L11/12: Reconfigurable Logic Architectures

Acknowledgements:

Materials in this lecture are courtesy of the following people and used with permission.
- Randy H. Katz (University of California, Berkeley, Department of Electrical Engineering & Computer Science)
- Frank Honore
History of Computational Fabrics

- **Discrete devices**: relays, transistors (1940s-50s)
- **Discrete logic gates**: (1950s-60s)
- **Integrated circuits**: (1960s-70s)
  - e.g. TTL packages: Data Book for 100’s of different parts
- **Gate Arrays (IBM 1970s)**
  - Transistors are pre-placed on the chip & Place and Route software puts the chip together automatically – only program the interconnect (mask programming)
- **Software Based Schemes (1970’s- present)**
  - Run instructions on a general purpose core
- **ASIC Design (1980’s to present)**
  - Turn Verilog directly into layout using a library of standard cells
  - Effective for high-volume and efficient use of silicon area
- **Programmable Logic (1980’s to present)**
  - A chip that be reprogrammed after it has been fabricated
  - Examples: PALs, EPROM, EEPROM, PLDs, FPGAs
  - Excellent support for mapping from Verilog
Reconfigurable Logic

- **Logic blocks**
  - To implement combinational and sequential logic

- **Interconnect**
  - Wires to connect inputs and outputs to logic blocks

- **I/O blocks**
  - Special logic blocks at periphery of device for external connections

**Key questions:**
- How to make logic blocks programmable? *(after chip has been fabbed!)*
- What should the logic granularity be?
- How to make the wires programmable? *(after chip has been fabbed!)*
- Specialized wiring structures for local vs. long distance routes?
- How many wires per logic block?
- Based on the fact that any combinational logic can be realized as a sum-of-products
- PALs feature an array of AND-OR gates with programmable interconnect
Outputs may be registered or combinational, positive or inverted.

Registered output may be fed back to AND array for FSMs, etc.

(Courtesy of Lattice Semiconductor Corporation. Used with permission.)
Anti-Fuse-Based Approach (Actel)

Rows of programmable logic building blocks + rows of interconnect

Anti-fuse Technology: Program Once

Use Anti-fuses to build up long wiring runs from short segments

I/O Buffers, Programming and Test Logic

Logic Module

Wiring Tracks

I/O Buffers, Programming and Test Logic

I/O Buffers, Programming and Test Logic

I/O Buffers, Programming and Test Logic

8 input, single output combinational logic blocks

FFs constructed from discrete cross coupled gates
Combinational block does not have the output FF

Example Gate Mapping

S-R Flip-Flop
Actel Routing & Programming

Programming an Antifuse

(Courtesy of Actel. Used with permission.)
RAM Based Field Programmable Logic - Xilinx

Programmable Interconnect

Configurable Logic Blocks (CLBs)

I/O Blocks (IOBs)

(Courtesy of Xilinx. Used with permission.)
The Xilinx 4000 CLB

Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)

(Courtesy of Xilinx. Used with permission.)
Two 4-input Functions, Registered Output and a Two Input Function

Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)

(Courtesy of Xilinx. Used with permission.)
5-input Function, Combinational Output

Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)

(Courtesy of Xilinx. Used with permission.)
LUT Mapping

- N-LUT direct implementation of a truth table: any function of n-inputs.
- N-LUT requires $2^N$ storage elements (latches)
- N-inputs select one latch location (like a memory)

Why Latches and Not Registers?

Latches set by configuration bitstream

4LUT example

(Courtesy of Xilinx. Used with permission.)
Configuring the CLB as a RAM

Memory is built using Latches not FFs

Read is same a LUT Function!
(Courtesy of Xilinx. Used with permission.)
Xilinx 4000 Interconnect

Single- and Double-Length Lines, with Programmable Switch Matrices (PSMs)

(Courtesy of Xilinx. Used with permission.)
Wires are not ideal!

(Courtesy of Xilinx. Used with permission.)
Add Bells & Whistles

Hard Processor

Gigabit Serial

Multiplier

Programmable Termination

BRAM

Clock Mgmt

I/O

Xilinx 4000 Flexible IOB

Adjust Transition Time

Outputs through FF or bypassed

Adjust the Sampling Edge

(Courtesy of Xilinx. Used with permission.)
The Virtex II CLB (Half Slice Shown)

(Courtesy of Xilinx. Used with permission.)
Adder Implementation

\[ Y = A \oplus B \oplus \text{Cin} \]

Dedicated carry logic

1 half-Slice = 1-bit adder

(Courtesy of Xilinx. Used with permission.)
Carry Chain

1 CLB = 4 Slices = 2, 4-bit adders

64-bit Adder: 16 CLBs

A\[63:0\] + B\[63:0\] → Y\[63:0\]

A\[63:60\] → CLB15 → Y\[63:60\]

A\[7:4\] → CLB1 → Y\[7:4\]

A\[3:0\] → CLB0 → Y\[3:0\]

CLBs must be in same column

(Courtesy of Xilinx. Used with permission.)
Virtex II Features

Double Data Rate registers

Digital Clock Manager

Embedded Multiplier

Block SelectRAM

(Courtesy of Xilinx. Used with permission.)
The Latest Generation: Virtex-II Pro

- FPGA Fabric
- Embedded memories
- Embedded PowerPC
- Hardwired multipliers
- High-speed I/O

(Courtesy of Xilinx. Used with permission.)
Technology Mapping: Schematic/HDL to Physical Logic units

Compile functions into basic LUT-based groups (function of target architecture)

```verbatim
always @(posedge Clock or negedge Reset)
begin
  if (! Reset)
    q <= 0;
  else
    q <= (a & b & c) | (b & d);
end
```
Design Flow – Placement & Route

- **Placement** – assign logic location on a particular device

  ![Placement Diagram]

  - LUT

- **Routing** – iterative process to connect CLB inputs/outputs and IOBs. Optimizes critical path delay – can take hours or days for large, dense designs

  ![Routing Diagram]

  - Iterate placement if timing not met
  - Satisfy timing? → Generate Bitstream to configure device

 Challenge! Cannot use full chip for reasonable speeds (wires are not ideal).

  Typically no more than 50% utilization.
Example: Verilog to FPGA

module adder64 (a, b, sum);
  input [63:0] a, b;
  output [63:0] sum;
  assign sum = a + b;
endmodule

64-bit Adder Example

Virtex II – XC2V2000

(Courtesy of Xilinx. Used with permission.)
How are FPGAs Used?

- **Prototyping**
  - Ensemble of gate arrays used to emulate a circuit to be manufactured
  - Get more/better/faster debugging done than with simulation

- **Reconfigurable hardware**
  - One hardware block used to implement more than one function

- **Special-purpose computation engines**
  - Hardware dedicated to solving one problem (or class of problems)
  - Accelerators attached to general-purpose computers (e.g., in a cell phone!)
FPGA provide a flexible platform for implementing digital computing

A rich set of macros and I/Os supported (multipliers, block RAMS, ROMS, high-speed I/O)

A wide range of applications from prototyping (to validate a design before ASIC mapping) to high-performance spatial computing

Interconnects are a major bottleneck (physical design and locality are important considerations)

“College students will study concurrent programming instead of “C” as their first computing experience.”

-- David B. Parlour, ISSCC 2004 Tutorial