CprE / ComS 583 Reconfigurable Computing

Prof. Joseph Zambreno
Department of Electrical and Computer Engineering
Iowa State University

Lecture #14 – FPGA Design Automation

Quick Points

- Course Deadlines
  - Project proposals – Sunday, September 30
    - Not all groups accounted for
  - HW #3 – Tuesday, October 9
  - Midterm – Tuesday, October 16
    - Assigned next week Tuesday (following conceptual review in class)
    - Short, not a homework
    - Work individually

Synthesis

syn·the·sis (sin'thu-sis) *n.* – the combining of the constituent elements of separate material or abstract entities into a single or unified entity

- For hardware, the "abstract entity" is a circuit description
- "Unified entity" is a hardware implementation
- Hardware compilation (but not really)

FPGA Synthesis

- The term "synthesis" has become overloaded in the FPGA world
- Examples:
  - System synthesis
  - Behavioral / high-level / algorithmic synthesis
  - RT-level synthesis
  - Logic synthesis
  - Physical synthesis
- Our usage: FPGA synthesis = behavioral synthesis + logic synthesis + physical synthesis

Logic Synthesis

- Input – Boolean description
- Goal – to develop an optimized circuit representation based on the logic design
  - Boolean expressions are converted into a circuit representation (gates)
  - Takes into consideration speed/area/power requirements of the original design
- For FPGA, need to map to LUTs instead of logic gates (technology mapping)

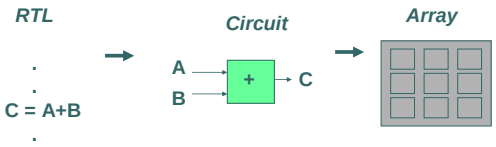
Behavioral Synthesis

- Inputs
  - Control and data flow graph (CDFG)
  - Cell library
    - Ex: fast adder, slow adder, multiplier, etc.
    - Speed/area/power characteristics
  - Constraints
    - Total speed/area/power
- Output
  - Datapath and control to implement

Outline

- Quick Points
- Introduction
- FPGA Design Flow
- FPGA Synthesis
  - Logic Synthesis
  - Behavioral Synthesis
- FPGA Placement and Routing
  - Metrics
  - Placement Techniques
  - Routing Techniques

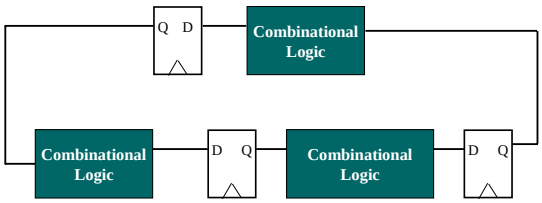
FPGA Design Translation



- CAD to translate circuit from text description to physical implementation well understood
- Most current FPGA designers use register-transfer level specification (allocation and scheduling)
- Same basic steps as ASIC design

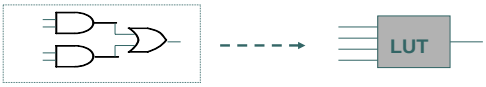
Register Transfer-Level Design

- A register-transfer machine has combinational logic connecting registers:

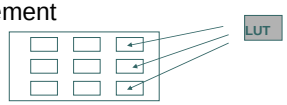


FPGA Circuit Compilation

- Technology Mapping

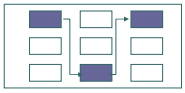


- Placement



Assign a logical LUT to a physical location

- Routing

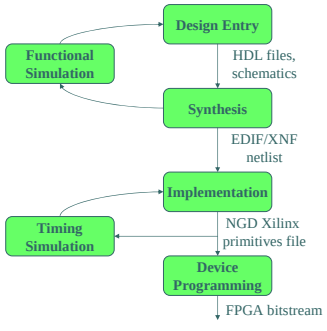


Select wire segments and switches for interconnection

Standard FPGA Design Flow

- Design Entry
- Synthesis
  - Design abstracted as a list of operations and dependencies
  - Transformed into state diagrams and then logic networks (netlist)
- Design Implementation
  - Translate – merges multiple design files into a single netlist
  - Map – groups logical components from netlist into IOBs and CLBs
  - Place & Route – place components on the FPGA and connect them
- Device File Programming
  - Generates a bitstream containing CLB/IOB configuration and routing information to be directly loaded onto the FPGA

FPGA Design Flow (Xilinx)



Design Flow with Test

Design and implement a simple unit permitting to speed up encryption with RC5-similar cipher with fixed key set on 8031 microcontroller. Unlike in the experiment 5, this time your unit has to be able to perform an encryption algorithm by itself, executing 32 rounds…..

Specification

```
Library IEEE;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity RC5_core is
        port(
                clock, reset, encr_decr: in std_logic;
                data_input: in std_logic_vector(31 downto 0);
                data_output: out std_logic_vector(31 downto 0);
                out_full: in std_logic;
                key_input: in std_logic_vector(31 downto 0);
                key_read: out std_logic;
        );
end RC5_core;
```

VHDL description

Functional simulation

Synthesized Circuit

Post-synthesis simulation

Design Flow with Test (cont.)

Synthesized Circuit

Post-synthesis simulation

Implementation

Timing simulation

Configuration

On chip testing

Synthesis Tools

- Interpret RTL code
- Produce synthesized circuit netlist in a standard EDIF format
- Give preliminary performance estimates
- Display circuit schematic corresponding to EDIF netlist

```
Performance Summary
*******************

Worst slack in design: -0.924

                           Requested     Estimated     Requested     Estimated
                             Clock         Clock
Starting Clock     Frequency     Frequency     Period          Period
   Slack      Type               Group
------------------------------------------------------------------------------
   --------------------
exam1|clk          85.0 MHz      78.8 MHz      11.765        12.688        -0.924
   inferred     Inferred_clkgroup_0
System             85.0 MHz      86.4 MHz      11.765        11.572         0.193
   system       default_clkgroup
==============================================================
```


Implementation

Synthesis

XILINX®

Circuit netlist

Timing Constraints

Constraint Editor

Electronic Design Interchange Format

Native Constraint File

EDIF NCF UCF

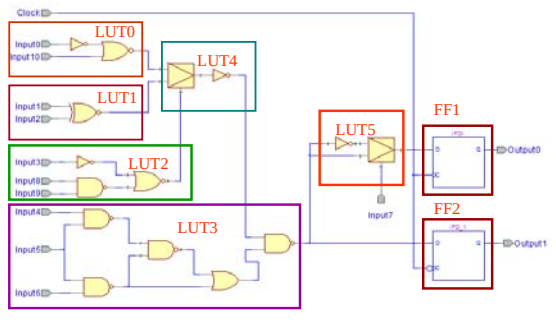
User Constraint File

Implementation

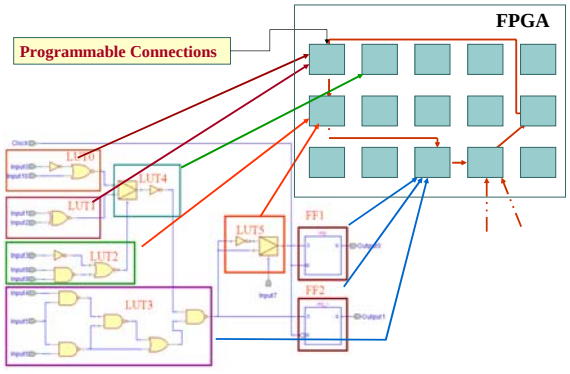
NGD

Native Generic Database file

Circuit Netlist and Mapping



Placing and Routing



Place and Route Report

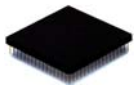
```
Timing Score: 0

Asterisk (*) preceding a constraint indicates it was not met.
   This may be due to a setup or hold violation.

----------------------------------------------------------------------------
  Constraint                                | Requested  | Actual     | Logic
                                            |            |            | Levels
----------------------------------------------------------------------------
  TS_clk = PERIOD TIMEGRP "clk" 11.765 ns   | 11.765ns   | 11.622ns   | 13
  HIGH 50%                                  |            |            |
----------------------------------------------------------------------------
  OFFSET = OUT 11.765 ns AFTER COMP "clk"   | 11.765ns   | 11.491ns   | 1
----------------------------------------------------------------------------
  OFFSET = IN 11.765 ns BEFORE COMP "clk"   | 11.765ns   | 11.442ns   | 2
----------------------------------------------------------------------------
```


Configuration

- Once a design is implemented, you must create a file that the FPGA can understand
  - This file is called a bit stream: a BIT file (.bit extension)
- The BIT file can be downloaded directly to the FPGA, or can be converted into a PROM file which stores the programming information



Logic Synthesis

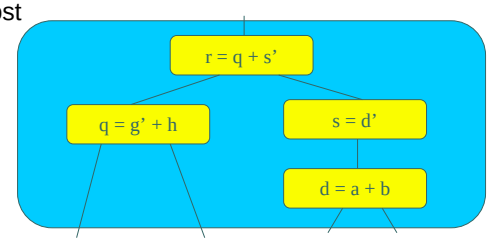
- Syntax-based translation
  - Translate HDL into logic directly (ab + ac)
  - Generally requires optimization
- Macros
  - Pre-designed logic
  - Generally identified by language features
- Hard macro: includes placement
- Soft macro: no placement

Logic Synthesis Phases

- **Technology-independent** optimizations
  - Works on Boolean expression equivalent
  - Estimates size based on number of literals
  - Uses factorization, resubstitution, minimization to optimize logic
  - Technology-independent phase uses simple delay models
- **Technology-dependent** optimizations
  - Maps Boolean expressions into a particular cell library
  - Mapping may take into account area, delay
  - Allows more accurate delay models
- Transformation from technology-independent to technology-dependent is called **library binding**

LUT-based Logic Synthesis

- Cost metric for static gates is literal
  - ax + bx' has four literals, requires 8 transistors
- Cost metric for FPGAs is logic element
- All functions that fit in an LE have the same cost



Behavioral Synthesis

- Sequential operation is not the most abstract description of behavior
- We can describe behavior without assigning operations to particular clock cycles
- High-level synthesis (behavioral synthesis) transforms an unscheduled behavior into a register-transfer behavior

Tasks in Behavioral Synthesis

- **Scheduling** – determines clock cycle on which each operation will occur
- **Allocation** – chooses which function units will execute which operations
- **Data dependencies** describe relationships between operations:
  - x <= a + b; value of x depends on a, b
- High-level synthesis must preserve data dependencies

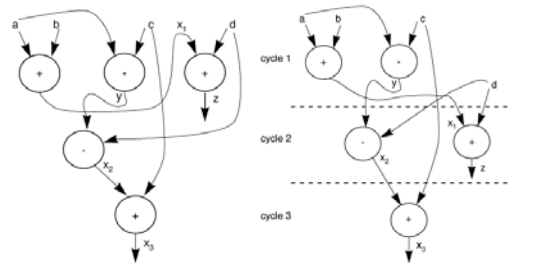
Data Flow Graphs

- Data flow graph (DFG) models data dependencies
- Does not require that operations be performed in a particular order
- Models operations in a basic block of a functional model—no conditionals
- Requires single-assignment form

original code	single-assignment form
x <= a + b;	x1 <= a + b;
y <= a * c;	y <= a * c;
z <= x + d;	z <= x1 + d;
x <= y - d;	x2 <= y - d;
x <= x + c;	x3 <= x2 + c;

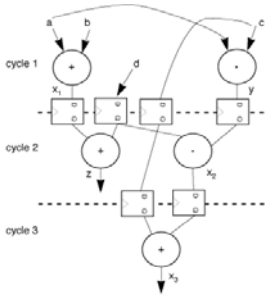
Data Flow Graphs (cont.)

- Data flow forms directed acyclic graph (DAG):



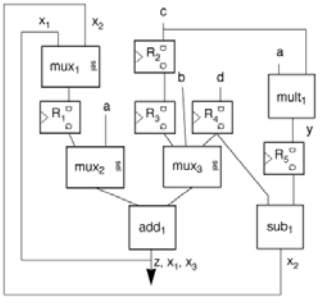
Binding Values to Registers

- Registers fall on clock cycle boundaries

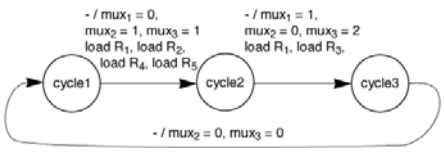


Choosing Functional Units

- Muxes allow for same unit used for different values at different times
- Multiplexer controls which value has access to the unit



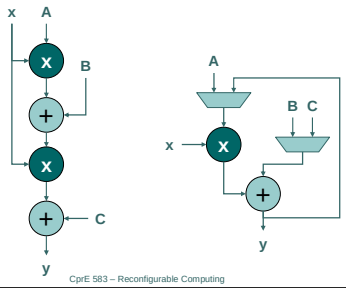
Building the Sequencer



Sequencer requires three states, even with no conditionals

Class Exercise

- How do the quadratic equation designs now compare? (total area usage including control)



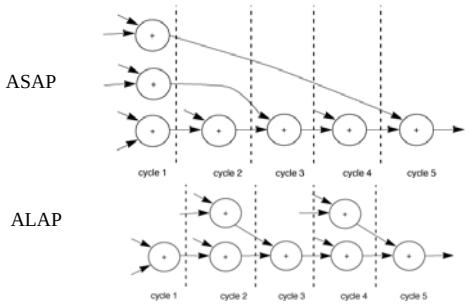
Choices During Behavioral Synthesis

- Scheduling determines number of clock cycles required
- Binding determines area, cycle time
- Area tradeoffs must consider shared function units vs. multiplexers, control
- Delay tradeoffs must consider cycle time vs. number of cycles

Finding Schedules

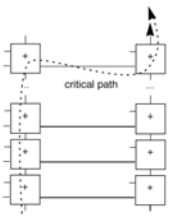
- Two simple schedules:
  - **As-soon-as-possible (ASAP)** schedule puts every operation as early in time as possible
  - **As-late-as-possible (ALAP)** schedule puts every operation as late in schedule as possible
- Many schedules exist between ALAP and ASAP extremes

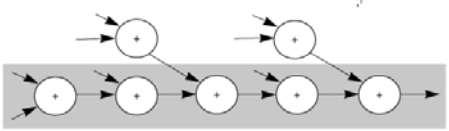
ASAP and ALAP schedules



Critical Path

- Longest path through data flow determines minimum schedule length
- Operator **chaining**:
  - May execute several operations in sequence in one cycle
  - Delay through function units may not be additive, such as through several adders





FPGA Synthesis Summary

- Synthesis is an overloaded term in the FPGA design world
  - Start from VHDL/Verilog/etc. or other system description
  - Generate bitstream, netlist, logic gates
- Relevant steps:
  1. Behavioral code to RTL code (.v)
  2. RTL code to logic netlist (.edn)
  3. Netlist to primitives file (.ngc)
  4. Primitives file to implementation file (.bit)

Placement and Routing

- Two critical phases of layout design:
  - **Placement** of components on the chip
  - **Routing** of wires between components
- Placement and routing interact, but separating layout design into phases helps us understand the problem and find good solutions

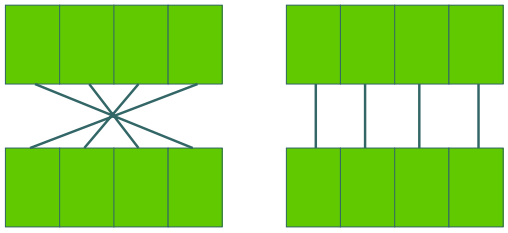
Placement Metrics

- Quality metrics for layout:
  - Area
  - Delay
  - Power
- Area and delay determined partly by wiring
- How do we judge a placement without wiring?
  - Estimate wire length without actually performing routing
- Design time may be important for FPGAs

FPGA Issues

- Often want a fast answer
  - May be willing to accept lower quality result for less place/route time
- May be interested in knowing wirability without needing the final configuration
- Fast placement: constructive placement, iterative improvement through simulated annealing

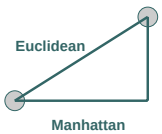
Wire Length as a Quality Metric



Bad Placement **Good Placement**

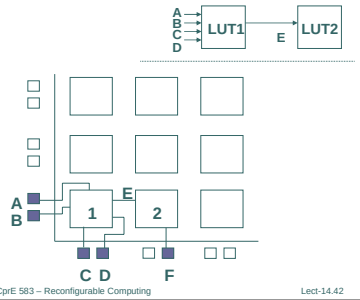
Wire Length Measures

- Estimate wire length by distance between components
- Possible distance measures:
  - Euclidean distance (sqrt($x^2 + y^2$))
  - Manhattan distance (x + y)
- Multi-point nets must be broken up into trees for good estimates



Placement

- Placement has a set of competing goals
- Can't optimize locally and globally simultaneously
- Use heuristic approaches to evaluate quality



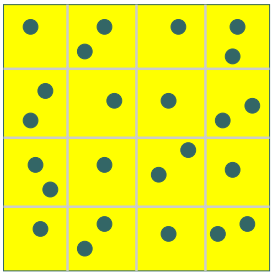
Placement Techniques

- Can construct an initial solution, improve an existing solution
- Pairwise interchange is a simple improvement metric:
  - Interchange a pair, keep the swap if it helps wire length
  - Some heuristic determines which two components to swap

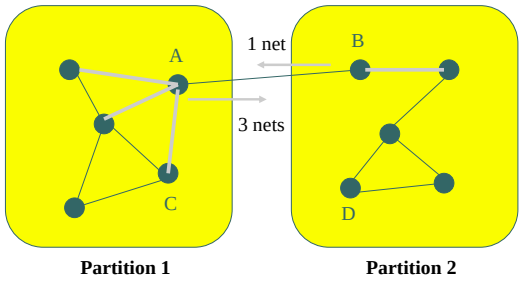
Placement by Partitioning

- Works well for components of fairly uniform size
- Partition netlist to minimize total wire length using **min-cut** criterion
- Partitioning may be interpreted as 1-D or 2-D layout

Recursive Partitioning



Min-Cut Bisecting Partitioning



Min-Cut Partitioning (cont.)

- Swapping A and B:
  - B drags 1 net
  - A drags 3 nets
  - total cut increase: 3 nets
- Conclusion: probably not a good swap, but must be compared with other pairs

Imagine … (Routing)

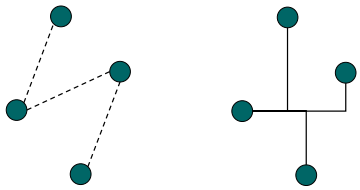
- You have to plan transportation (i.e. roads and highways) for a new city the size of Chicago
- Many dwellings need direct roads that can't be used by anyone else
- You can affect the layout of houses and neighborhoods but the architects and planners will complain
- And … you're told that the time along any path can't be longer than a fixed amount
- What are some of your considerations?

Some Considerations

- How many levels do the roads need to go?
  - Remember: higher is more expensive
- How to avoid congestion?
- What basic structure do I want for my roads?
  - Manhattan?
  - Chicago?
  - Boston?
- Automated routing tools have to solve problems of comparable complexity on every leading-edge chip

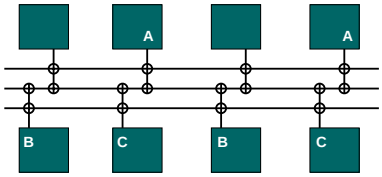
Routing Sub-Problems

- Shortest Path (two-pin nets – $O(N_3)$)
- Steiner Tree (easy for *n*-pin where *n* <= 5; NP-complete in general)
- Compatibility (NP-complete)



Routing Compatibility

- Example: satisfy three simultaneous net connections (A–A, B–B, C–C)
- A–A cannot use middle track
- Greedy approach will not be sufficient

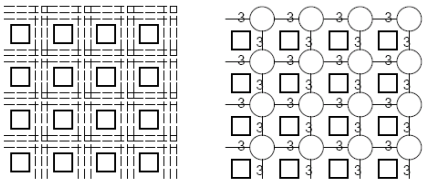


Standard Approach

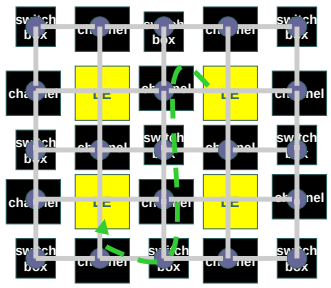
- Major phases in routing:
  - **Global routing** assigns nets to routing areas
  - **Detailed routing** designs the routing areas
  - One phase routers – channel assignment and wire selection happens in one routing pass
- Two phase routers were initially popular
  - Simpler to write and faster to execute
  - More closely models ASIC routing techniques
- One phase routers shown to give MUCH better performance
- Net ordering is a major problem
  - Order in which nets are routed determines quality of result
  - Net ordering is a heuristic

Global routing

- Choose a sequence of channels
  - Not tracks within a channel
- Must take capacity into account
- Channel graph allows path algorithms to be used for global routing



Channel Graph



Placement and Routing Summary

- Placement
  - Placement and clustering of modules critically important for subsequent routing step
  - Often initial placement performed and then iteratively improved
  - Mincut partitioning approaches sometimes used for initial placement
  - Can benefit from simulated annealing approaches, given an accurate cost function
- Routing
  - Routing a difficult problem based on device size, complexity
  - Hard part of routing is the compatibility problem
  - Can be attacked using iterative or simulated annealing approaches