitter Tolerance

Jitter tolerance is defined as the peak-to-peak amplitude of sinusoidal jitter applied on the input that causes a predefined, acceptable loss at the output. For example jitter applied to the input of an OC-N equipment interface that causes an equivalent 1dB optical power penalty.

Jitter Transfer

Jitter transfer is defined as the ratio of jitter on the output of a device to the jitter applied on the input of the device, versus frequency. Jitter transfer is important in applications where the system is utilized in a loop-timed mode, where the recovered clock is used as the source of the transmit clock.

JTAG (Joint Test Action Group)

Earlier name for IEEE 1149.1 boundary scan, a method for testing boards and integrated circuits. Also see “Parallel Cable IV.”

K

Kernel

An essential part of any multitasking operating system software which controls how the rest of the system can operate. The kernel is to software what the CPU is to hardware.

L

LAN (Local Area Network)

A computer network that spans a relatively small area. Most LANs are confined to a single building or group of buildings. However, one LAN can be connected to other LANs over any distance via telephone lines and radio waves. A system of LANs connected in this way is called a WAN (Wide Area Network).

Latency

The time between when something happens and when its response is generated. This is often critical in real-time applications.

LC (Logic Cell)

Metric for FPGA density. The basic building block of the Virtex-II Pro CLB. An LC includes a 4-input function generator, carry logic, and a storage element.

LDT (Lightning Data Transport)

See “HyperTransport™.”
**Linker**

A software development tool that accepts one or more object files as input and outputs a relocatable program. The linker is thus run after all of the source files have been compiled or assembled.

**Little Endian**

A representation of a multi-byte value that has the least significant byte of any multi-byte data field stored at the lowest memory address. Also see “Big Endian.”

**LogiBLOX**

Library of logic modules, often with user-definable parameters, like data width. Similar to LPM.

**Logical Address**

Synonym for effective address.

**Loopback**

Path in a high-speed transceiver which connects the output to the input, on either the PMA or PCS side, for testing purposes.

**LPM (Library of Parametrized Modules)**

Library of logic modules, often with user-definable parameters, like data width. Similar to LogiBLOX.

**LUT (Look-Up Table)**

Also called a function generator with N inputs and one output. Can implement any logic function of its N inputs. N is between 3 and 6; most popular are 4-input LUTs.

**LUT SelectRAM**

Shallow RAM structure implemented in CLB look-up tables (LUTs). Also see “Block SelectRAM.”

**LVDS (Low Voltage Differential Signaling)**

A differential I/O standard commonly used for high-speed, low-swing signals.

**M**

**Machine Language**

A computer language that is directly executable by a computer without the need for translation by a compiler or an assembler. Although the computer works on binary patterns, the program can usually be entered in octal or hexadecimal.
MAN (Metropolitan Area Network)
A data network designed for a town or city. In terms of geographic breadth, a MAN is larger than a LAN (Local Area Network), but smaller than a WAN (Wide Area Network).

Mapping
Process of assigning portions of the logic design to the physical chip resources (CLBs). With FPGAs, mapping is more demanding and more important a process than with gate arrays. Also see “Synthesis.”

Masking
A process in which an operation can be performed on a single bit.

Memory Map
Documentation that lists or shows the function of each location in memory.

Memory-Mapped I/O
A system of I/O in which each I/O location is treated as if it were memory.

MicroBlaze
A 32-bit soft processor developed by Xilinx

Miss
An indication that requested information does not exist in the accessed cache array, the associated fill buffer, or on the corresponding OCM interface.

MMU (Memory Management Unit)
Performs address translation (logical to physical) and protection functions. The MMU divides logical storage into pages.

Mnemonic
A easy to remember string representing a processor instruction. For example, EIEIO is the mnemonic of the PPC405 instruction ‘Enforce In Order Execution of I/O’.

MTBF (Mean Time Between Failures)
The statistically relevant up-time between equipment failures. Also see “FIT (Failure In Time).”

Multilevel Signaling
System where multiple logic levels are utilized instead of just two (High and Low). This enables the transmission of multiple bits in a single waveform.
See http://www.signalintegrity.com/Pubs/misc/MLS.htm.
MultiLINX Cable

The MultiLINX cable provides many complex functions and can be loaded with new firmware as it becomes available. It can be connected to the host computer in two ways: via a Serial port or a USB port. The MultiLINX cable is supported by the Hardware Debugger software for Slave Serial and SelectMAP/Slave Parallel programming (as appropriate), as well as readback/verify. It is also supported by the iMPACT software for JTAG programming of both CPLDs and FPGAs.

MultiPRO Desktop Tool

The MultiPRO Desktop Tool is a multi-function programming and configuration cable. Designed for use in a desktop environment, the MultiPRO Desktop Tool interfaces to a PC using the industry standard IEEE 1284 parallel port. When mated with MultiPRO device adapters, it can program individual Platform Flash PROMs, XC18V00 ISP PROMs and CoolRunner-II CPLDs.

Multiprocessing

The use of more than one processor in a single computer system. So-called "multiprocessor systems" usually have a common memory space through which the processors can communicate and share data. In addition, some multiprocessor systems support parallel processing.

Netlist

Textual description of logic and interconnects. Also see “XNF File” and “EDIF (Electronic Data Interchange Format).”

Non-maskable Interrupt (NMI)

An interrupt that cannot be turned off.

NRE (Non-Recurring Engineering) Charges

Start-up cost for the creation of an ASIC, gate array, or HardWire. Pays for layout, masks, and test development. FPGAs and CPLD do not require NRE.

Object Code

The form of software after it has been translated (compiled) from the source code format a programmer writes into the machine format a microprocessor can understand. A set of processor-readable opcodes and data. The output of compilers, assemblers, linkers, and locators are files containing object code.
OCM (On-Chip Memory)

Interface that supports the attachment of additional memory to the instruction and data caches, and that can be accessed at performance levels matching the cache arrays.

On-Chip Debugger

It can be considered as advanced on-chip debug monitor. It usually allows code download, memory/resource access, single stepping, reset, status, etc.

OPB (On-chip Peripheral Bus)

A CoreConnect bus. OPB is architected to alleviate system performance bottlenecks by reducing capacitive loading on the Processor Local Bus (PLB). OPB is designed to support lower-performance/speed peripherals such as IIC, UART, GPIO, USB, External Bus Controller, etc.

Optimization

Design change to improve performance. Also see “Synthesis.”

Overshoot

Phenomenon where a signal rises to a level greater than its steady-state voltage before settling to its steady-state voltage.

Pad

Pad bits are extra bits used to make the total number of bits in a frame an integral multiple of 32, the number of bits in a configuration word. A pad word is an extra word used at the end of a configuration frame for pipelining. A pad frame is an extra configuration frame used at the beginning of a configuration readback and at the end of a configuration write for pipelining.

Parallel Cable IV

Xilinx Parallel Cable IV (PC IV) is a high-speed download cable that configures or programs all Xilinx FPGA, CPLD, ISP PROM, and System ACE MPM devices. The cable takes advantage of the IEEE 1284 ECP protocol and Xilinx iMPACT software to increase download speeds over eight times faster than existing solutions.

Partitioning

In FPGAs, the process of dividing the logic into subfunctions that can later be placed into individual CLBs. Partitioning precedes placement.
Glossary

PCMCIA


PCS (Physical Coding Sublayer)

Part of the physical layer of the ISO/OSI reference stack model for Gigabit Ethernet. The PCS encodes 8-bit data octets into 10-bit code groups, which it passes down to the “PMA (Physical Media Attachment).” In reverse direction, it also decodes 10-bit code groups passed up from the PMA.

Peak-to-Peak

In the case of peak-to-peak voltage, a measure of a signal’s total amplitude. In the case of peak-to-peak jitter, a measure of the extremes of excursion of the bit transition times.

PECL (Positive Emitter-Coupled Logic)

A differential I/O standard based on the ECL standard, but which operates with a positive supply voltage. (ECL uses a negative supply voltage.) PECL is used in clocking and high-speed data applications.

Peripheral Component Interface (PCI)

Synchronous bus standard characterized by short range, light loading, low cost, and high performance. 66 MHz PCI can support data byte transfers up to 528 megabytes per second (MB/s) on 64 parallel data lines.

Physical Address

The actual address that is placed on the address bus when accessing a physically implemented memory location or register. This address can be translated from the effective address. When address translation is not used, this address is equal to the effective address.

Pin-Locking

Rigidly defining and maintaining the functionality and timing requirements of device pins while the internal logic is still being designed or modified. Pin-locking has become important, since circuit board fabrication times are longer than PLD design implementation times.

PIP (Programmable Interconnect Point)

In Xilinx FPGAs, a point where two signal lines can be connected, as determined by the device configuration.

Placement

In FPGAs, the process of assigning specific parts of the design to specific locations (CLBs) on the chip. Usually done automatically. Also see “Partitioning.”
PLB (Processor Local Bus)

A CoreConnect bus. PLB interconnects high-bandwidth devices such as processor cores, external memory interfaces, PCI, and DMA controllers. PLB offers 64- and 128-bit implementations.

PLD (Programmable Logic Device)

Generic name for all programmable logic: PALs, CPLDs, and FPGAs.

PLL (Phase-Locked Loop)

An electronic circuit that controls an oscillator so that it maintains a constant phase angle relative to a reference signal.

PMA (Physical Media Attachment)

Part of the physical layer of the ISO/OSI reference stack model for Gigabit Ethernet. Serializes code-grouped data passed to it from the PCS (Physical Coding Sublayer), and deserializes data to be passed up to the PCS.

PowerPC

A RISC-based computer architecture developed jointly by IBM, Apple Computer, and Motorola corporation. The name PowerPC is derived from IBM’s name for the POWER (Performance Optimization With Enhanced RISC) architecture.

PRBS (Pseudo-Random Bit Sequence)

A pattern that appears to be random, but is actually a predictable and repeatable sequence with a very long interval (i.e., billions of bits before repeating), depending on the pattern.

Pre-emphasis

Pre-emphasis is magnitude boosting of high frequency spectral components before launching the signal (wave) onto the Transmission Line. Transmission Lines embedded in most standard PCB materials (FR4, Rogers 43xx, Nelco and Rogers) suffer varying degrees of dispersion and loss in the 1 gigahertz spectrum. This is mostly due to conductance losses (leakage from the copper trace to any other conducting structure) and Skin Effect. Dispersion is a phenomenon whereby spectral components travel at different velocities. The waveform looks smeared when it arrives at the receiver.

Both of these "characteristics" play into a diminished and poorly received signal. By boosting the high freq. spectral components, the magnitude of these components can be diminished as the wave travels through the Transmission Line, but since it starts out larger than the lower frequency components, the composite signal arrives at the receiver looking the way it was intended.

Pre-emphasis is done by simply increasing the maximum amplitude of the signal for one bit period. If the signal is 1 bit in duration, the amplitude is allowed to rise to a value which is some percentage...
greater in magnitude. At this point, if the signal is to stay at the same logic state, the driver sends a decreased magnitude signal, or nominal logic level. Every time a transition occurs, the greater magnitude level is used. For all times after this that the same level is to be transmitted, the nominal magnitude is used.

**Priority**

The level of importance of an event. Most often, interrupts are assigned priorities.

**Privileged (or Supervisor) Mode**

Privileged mode allows programs to access all registers and execute all instructions supported by the processor. Normally, the operating system and low-level device drivers operate in this mode.

**Program Counter**

A register that places addresses on the bus to retrieve information stored within a program.

**PROGRAM Pin**

The PROGRAM pin is an active-Low input that forces clearing of the FPGA configuration memory and is used to initiate a configuration cycle. While PROGRAM is held Low, the FPGA drives INIT Low and continues to clear the configuration memory. When PROGRAM goes High, the FPGA finishes the current clear cycle, executes another complete clear cycle, goes into a WAIT state, and releases INIT.

**Random Jitter**

Jitter caused by Power Supply noise, temperature variations and crosstalk.

**Readback**

Initiating a readback causes the configuration memory to become accessible to be serially clocked out and read from the device, or (byte-wide in SelectMAP/Slave Parallel modes). The configuration memory contains the configuration data, facilitating a Read-Verification of the data. The configuration memory can also contain the CLB output logic states facilitating a Read-Capture of the internal logic states. Read-Verification and Read-Capture are used by the Hardware Debugger for hardware verification. For information on the readback specification and timing, refer to *The Programmable Logic Data Book*. For information on using the readback component in a design, refer to the *Libraries Guide*. For information on enabling the readback function in the Implementation Software, refer to the *Development System Reference Guide*. For information on using the Hardware Debugger refer to the *Hardware Debugger Reference/User Guide*. For information
on connecting the XChecker cable for readback, refer to the Hardware Users Guide.

**Readback Data**

Configuration data read from a Virtex-II Pro device. The data is organized as configuration frames.

**Real Address**

Synonym for physical address.

**Real Mode**

In real mode, programs address physical memory directly.

**Register**

A memory location that is part of a processor or an I/O device. In other words, it’s not normal memory. Generally, each bit or set of bits within the register controls some behavior of the larger device.

**Relative Addressing**

An addressing mode that calculates a new address based on the position of an instruction within a program.

**Ringing**

Common name for the characteristic waveform seen when a transmission line ends at a high impedance discontinuity. The signal first overshoots the target voltage, then sags below, then overshoots again—and continues this oscillating pattern with decreasing swing amplitude until finally settling at the target voltage.

**RISC (Reduced Instruction Set Computer)**

A type of microprocessor architecture that runs very fast by simplifying the number of its commands. PPC405 is a RISC microprocessor.

**Rise Time**

The time it takes for a signal to rise from 10% of its total logic swing to 90% of its total logic swing.

**ROM Emulator**

A debugging tool that takes the place of or emulates the ROM on the target board. A ROM emulator acts very much like a debug monitor, except that it includes its own serial or network connection to the host.

**ROM Monitor**

A piece of debugging code which usually communicates via a serial connection to a host computer or terminal. Also see “Debug Monitor.”
Routing
The interconnection or the process of creating the desired interconnection of logic cells to make them perform the desired function. Routing follows after partitioning and placement.

RTOS (Real Time Operating System)
Also called real-time multitasking kernel. Software which ensures that time critical events are processed simultaneously and efficiently within a predictable response time. In general, the use of an RTOS simplifies the design process of a system by allowing the application to be divided into multiple independent tasks.

SAN (Storage Area Network)
A high-speed subnetwork of shared storage devices. A storage device is a machine that contains nothing but a disk or disks for storing data. A SAN’s architecture works in a way that makes all storage devices available to all servers on a LAN (Local Area Network) or WAN (Wide Area Network).

Schematic
Graphic representation of a logic design in the form of interconnected gates, flip-flops, and larger blocks. Older and more visually intuitive alternative to the increasingly more popular equation-based or high-level language textual description of a logic design.

SelectMAP Interface
One of the configuration interfaces on the Virtex-II Pro device. This is a byte-serial interface. The pins in the SelectMAP interface can be used as user I/O after configuration has been completed or remain configured as a configuration interface.

SelectRAM
Xilinx-specific name for RAM implemented in CLBs.

SERDES (Serializer/Deserializer)
A common name for a high-speed transceiver that performs both parallel-to-serial and serial-to-parallel conversion.

Simulation
Computer modeling of logic and (sometimes) timing behavior of logic driven by simulation inputs (stimuli or vectors).

Single-Ended
Method of signaling which, unlike differential signaling, only transmits signals over one net.
Skin Effect Loss

Electrical loss in a non-ideal medium due to skin effect. Skin effect is the tendency for high-frequency signal components to travel close to the surface of the medium.

Slice

A subdivision of the Virtex-II Pro CLB. There are four vertical slices in each Virtex-II Pro CLB. Each slice contains two Logic Cells.

SNR (Signal-to-Noise Ratio)

A measure of signal strength relative to background noise. The ratio is usually measured in decibels (dB). If the incoming signal strength in microvolts is $V_S$, and the noise level, also in microvolts, is $V_N$, then the signal-to-noise ratio, SNR, in decibels, is given by the formula:

$$\text{SNR} = 20 \log_{10} \left( \frac{V_S}{V_N} \right)$$

Soft IP

A synthesizable Intellectual Property which can be readily incorporated into an FPGA. Soft IP solves many of the time-to-market issues and also can simplify verification if a proper test bench is included.

Software Interrupt

An interruption of a program that is initiated by a software instruction. Software interrupts are commonly used to implement breakpoints and operating system entry points. Unlike true interrupts, they occur synchronously with respect to program execution.

SONET (Synchronous Optical NETwork)

A standard for connecting fiber-optic transmission systems. SONET was proposed by Bellcore in the mid-1980s and is now an ANSI standard.

SPECint95 and SPECfp95

Acronym for Standard Performance Evaluation Corporation, a nonprofit corporation set up by many computer and microprocessor vendors to create a standard set of benchmark tests. The most widely used set of tests, known as SPEC95, results in two sets of measurements, one for integer operations (SPECint95) and one for floating-point operations (SPECfp95). The SPEC95 benchmark tests are also called CPU95 tests.

SRAM

Static random access memory. Read-Write memory with data stored in latches. Faster than DRAM and with simpler timing requirements, but smaller in size and about four times more expensive than DRAM of the same capacity.
Stack
An area of memory used to implement a data structure that follows the last in, first out method of access. The stack is usually used by the processor to keep track of subroutine calls and returns.

Stack Pointer
A special purpose register that tracks the location of the last entry in the stack.

Static Timing
Detailed description of on-chip logic and interconnect delays.

Submicron
The smallest feature size is usually expressed in microns (\(\mu\) = one millionth of a meter, or a thousandth of a millimeter). The state of the art is moving from 0.35\(\mu\) to 0.25\(\mu\), and soon may reach 0.18\(\mu\). For comparison purposes, the wavelength of visible light is 0.4\(\mu\) to 0.8\(\mu\). One thousandth of an inch, or 1 mil, is 25.4\(\mu\).

Sync Word
A 32-bit word with a value that is used to synchronize the configuration logic.

Synchronous
Circuitry that changes state only in response to a common clock, as opposed to asynchronous circuitry that responds to a multitude of derived signals. Synchronous circuits are easier to design, debug, modify, and better tolerate parameter changes and speed upgrades than asynchronous circuits.

Synthesis
Optimization process of adapting a logic design to the logic resources available on the chip, like look-up tables, Longline, and dedicated carry. Synthesis precedes mapping.

T

TBUFs
Buffers with a 3-state option, where the output can be made inactive. Used for multiplexing different data sources onto a common bus.

Termination
Usually implemented with passive components, termination is used to interface drivers, receivers, and traces that have differing impedance values. Typically, device drivers and receivers do not match the impedance of the PCB trace that connects them.
Termination resistors are employed to match the impedances of these components, maximizing signal transmission and reducing noise.

**Timing**

Relating to delays, performance, or speed.

**Timing Driven**

A design or layout method that takes performance requirements into consideration.

**TLB (Translation Lookaside Buffer)**

TLB is part of an MMU. It has a table used in a virtual memory system keeping track of the physical address page number associated with each virtual address page number. A TLB is used in conjunction with a cache whose tags are based on virtual addresses.

**Trace**

A PPC405 feature which supports tracing of the instruction stream being executed out of the instruction cache in real time.

**Trap**

1. A program interrupt, usually an interrupt caused by some exceptional situation in the user program. In most cases, the OS performs some action, then returns control to the program.
2. Internally generated exceptions that deal with such instances as arithmetic overflow, divide by zero, and bound check failure.

**UART (Universal Asynchronous Receiver/Transmitter)**

An 8-bit parallel-to-serial and serial-to-parallel converter, combined with parity and start-detect circuitry, and sometimes even FIFO buffers. Used widely in asynchronous serial communications interfaces, such as modems.

**UI (Unit Interval)**

Unit of time corresponding to one bit period. A unit interval is the time it takes to send one bit.

**USB (Universal Serial Bus)**

A low-cost, low-speed, self-clocking bit-serial bus (1.5 MHz and 12 MHz) using four wires (VCC, ground, differential data) to daisy-chain up to 128 devices.

**User Mode**

User mode restricts access to some registers and instructions. Normally, application programs operate in this mode.
V

Virtual Address
An intermediate address used to translate an effective address into a physical address. It consists of a process ID and the effective address. It is only used when address translation is enabled.

Virtual Mode
In virtual mode, programs address virtual memory and virtual-memory addresses are translated by the processor into physical-memory addresses. This allows programs to access much larger address spaces than might be implemented in the system.

VME
Older bus standard, popular with MC68000-based industrial computers.

Von-Neuman Architecture
Von-Neuman architecture has a shared bus for instruction and data access. Therefore simultaneous instruction and data transfer is not possible.

W

WAN (Wide Area Network)
A computer network that spans a relatively large geographical area. Typically, a WAN consists of two or more local-area networks (LANs).

Watchdog Timer
A hardware timer that is periodically reset by software. If the software crashes or hangs, the watchdog timer will expire, and the entire system will be reset automatically.

WDM (Wavelength Division Multiplexing)
A type of multiplexing developed for use on optical fiber. WDM modulates each of several data streams onto a different part of the light spectrum.

Word
Four bytes, or 32 bits.

Write Back
A Cache write policy in which data written into the cache by the CPU is not written into main memory until that data line in the cache is to be replaced.
**WRITE Pin**

The WRITE pin is an input to Virtex-II Pro devices in the SelectMAP/Slave Parallel mode, indicating to the device which direction data is flowing on the Data bus. When WRITE is asserted (Low), data is entering the device (configuration). When WRITE is de-asserted (High), data is leaving the device (readback). If WRITE changes state when the device isn’t expecting it, an abort occurs. For more information on the WRITE pin, refer to *The Programmable Logic Data Book*, and in this User Guide, “Design Considerations,” page 67.

**Write Through**

A Cache write policy. A technique for writing data from the CPU simultaneously into the cache and into main memory to assure coherency.

**XNF File**

Xilinx-proprietary description format for a logic design. Alternative is EDIF.
## Index

### Numerics

| 3-state output buffer | 193 |

### A

AC coupling
- defined 521

address mapping
- defined 131

addressing modes
- defined 521

addressing scheme 131

ALU
- defined 521

ASIC
- defined 521

Virtex-II Pro compared with 21–25, 479

asynchronous
- defined 521

ATM
- defined 521

IP core for 276, 279

### B

BER
- defined 522

BGN files 501

bidirectional LVDS 261

bidirectional signals 195

big endian
- defined 522

BIT files
- description 500
- disabling 507

loading downward 510

loading up or down 510

loading upward 511

bit swapping
- description 509

disabling 509

BitGen
- -b option 501
- -d option 501

description 499

disabling DRC 501

DRC file 501

encryption options 267

-g option 501

-h option 507

input files 500

-j option 507

### C

cache
- defined 523

cache line
- defined 523

capacitors

decoupling 482

cascadable shift registers 150

CCLK 289

- defined 524

and configuration mode 289

and master serial programming mode 302

CDR 23

- defined 524

channel bonding
- defined 524

characteristic impedance

See impedance

characteristics

land pads 488

checksum 510

ChipScope ILA 381

classification and export considerations 267

CLB
- defined 524

clearing configuration memory 292

CLK 107

CLK2X 99

CLKDV 99

CLKFB 95

CLKIN 95

clock de-skew 89

clock networks 68–89

clocks 68

buffer input 71

distribution 68

forwarding 250

global buffers 69

global networks 68

multiplexer waveform 81

multiplexers 68

phase shifting 102

resources 69

coherecy
- defined 525

comma
- defined 525

commands

file, executing 510

compiler
- defined 525

memory 147

configuration 289

bitstream
- defined 525

Boundary Scan mode 290

clearing memory 292

-g option 501

JTAG 486

Master SelectMAP mode 290

Master Serial mode 290

mode pins 289

modes 289, 290

process 291

Slave SelectMAP mode 290

Slave Serial mode 290

conflict resolution 124

constraining placement 156

content-addressable memory (CAM) 150
control signals 90
CORE Generator system 271–287
CoreConnect™ bus architecture defined 526
crosstalk 486
defined 526
CS pin defined 526

D
Data Encryption Standard (DES) 266
DC coupling defined 527
DCI 223–241
I/O buffer library 230
software support 230
DCM 89, 89–121
clock de-skew 90
control signals 90
frequency synthesis 89, 98
legacy support 97
miscellaneous timing parameters 63
operating frequency ranges 60
overview 89
phase shifting 89, 102
port signals 95
timing model 60
waveforms 118
DCMs
clock de-skew 89
control signals 90
frequency synthesis 89
phase shifting 89, 102
port signals 95
waveforms 118
DCR defined 527
and processor block timing model 65
DDR I/O 241–255
input 241
output 243
output with 3-state control 245
SDRAM 249
debugger defined 527
hardware 507
debugging
using ChipScope ILA 381
decoupling capacitors 482
dedicated pins 289
DES 266
de-skew circuit 94
differential signaling 262
defined 528
Digital Clock Manager (DCM) 89
DIN pin defined 528
distributed SelectRAM 140–150
DLLs
characteristics 94
source clock input 95
DONE pin defined 528
DOUT pin defined 528
DOUT/BUSY pin defined 528
DRC
disabling for BitGen 501
DRC file 501
DSOCM defined 529
See OCM: data side
dynamic read operations 151

E
EDIF 133, 145, 147, 155, 249, 273, 275
defined 529
effective address defined 529
embedded multipliers 177
timing model 40
timing parameters 40
encryption
BitGen options 267
bitstream 266–270
ESD defined 529
export considerations 267

F
fall time defined 530
FDDRCPE 248
FDDRRSE 248
FIFO 276
defined 530
application notes 133
generating async 122
sync and async using CORE Generator 140
fine phase adjustment 102
flip-chip advantages 478
flip-chip packages 478
frequency synthesis 89
fully synchronous shift registers 157

G
global clock buffers 69
global clock nets 68
global clocks
input to output timing parameters 56
setup and hold timing parameters 58
GTL 207
GTL+ 208

H
Hardware Debugger 507
Harvard architecture defined 532
HSTL_I 209, 213
HSTL_II 210, 214
HSTL_III 211, 215
HSTL_IV 212, 216
HyperTransport™ defined 533
See LDT

I
I/O banks 383
I/O Buffer Information Specification
See IBIS
I/O standards
single-ended 186
IBIS 493, 494
advantages 494
file structure 494
generation 494
I/V and dV/dt curves 495
ramp keyword 495
simulations 496
simulators 497
IEEE 1149.1 498
IEEE 1532 498
impedance defined 533
INIT pin defined 533
input DDR 241
input files
BitGen 500
PROMGen 508
instruction set defined 534
boundary scan 320
IOBs
3-state timing parameters 51
input timing parameters 44
output timing parameters 48
as shift registers 150–160
LVCMOS15 221
LVCMOS18 221
LVCMOS25 222
LVCMOS33 222
LVDS 256–262
defined 536
3-state buffer termination 261
bidirectional 261
primitives 256
transmitter termination 259
LVPECL 263–265
primitives 263
receiver termination 264
transmitter termination 265
LVTTL 220

N
National Institute of Standards and Technology (NIST) 266
NO_CHANGE mode 124

O
OBUF 191
OBUF 193
OCM
defined 539
and processor block timing model 65
on-chip peripheral bus
See OPB
OPB
defined 539
and MicroBlaze™ 285
IP cores for 278
IP for 286
operating frequency ranges 60
optimization
defined 539
output buffer (OBUF) 191
output DDR 243
with 3-state control 245
output drive strength 198
output files
BitGen 500
name, PROMGen 511
overwriting 507
PROMGen 508
output power/ground pairs 200
overshoot
defined 539
overview of user guide 17

P
Packages
flip-chip 478
thermal considerations 478
FF1148
package diagram 474
banked pin diagram
XC2VP40 446
XC2VP50 444
composite pin diagram
XC2VP40 445
XC2VP50 443
dedicated pin diagram
XC2VP40 447
XC2VP50 447
FF1152
package diagram 473
banked pin diagram
XC2VP20 440
XC2VP30 437
XC2VP40 434
XC2VP50 434
composite pin diagram
XC2VP20 439
XC2VP30 436
XC2VP40 433
XC2VP50 431
dedicated pin diagram
XC2VP20 441
XC2VP30 438
XC2VP40 435
XC2VP50 432

FF1517
package diagram 475
banked pin diagram
XC2VP50 452
XC2VP70 450
composite pin diagram
XC2VP50 451
XC2VP70 449
dedicated pin diagram
XC2VP50 453
XC2VP70 453

FF1696
package diagram 477
banked pin diagram
XC2VP100 464
composite pin diagram
XC2VP100 463
dedicated pin diagram
XC2VP100 465

FF1704
package diagram 476
banked pin diagram
XC2VP100 456
XC2VP70 458
XC2VPX70 461
composite pin diagram
XC2VP100 455
XC2VP70 457
XC2VPX70 460
dedicated pin diagram
XC2VP100 459
XC2VP70 459
XC2VPX70 462

FF672
package diagram 471
banked pin diagram
XC2VP2 421
XC2VP4 419
XC2VP7 416
composite pin diagram
XC2VP2 420
XC2VP4 418
XC2VP7 415
dedicated pin diagram
XC2VP2 422

pads 488
defined 539
Parallel Cable IV
defined 539
parallel termination 484
Pb-free packages 383
PC20-84 specification 514
PCB layout considerations 481
PCF files
BitGen 500
PCI
defined 540
PCI33_3 220
PC166_3 220
persist option 290
phase shifting 89, 102
physical address
defined 540
pin-locking
defined 540
pinout diagrams 395
pins 289
chip enable
See CS Pin 526
CS
defined 526
dedicated 289
DIN
defined 528
DONE
defined 528
DOUT
defined 528
DOUT/BUSY
defined 528
dual-function 290
INIT
defined 533
power 292
PROGRAM
defined 542
types 384
WRITE
defined 549
pin-to-pin timing model 55
placement
constraints 156
defined 540
PLB
defined 541
port addressing scheme 131
port signals 95
power analysis software 493
power pins 292
PowerPC 405 processor block
timing model 65
pre-emphasis
defined 541
primitives
| **LVDS** 256, 263 | challenges 487 |
| PRM files 509 | strategy 488 |
| PROGRAM pin | RST 95 |
| PROMGen |
| -b option 509 | defined 542 |
| -c option 510 | parallel 484 |
| -d option 510 | series 485 |
| description 507, 508 | techniques 199 |
| examples 512 | thermal considerations 478 |
| flow diagram 507 | thermal management 480 |
| -help option 510 | Timing Analyzer 27 |
| input files 508 | timing models 27 |
| -l option 510 | block SelectRAM 36 |
| -n option 510 | DCM 60 |
| -o option 511 | embedded multiplier 40 |
| options 509 | IOB 43 |
| output file name 511 | pin-to-pin 55 |
| output files 508 | PPC405 65 |
| -p option 511 | RocketIO MGT 65 |
| -r option 511 | timing parameters |
| -s option 511 | block SelectRAM 37 |
| supported families 507 | embedded multiplier 40 |
| -u option 511 | general slice 29 |
| -x option 511 | global clock input to output 56 |
| **S** | global clock setup and hold 58 |
| SelectIO-Ultra | IOB 3-state 51 |
| single-ended resources 185–223 | IOB input 44 |
| SelectMAP | IOB output 48 |
| See configuration modes | miscellaneous DCM 63 |
| SERDES | slice distributed RAM 32 |
| defined 544 | slice SRL 35 |
| series termination 485 | TMULT 40 |
| shift registers | transmission line effects 199, 483 |
| cascadable 150 | TRCE 27 |
| fully synchronous 157 | Triple Data Encryption Algorithm (TDEA) 266 |
| operation 150 | Triple DES 266 |
| static length 158 | **V** |
| signals | VBATT 269, 385, 517 |
| bidirectional 195 | VCC decoupling 481 |
| simulation 24, 118, 128 | VCCAUX 385 |
| defined 544 | VCCINT 385 |
| simultaneous-switching output (SSO) | VCCO 198, 385 |
| 200 | verification |
| single-ended | using ChipScope ILA 381 |
| defined 544 | VHDL and Verilog templates 114, 134, 143, 148, 159, 167, 173, 184, 250 |
| I/O standards 186 | Virtex-II Pro |
| SelectIO-Ultra resources 185–223 | compared with ASIC 21–25, 479 |
| Slave SelectMAP mode 290 | DCI 223–241 |
| Slave Serial mode 290 | DES 266 |
| slew rate 198 | LUTs 150 |
| slices 171 | multiplexers 161 |
| defined 545 | package specifications 467 |
| SO20 specification 515 | pinout diagrams 395 |
| solder balls 487 | slices 171 |
| specifications | VQ44 specification 516 |
| PC20-84 514 | VREF 198 |
| PROM packages 513 | determined 546 |
| SO20 515 | parallel 484 |
| VQ44 516 | series 485 |
| SRL16 150 | techniques 199 |
| SRLC16 150 | thermal considerations 478 |
| SSTL2_1 217 | thermal management 480 |
| SSTL2_II 218, 219 | Timing Analyzer 27 |
| STARTUP_WAIT attribute 95 | timing models 27 |
| static length shift registers 158 | block SelectRAM 36 |
| static read operations 152 | DCM 60 |
| Sum of Products (SOP) 171–177 | embedded multiplier 40 |
| synchronous | IOB 43 |
| defined 546 | pin-to-pin 55 |
| synchronous DRAM 249 | PPC405 65 |
| RocketIO transceiver | RocketIO MGT 65 |
| loopback | timing parameters |
| defined 536 | block SelectRAM 37 |
| timing model 65 | embedded multiplier 40 |
| routability guidelines 487 | general slice 29 |
| routing | global clock input to output 56 |
| defined 544 | global clock setup and hold 58 |
| | IOB 3-state 51 |
| | IOB input 44 |
| | IOB output 48 |
| | miscellaneous DCM 63 |
| | slice distributed RAM 32 |
| | slice SRL 35 |
| **R** |
| rawbits file 501 | TMULT 40 |
| RBT files 501 | transmission line effects 199, 483 |
| read operations | TRCE 27 |
| dynamic 151 | Triple Data Encryption Algorithm (TDEA) 266 |
| static 152 | Triple DES 266 |
| READ_FIRST mode 124 | **V** |
| readback | VBATT 269, 385, 517 |
| defined 542 | VCC decoupling 481 |
| register | VCCAUX 385 |
| defined 543 | VCCINT 385 |
| ringing | VCCO 198, 385 |
| defined 543 | verification |
| on PC board signals 484 | using ChipScope ILA 381 |
| rise time | VHDL and Verilog templates 114, 134, 143, 148, 159, 167, 173, 184, 250 |
| defined 543 | Virtex-II Pro |
| RocketIO transceiver | compared with ASIC 21–25, 479 |
| loopback | DCI 223–241 |
| defined 536 | DES 266 |
| timing model 65 | LUTs 150 |
| routability guidelines 487 | multiplexers 161 |
| routing | package specifications 467 |
| defined 544 | pinout diagrams 395 |
| | slices 171 |
| | VQ44 specification 516 |
| | VREF 198 |
write-input multiplexers 165
WRITE pin
defined 549
WRITE_FIRST mode 123
<table>
<thead>
<tr>
<th>Sales Offices</th>
<th>Headquarters</th>
<th>European Headquarters</th>
</tr>
</thead>
<tbody>
<tr>
<td>2100 Logic Drive</td>
<td>Benchmark House, 203 Brooklands Rd.</td>
<td></td>
</tr>
<tr>
<td>San Jose, CA 95124</td>
<td>Weybridge Surrey KT13 0RH</td>
<td>United Kingdom</td>
</tr>
<tr>
<td>Tel: (408) 559-7778</td>
<td>Tel: +44-1-870-7350-600</td>
<td>Fax: +44-1-870-7350-601</td>
</tr>
<tr>
<td>Fax: (408) 559-7114</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TWX: (510) 600-8750</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>North America</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Ledgewood, NJ</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Raleigh, NC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Brecksville, OH</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Portland, OR</td>
<td></td>
</tr>
<tr>
<td></td>
<td>West Chester, PA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Dallas, TX</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Salt Lake City, UT</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bellevue, WA</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Oakville, Ontario Canada</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kanata, Ontario Canada</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Madison, AL</td>
<td>Tel: (973) 584-7199</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax: (973) 584-1390</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Phoenix, AZ</td>
<td>Tel: (919) 846-3922</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax: (919) 846-8316</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Irvine, CA</td>
<td>Tel: (330) 659-3131</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax: (330) 659-9254</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>San Diego, CA</td>
<td>Tel: (503) 293-9016</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax: (503) 293-3858</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sunnyvale, CA</td>
<td>Tel: (610) 430-3300</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax: (610) 430-0470</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Greenwood Village, CO</td>
<td>Tel: (408) 245-9850</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax: (408) 245-9865</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Winter Park, FL</td>
<td>Tel: (949) 727-0780</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax: (949) 727-3128</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Schaumburg, IL</td>
<td>Tel: (847) 605-1972</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax: (847) 605-1976</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Deephaven, MN</td>
<td>Tel: (612) 473-4816</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax: (612) 473-5060</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Marriottsville, MD</td>
<td>Tel: (410) 442-9748</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax: (410) 442-9749</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Nashua, NH</td>
<td>Tel: (603) 891-1098</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax: (603) 891-0890</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For information on Xilinx North American Sales Representative offices, see [http://www.xilinx.com/company/sales/na_reps.htm](http://www.xilinx.com/company/sales/na_reps.htm)

For information on Xilinx International Sales Representative offices, see [http://www.xilinx.com/company/sales/int_reps.htm](http://www.xilinx.com/company/sales/int_reps.htm)