Quick Points

- HW #2 coming out over the weekend
  - Due Thursday, September 21 (12:00pm)
    - LUT mapping
    - Comparing FPGA devices
    - Synthesizing arithmetic operators

Recap

- Hard-wired carry logic support

Altera FLEX 8000  
Xilinx XCV4000

Recap (cont.)

- If one operand is constant:
  - More speed?
  - Less hardware?

Recap (cont.)

- Square-root carry select adders

Recap (cont.)

- Carry save multiplication
Recap (cont.)

- If one operand is constant:
  - Can greatly reduce the number of adders
  - Removes all and gates

\[ \begin{align*}
  Y_0 &= 0 \\
  Z_0 &= X_0 + X_1 + X_2 + X_3 \\
  Y_1 &= 1 \\
  Y_2 &= 0 \\
  Z_1 &= 10101011 \\
  \end{align*} \]

LUT-Based Constant Multipliers

- Constants can be changed in the LUTs to program new multipliers

\[ \begin{align*}
  10101011 \times \begin{pmatrix} X_1 & X_2 & X_3 \end{pmatrix} &= S_0 - S_15 \\
  \begin{pmatrix} A_4 & A_5 \end{pmatrix} &= X_1 + X_2 + X_3 \\
  \end{align*} \]

Outline

- Recap
- More Multiplication
- Handling Fractional Values
  - Fixed Point
  - Floating Point
- Some Modern FPGA Devices
  - Xilinx – XC5200, Virtex (-II / -II Pro / -4 / -5), Spartan (-II / -3)
  - *Altera – FLEX 10K, APEX (20K / II), ACEX 1K, Cyclone (II), Stratix (GX / II / II GX)

Partial Product Generation

- AND gates in multiplication are wasteful
  - Option 1 – use cascade logic
  - Option 2 – break into smaller (2x2) multipliers

\[ \begin{align*}
  42 &= \begin{pmatrix} 101010 \end{pmatrix} \text{ Multiplicand} \\
  \times 11 &= \begin{pmatrix} 1011 \end{pmatrix} \text{ Multiplier} \\
  &= \begin{pmatrix} 0110 & 0110 & 0110 & 0100 & 0100 & 0100 & + \end{pmatrix} \\
  &= \begin{pmatrix} 0110 & 0110 & 0110 & 0100 & 0100 & 0100 & + \end{pmatrix} \text{ Product} \\
  &= \begin{pmatrix} 0111001110 \end{pmatrix} \text{ Product} \\
  \end{align*} \]

Representation Compression

- Multiplication can be simplified if the representation is compressed
  - Standard – binary representation \((0,1)\times 2^n\)
  - Canonical Signed Digit (CSD) representation \((-1,0,1)\times 2^n\)
- To encode CSD:
  - Set \( C = (B + (B ‘<<1 )) \)
  - Calculate \(-2C = 2^{n}(C ‘>>1)\)
  - \( D_i = B_i + C_i – 2C_{i+1} \) where \( C_{i+1} \) is the carryout of \( B_i + C_i \)
- Example: \( B = 61d = 01110101 \)
  - \( C = 01111011\) and \( B = 01110110 \) with padding
- For any \( n \) bit number, there can only be \( n/2 \) nonzero digits in a CSD representation (every other bit)

Booth Encoding

- Variation on CSD encoding:
  - \( E_i = -2B_i + B_{i-1} + B_{i-2} \)
  - Select a group of 3 digits, add the two least significant digits, and then subtract \( 2x \) the most significant bit
  - \( E_i \) is \((-2, -1, 0, 1)\times 2^n\)
- Example:
  - \( B = 61d = 01110101b + 00000000b \text{ (with padding)} \)
  - \( E = 010(-1) \)
- Reduces the number of partial products for multiplication by \( \frac{1}{2} \)
- Can automatically handle negative numbers
Fractional Arithmetic

- Many important computations require fractional components
- Fractional arithmetic often ignored in FPGA literature
  - Complex standards (ex. IEEE special cases)
  - Resource intensive and slow
- Why not just extend the binary representation past the decimal point?

Fixed-Point Representation

- Separate value into Integer (I) and Fractional remainder (F)
  
  \[ I \quad F \]

- F bits represent \( \{0,1\} \times 2^{-n} \)
- How large to make I and F depends on application
  - Ex: Q16.16 is 16 bits of integer \([-2^{15}, 2^{16})\] with 16 bits of fraction – increments of \(2^{-16}\) or 0.0000152587890625
  - Ex: Q1.127 is a normalized integer \([-1,1)\] with 127 bits of fraction – increments of \(2^{-127}\) or 5.8774717541114375398436826861112e-39

Fixed-Point Arithmetic

- Addition, subtraction the same (Q4.4 example):

  
  \[
  
  \begin{array}{c}
  3.6250 \quad 0011.1010 \\
  + 2.8125 \quad 0010.1101 \\
  \hline
  6.4375 \quad 0110.0111 
  \end{array}
  
  \]

- Multiplication requires realignment:

  \[
  
  \begin{array}{c}
  3.6250 \quad 0011.1010 \\
  \times 2.8125 \quad 0010.1101 \\
  \hline
  00111010 \\
  00111010 \\
  00111010 \\
  00111010 \\
  \hline
  10.1953125 \quad 1010.00110010 
  \end{array}
  
  \]

Fixed-Point Issues

- Overflow/underflow
- Quantization Errors
  - After rounding down previous example
    \[ 3.625 \times 2.8125 = 10.1875 \] (0.08% error)
  - In Q4.4, 2 divided by 3 = 0.625 (6.25% error)
- Scaling
- Dynamic range needed for some applications

IEEE 754 Floating Point

- Single precision: \( V = (-1)^S \times 2^{E-127} \times (1.F) \)

  \[
  \begin{array}{c}
  1 \quad 8 \quad 23 \\
  \hline
  S \quad E \quad F 
  \end{array}
  
  \]

- Double precision: \( V = (-1)^S \times 2^{E-1023} \times (1.F) \)

  \[
  \begin{array}{c}
  1 \quad 11 \quad 52 \\
  \hline
  S \quad E \quad F 
  \end{array}
  
  \]

- Special conditions – not a number (NaN), ±0, ±infinity
- Gradual underflow

Floating Point FPGA Hardware

- Xilinx XCV4085
- Addition
  - Single-precision – 587 4-LUTs
  - Double-precision – 1334 4-LUTs
- Multiplication
  - Single-precision – 1661 4-LUTs
  - Double-precision – 4381 4-LUTs
- Division
  - Single-precision – 1583 4-LUTs
  - Double-precision – 4910 4-LUTs
- For double-precision, can only fit any two of three units on a single device!
- See [Und04] for details
### Capacity Trends

<table>
<thead>
<tr>
<th>Year</th>
<th>Device</th>
<th>Logic Cells</th>
<th>Max Logic Gates</th>
<th>VersaBlock Array</th>
<th>CLBs</th>
<th>I/Os</th>
</tr>
</thead>
<tbody>
<tr>
<td>1985</td>
<td>XC2000</td>
<td>50 MHz</td>
<td>1K gates</td>
<td>8x8</td>
<td>64</td>
<td>84</td>
</tr>
<tr>
<td>1987</td>
<td>XC4000</td>
<td>100 MHz</td>
<td>250K gates</td>
<td>10x12</td>
<td>120</td>
<td>112</td>
</tr>
<tr>
<td>1989</td>
<td>Virtex</td>
<td>200 MHz</td>
<td>1M gates</td>
<td>14x14</td>
<td>196</td>
<td>148</td>
</tr>
<tr>
<td>1991</td>
<td>Virtex-E</td>
<td>240 MHz</td>
<td>4M gates</td>
<td>18x18</td>
<td>324</td>
<td>224</td>
</tr>
<tr>
<td>1995</td>
<td>XC5200</td>
<td>50 MHz</td>
<td>23K gates</td>
<td>22x22</td>
<td>484</td>
<td>484</td>
</tr>
<tr>
<td>1999</td>
<td>Virtex-II Pro</td>
<td>450 MHz</td>
<td>8M gates*</td>
<td>22x22</td>
<td>1,936</td>
<td>1,936</td>
</tr>
<tr>
<td>2004</td>
<td>Virtex-4</td>
<td>500 MHz</td>
<td>16M gates*</td>
<td>22x22</td>
<td>4,096</td>
<td>4,096</td>
</tr>
<tr>
<td>2006</td>
<td>Virtex-5</td>
<td>550 MHz</td>
<td>24M gates*</td>
<td>22x22</td>
<td>8,192</td>
<td>8,192</td>
</tr>
</tbody>
</table>

### Xilinx XC5200 FPGA

- Successor to the XC4000
- Relatively small amount of CLBs with faster interconnect

<table>
<thead>
<tr>
<th>Device</th>
<th>XC5202</th>
<th>XC5204</th>
<th>XC5206</th>
<th>XC5210</th>
<th>XC5215</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Cells</td>
<td>256</td>
<td>480</td>
<td>784</td>
<td>1,296</td>
<td>1,936</td>
</tr>
<tr>
<td>CLBs</td>
<td>64</td>
<td>120</td>
<td>196</td>
<td>324</td>
<td>484</td>
</tr>
<tr>
<td>Flip-Flops</td>
<td>256</td>
<td>480</td>
<td>784</td>
<td>1,296</td>
<td>1,936</td>
</tr>
<tr>
<td>I/Os</td>
<td>84</td>
<td>124</td>
<td>148</td>
<td>196</td>
<td>244</td>
</tr>
</tbody>
</table>

### Xilinx Spartan FPGAs

- Meant to be low-power / low-cost version of XC4000 series (on newer process technology)

<table>
<thead>
<tr>
<th>Device</th>
<th>XC5205</th>
<th>XC5210</th>
<th>XC5220</th>
<th>XC5230</th>
<th>XC5240</th>
</tr>
</thead>
<tbody>
<tr>
<td>Logic Cells</td>
<td>238</td>
<td>466</td>
<td>950</td>
<td>1,368</td>
<td>1,862</td>
</tr>
<tr>
<td>Max Logic Gates</td>
<td>5,000</td>
<td>10,000</td>
<td>20,000</td>
<td>30,000</td>
<td>40,000</td>
</tr>
<tr>
<td>CLB Matrix</td>
<td>10 x 10</td>
<td>14 x 14</td>
<td>20 x 20</td>
<td>24 x 24</td>
<td>28 x 28</td>
</tr>
<tr>
<td>Total CLBs</td>
<td>100</td>
<td>196</td>
<td>400</td>
<td>576</td>
<td>784</td>
</tr>
<tr>
<td>Flip-Flops</td>
<td>360</td>
<td>616</td>
<td>1,120</td>
<td>1,536</td>
<td>2,016</td>
</tr>
<tr>
<td>I/Os</td>
<td>77</td>
<td>112</td>
<td>160</td>
<td>192</td>
<td>224</td>
</tr>
<tr>
<td>Dist. RAM Bits</td>
<td>3,200</td>
<td>6,272</td>
<td>12,800</td>
<td>18,432</td>
<td>25,088</td>
</tr>
</tbody>
</table>
Xilinx Spartan (cont.)

- Individual LUTs can be programmed as 16x1 RAMs and combined to form larger memory structures

Xilinx Virtex FPGAs

<table>
<thead>
<tr>
<th>Device</th>
<th>Logic Cells</th>
<th>Max Logic Gates</th>
<th>CLB Array</th>
<th>I/O Bits</th>
<th>Block RAM Bits</th>
<th>Select RAM+ Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>XCV50</td>
<td>1,728</td>
<td>57,906</td>
<td>16 x 24</td>
<td>180</td>
<td>32,768</td>
<td>24,576</td>
</tr>
<tr>
<td>XCV100</td>
<td>2,700</td>
<td>108,904</td>
<td>20 x 30</td>
<td>180</td>
<td>40,960</td>
<td>38,400</td>
</tr>
<tr>
<td>XCV150</td>
<td>3,888</td>
<td>164,674</td>
<td>24 x 38</td>
<td>260</td>
<td>49,152</td>
<td>55,296</td>
</tr>
<tr>
<td>XCV200</td>
<td>5,292</td>
<td>238,666</td>
<td>28 x 42</td>
<td>284</td>
<td>57,844</td>
<td>75,264</td>
</tr>
<tr>
<td>XCV250</td>
<td>6,912</td>
<td>322,970</td>
<td>32 x 48</td>
<td>316</td>
<td>65,536</td>
<td>98,304</td>
</tr>
<tr>
<td>XCV300</td>
<td>10,800</td>
<td>468,252</td>
<td>40 x 60</td>
<td>404</td>
<td>81,920</td>
<td>153,600</td>
</tr>
<tr>
<td>XCV350</td>
<td>15,552</td>
<td>661,111</td>
<td>48 x 72</td>
<td>512</td>
<td>98,304</td>
<td>221,184</td>
</tr>
<tr>
<td>XCV400</td>
<td>21,168</td>
<td>888,439</td>
<td>56 x 84</td>
<td>512</td>
<td>114,688</td>
<td>301,058</td>
</tr>
<tr>
<td>XCV450</td>
<td>27,648</td>
<td>1,124,022</td>
<td>64 x 96</td>
<td>512</td>
<td>131,072</td>
<td>393,216</td>
</tr>
</tbody>
</table>

Xilinx Virtex (cont.)

- 4 4-LUTs / FFs per CLB
- Organized into 2 “slices”

Xilinx Virtex (cont.)

- Block Select+RAM – dedicated blocks of on-chip, true dual port read/write synchronous RAM
- 4Kbit of RAM with different aspect ratios
- Faster, less flexible than distributed RAM using LUTs

Virtex-E – updated, larger version of Virtex devices

Xilinx Spartan-II

- CLB structure similar to Virtex

<table>
<thead>
<tr>
<th>Device</th>
<th>Logic Cells</th>
<th>System Gates</th>
<th>CLB Array</th>
<th>I/O Bits</th>
<th>Distributed RAM Bits</th>
<th>Select RAM Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC2S15</td>
<td>432</td>
<td>15,000</td>
<td>8 x 12</td>
<td>86</td>
<td>6,144</td>
<td>16,384</td>
</tr>
<tr>
<td>XC2S30</td>
<td>972</td>
<td>30,000</td>
<td>12 x 18</td>
<td>92</td>
<td>13,824</td>
<td>24,576</td>
</tr>
<tr>
<td>XC2S50</td>
<td>1,728</td>
<td>50,000</td>
<td>16 x 24</td>
<td>176</td>
<td>24,576</td>
<td>32,768</td>
</tr>
<tr>
<td>XC2S100</td>
<td>2,700</td>
<td>100,000</td>
<td>20 x 30</td>
<td>176</td>
<td>38,400</td>
<td>48,960</td>
</tr>
<tr>
<td>XC2S150</td>
<td>3,888</td>
<td>150,000</td>
<td>24 x 36</td>
<td>260</td>
<td>55,296</td>
<td>49,152</td>
</tr>
<tr>
<td>XC2S200</td>
<td>5,292</td>
<td>200,000</td>
<td>28 x 42</td>
<td>284</td>
<td>75,264</td>
<td>57,344</td>
</tr>
</tbody>
</table>

Xilinx Virtex-II Platform FPGAs

- “Platform” FPGA == Multiplier??
Xilinx Virtex-II (cont.)
- 4 Slices per CLB, 2 4-LUTs per slice
- 8 LUTs per CLB
- Block Select+RAMs now 18Kbit each

Block Multipliers
- Synthesis tools can take larger multipliers and break them down into 18x18 multipliers

Xilinx Virtex-II Pro FPGAs

PASCAL
Xilinx Virtex-II Pro (cont.)
- PowerPC processor block features
  - 300+ MHz Harvard architecture (RISC)
  - Five-stage pipeline
  - Hardware multiply/divide
  - Thirty-two 32-big GPRs
  - 16 KB two-way instruction cache
  - 16 KB two-way data cache
  - On-Chip Memory (OCM) interface
  - IBM CoreConnect (OPB, PLB) interfaces

PPC 405 details
### Xilinx Spartan-3 FPGAs

- CLB structure similar to Virtex-II

<table>
<thead>
<tr>
<th>Device</th>
<th>System Gates</th>
<th>CLB Array</th>
<th>Multiplier Blocks</th>
<th>Max I/O Pads</th>
<th>Distr. RAM Bits</th>
<th>Select RAM* Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC3S50</td>
<td>50K</td>
<td>16 x 12</td>
<td>4</td>
<td>124</td>
<td>12K</td>
<td>72K</td>
</tr>
<tr>
<td>XC3S200</td>
<td>200K</td>
<td>24 x 20</td>
<td>12</td>
<td>173</td>
<td>30K</td>
<td>216K</td>
</tr>
<tr>
<td>XC3S400</td>
<td>400K</td>
<td>32 x 28</td>
<td>16</td>
<td>264</td>
<td>56K</td>
<td>288K</td>
</tr>
<tr>
<td>XC3S1000</td>
<td>1M</td>
<td>48 x 40</td>
<td>24</td>
<td>391</td>
<td>120K</td>
<td>432K</td>
</tr>
<tr>
<td>XC3S1500</td>
<td>1.5M</td>
<td>64 x 52</td>
<td>32</td>
<td>487</td>
<td>208K</td>
<td>576K</td>
</tr>
<tr>
<td>XC3S2000</td>
<td>2M</td>
<td>80 x 64</td>
<td>40</td>
<td>565</td>
<td>320K</td>
<td>720K</td>
</tr>
<tr>
<td>XC3S4000</td>
<td>4M</td>
<td>96 x 72</td>
<td>96</td>
<td>712</td>
<td>432K</td>
<td>1,728K</td>
</tr>
<tr>
<td>XC3S5000</td>
<td>5M</td>
<td>104 x 80</td>
<td>104</td>
<td>784</td>
<td>520K</td>
<td>1,872K</td>
</tr>
</tbody>
</table>

### Xilinx Virtex-4 FPGAs

- Comes in three varieties:
  - Virtex-4 LX: most amount of LUTs
  - Virtex-4 FX: has PowerPCs like V2P
  - Virtex-4 SX: contains most amount of XtremeDSP slices
- CLB structure similar to Virtex-II
  - Largest LX device – 89,088 slices = 178,176 4-LUTs!
  - FX devices limited to 2 PPC 405s like Virtex-II Pro
- XTremeDSP Slices:
  - Same 18x18 block multiplier, now with optional pipelining
  - Includes built-in 48-bit accumulator for MAC operations

### Xilinx Virtex-5

- CLB slices uses 6-input LUTs
- Block RAMs now 36Kbits per block
- DSP slices now support 25x18 MAC
- Diagonal routing
- Only LX series available now

### Summary

- Handling fractional math in hardware is important, and expensive
  - Data point – 3 double-precision dividers in a Xilinx XC2VP30
  - Data point – cannot fit a double-precision multiplier in a Xilinx XC3S50
  - Fixed point an alternative, but not practical for all applications
- Xilinx FPGAs
  - 4-LUTs arranged in slices, CLBs (except for V5)
  - Physical SRAM blocks for fast memory
  - Physical multipliers for fast DSP operations
  - Some physical CPUs to manage embedded systems