EECS150 - Digital Design
Lecture 5 - Field Programmable Gate Arrays (FPGAs)

February 4, 2002
John Wawrzynek
Outline

• What are FPGAs?
• Why use FPGAs (a short history lesson).
• FPGA variations
• Internal logic blocks.
• Break/Announcements
• Designing with FPGAs.
• Specifics of Xilinx 4000 series.
FPGA Overview

- Basic idea: two-dimensional array of logic blocks and flip-flops with a means for the user to configure:
  1. the interconnection between the logic blocks,
  2. the function of each block.

Simplified version of FPGA internal architecture:
Why FPGAs?

• By the early 1980’s most of the logic circuits in typical systems were absorbed by a handful of standard large scale integrated circuits (LSI).
  – Microprocessors, bus/IO controllers, system timers, ...
• Every system still had the need for random “glue logic” to help connect the large ICs:
  – generating global control signals (for resets etc.)
  – data formatting (serial to parallel, multiplexing, etc.)
• Systems had a few LSI components and lots of small low density SSI (small scale IC) and MSI (medium scale IC) components.
Why FPGAs?

- Custom ICs where sometimes designed to replace the large amount of glue logic:
  - reduced system complexity and manufacturing cost, improved performance.
  - However, custom ICs are relatively very expensive to develop, and delay introduction of product to market (time to market) because of increased design time.

- Note: need to worry about two kinds of costs:
  1. cost of development, sometimes called non-recurring engineering (NRE)
  2. cost of manufacture
     - A tradeoff usually exists between NRE cost and manufacturing costs
Why FPGAs?

• Therefore the custom IC approach was only viable for products with very high volume (where NRE could be amortized), and which were not TTM sensitive.

• FPGAs were introduced as an alternative to custom ICs for implementing glue logic:
  – improved density relative to discrete SSI/MSI components (within around 10x of custom ICs)
  – with the aid of computer aided design (CAD) tools circuits could be implemented in a short amount of time (no physical layout process, no mask making, no IC manufacturing)
    • lowers NREs
    • shortens TTM

• Because of Moore’s law the density (gates/area) of FPGAs continued to grow through the 80’s and 90’s to the point where major data processing functions can be implemented on a single FPGA.
Why FPGAs?

- FPGAs continue to compete with custom ICs for special processing functions (and glue logic) but now also compete with microprocessors in dedicated and embedded applications.
  - Performance advantage over microprocessors because circuits can be customized for the task at hand. Microprocessors must provide special functions in software (many cycles).

- Summary:

<table>
<thead>
<tr>
<th>performance</th>
<th>NREs</th>
<th>Unit cost</th>
<th>TTM</th>
</tr>
</thead>
<tbody>
<tr>
<td>ASIC</td>
<td>ASIC</td>
<td>FPGA</td>
<td>ASIC</td>
</tr>
<tr>
<td>FPGA</td>
<td>FPGA</td>
<td>MICRO</td>
<td>FPGA</td>
</tr>
<tr>
<td>MICRO</td>
<td>MICRO</td>
<td>ASIC</td>
<td>MICRO</td>
</tr>
</tbody>
</table>

ASIC = custom IC, MICRO = microprocessor
FPGA Variations

- Families of FPGA’s differ in:
  - physical means of implementing user programmability,
  - arrangement of interconnection wires, and
  - the basic functionality of the logic blocks.

- Most significant difference is in the method for providing flexible blocks and connections:

  - Anti-fuse based (ex: Actel)

  + Non-volatile, relatively small
  - fixed (non-reprogrammable)
User Programmability

- Latch-based (Xilinx, Altera, …)

  - reconfigurable
  - volatile
  - relatively large.

- Latches are used to:
  1. make or break cross-point connections in the interconnect
  2. define the function of the logic blocks
  3. set user options:
     - within the logic blocks
     - in the input/output blocks
     - global reset/clock
  “Configuration bit stream” can be loaded under user control:
     - All latches are strung together in a shift chain:
• **4-input look up table (LUT)**
  – implements combinational logic functions

• **Register**
  – optionally stores output of LUT
4-LUT Implementation

- n-bit LUT is implemented as a $2^n$ x 1 memory:
  - inputs choose one of $2^n$ memory locations.
  - memory locations (latches) are normally loaded with values from user’s configuration bit stream.
  - Inputs to mux control are the CLB inputs.

- Result is a general purpose “logic gate”.
  - n-LUT can implement *any* function of n inputs!

Latches programmed as part of configuration bit-stream
LUT as general logic gate

- An n-lut as a direct implementation of a function **truth-table**.
- Each latch location holds the value of the function corresponding to one input combination.

**Example: 4-lut**

<table>
<thead>
<tr>
<th>INPUTS</th>
<th>F(0,0,0,0)</th>
<th>F(0,0,0,1)</th>
<th>F(0,0,1,0)</th>
<th>F(0,0,1,1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000</td>
<td>store in 1st latch</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0001</td>
<td></td>
<td>store in 2nd latch</td>
<td></td>
<td></td>
</tr>
<tr>
<td>0010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0111</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1001</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1010</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1011</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1100</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1101</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1110</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1111</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Example: 2-lut**

<table>
<thead>
<tr>
<th>INPUTS</th>
<th>AND</th>
<th>OR</th>
</tr>
</thead>
<tbody>
<tr>
<td>00</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>01</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Implements *any* function of 2 inputs.

How many of these are there?

How many functions of n inputs?
Announcements

• Quiz results
• Administrative Q&A.
• New reading posted:
  – large section of Xilinx 4000 databook
  – All of chapter 2 in Mano
FPGA Generic Design Flow

- **Design Entry:**
  - Create your design files using:
    - schematic editor or
    - hardware description language (Verilog, VHDL)
- **Design “implementation” on FPGA:**
  - *Partition, place, and route* to create bit-stream file
- **Design verification:**
  - Use Simulator to check function,
  - other software determines max clock frequency.
  - Load onto FPGA device (cable connects PC to development board)
    - check operation at full speed in real environment.
Example Partition, Placement, and Route

- Idealized FPGA structure:
- Example Circuit:
  - collection of gates and flip-flops

Circuit combinational logic must be “covered” by 4-input 1-output “gates”.
Flip-flops from circuit must map to FPGA flip-flops.
(Best to preserve “closeness” to CL to minimize wiring.)
Placement in general attempts to minimize wiring.
Xilinx FPGAs (4000 Series)

Table 1: XC4000E and XC4000X Series Field Programmable Gate Arrays

<table>
<thead>
<tr>
<th>Device</th>
<th>Logic Cells</th>
<th>Max Logic Gates (No RAM)</th>
<th>Max. RAM Bits (No Logic)</th>
<th>Typical Gate Range (Logic and RAM)*</th>
<th>CLB Matrix</th>
<th>Total CLBs</th>
<th>Number of Flip-Flops</th>
<th>Max. User I/O</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC4002XL</td>
<td>152</td>
<td>1,600</td>
<td>2,048</td>
<td>1,000 - 3,000</td>
<td>8 x 8</td>
<td>64</td>
<td>256</td>
<td>64</td>
</tr>
<tr>
<td>XC4003E</td>
<td>238</td>
<td>3,000</td>
<td>3,200</td>
<td>2,000 - 5,000</td>
<td>10 x 10</td>
<td>100</td>
<td>360</td>
<td>80</td>
</tr>
<tr>
<td>XC4005E/XL</td>
<td>466</td>
<td>5,000</td>
<td>6,272</td>
<td>3,000 - 9,000</td>
<td>14 x 14</td>
<td>196</td>
<td>616</td>
<td>112</td>
</tr>
<tr>
<td>XC4006E</td>
<td>608</td>
<td>6,000</td>
<td>8,192</td>
<td>4,000 - 12,000</td>
<td>16 x 16</td>
<td>256</td>
<td>768</td>
<td>128</td>
</tr>
<tr>
<td>XC4008E</td>
<td>770</td>
<td>8,000</td>
<td>10,368</td>
<td>6,000 - 15,000</td>
<td>18 x 18</td>
<td>324</td>
<td>936</td>
<td>144</td>
</tr>
<tr>
<td>XC4010E/XL</td>
<td>950</td>
<td>10,000</td>
<td>12,800</td>
<td>7,000 - 20,000</td>
<td>20 x 20</td>
<td>400</td>
<td>1,120</td>
<td>160</td>
</tr>
<tr>
<td>XC4013E/XL</td>
<td>1368</td>
<td>13,000</td>
<td>18,432</td>
<td>10,000 - 30,000</td>
<td>24 x 24</td>
<td>576</td>
<td>1,536</td>
<td>192</td>
</tr>
<tr>
<td>XC4020E/XL</td>
<td>1962</td>
<td>20,000</td>
<td>25,088</td>
<td>13,000 - 40,000</td>
<td>28 x 28</td>
<td>784</td>
<td>2,016</td>
<td>224</td>
</tr>
<tr>
<td>XC4025E</td>
<td>2432</td>
<td>25,000</td>
<td>32,768</td>
<td>15,000 - 45,000</td>
<td>32 x 32</td>
<td>1,024</td>
<td>2,560</td>
<td>256</td>
</tr>
<tr>
<td>XC4028EX/XL</td>
<td>2432</td>
<td>28,000</td>
<td>32,768</td>
<td>18,000 - 50,000</td>
<td>32 x 32</td>
<td>1,024</td>
<td>2,560</td>
<td>256</td>
</tr>
<tr>
<td>XC4036EX/XL</td>
<td>3078</td>
<td>36,000</td>
<td>41,472</td>
<td>22,000 - 65,000</td>
<td>36 x 36</td>
<td>1,296</td>
<td>3,168</td>
<td>288</td>
</tr>
<tr>
<td>XC4044XL</td>
<td>3800</td>
<td>44,000</td>
<td>51,200</td>
<td>27,000 - 80,000</td>
<td>40 x 40</td>
<td>1,600</td>
<td>3,840</td>
<td>320</td>
</tr>
<tr>
<td>XC4052XL</td>
<td>4598</td>
<td>52,000</td>
<td>61,952</td>
<td>33,000 - 100,000</td>
<td>44 x 44</td>
<td>1,936</td>
<td>4,576</td>
<td>352</td>
</tr>
<tr>
<td>XC4062XL</td>
<td>5472</td>
<td>62,000</td>
<td>73,728</td>
<td>40,000 - 130,000</td>
<td>48 x 48</td>
<td>2,304</td>
<td>5,376</td>
<td>384</td>
</tr>
<tr>
<td>XC4085XL</td>
<td>7448</td>
<td>85,000</td>
<td>100,352</td>
<td>55,000 - 180,000</td>
<td>56 x 56</td>
<td>3,136</td>
<td>7,168</td>
<td>448</td>
</tr>
</tbody>
</table>

* Max values of Typical Gate Range include 20-30% of CLBs used as RAM.
Xilinx FPGAs (CLB detail)

Figure 1: Simplified Block Diagram of XC4000 Series CLB (RAM and Carry Logic functions not shown)

Spring 2002 EECS150 - Lec05-FPGA
Xilinx FPGAs (IOB detail)

Figure 15: Simplified Block Diagram of XC4000E IOB
Xilinx FPGAs (interconnect detail)
Xilinx 4000 series FPGAs

- How they differ from idealized array:
  - In addition to their use as general logic “gates”, LUTs can alternatively be used as general purpose RAM.
    - Each 4-lut can become a 16x1-bit RAM array.
  - Special circuitry to speed up “ripple carry” in adders and counters.
    - Therefore adders in the “Xilinx Unified Library” operate much faster than adders built from gates and luts alone.
  - Many more wires, including tri-state capabilities.