Outline
- What are FPGAs?
- Why use FPGAs (a short history lesson).
- FPGA variations
- Internal logic blocks.
- Break/Announcements
- Designing with FPGAs.
- Specifics of Xilinx Virtex-E 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.

Why FPGAs?
- By the early 1980’s most of the logic circuits in typical systems where 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 time to market 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:

  + Latch-based (Xilinx, Altera, …)
    - reconfigurable
    - volatile
    - relatively large.

Idealized FPGA Logic Block

- 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 $2^n \times 1$ 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!

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

Implement any function of 2 inputs.
How many of these are there?
How many functions of n inputs?
Announcements

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:
  - collection of gates and flip-flops
- Example Circuit:
  - 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.
Virtex-E Configurable Logic Block (CLB)

2 “logic slices”

Details of Virtex-E Slice

Block SelectRam

Dual Port Ram

Xilinx FPGAs (interconnect detail)
Connections to Dedicated Horizontal Lines

Global Clock Distribution

Virtex-E Family of Parts

<table>
<thead>
<tr>
<th>Device</th>
<th>System Gates</th>
<th>Logic Gates</th>
<th>CLB Array</th>
<th>Logic Cells</th>
<th>Differential I/O Pairs</th>
<th>User I/O</th>
<th>BlockRAM Bits</th>
<th>Distributed RAM Bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>XC95100E</td>
<td>1,500</td>
<td>220</td>
<td>10 x 20</td>
<td>100</td>
<td>2</td>
<td>2</td>
<td>20,000</td>
<td>20,000</td>
</tr>
<tr>
<td>XC95300E</td>
<td>3,000</td>
<td>440</td>
<td>10 x 20</td>
<td>200</td>
<td>4</td>
<td>4</td>
<td>40,000</td>
<td>40,000</td>
</tr>
<tr>
<td>XC95500E</td>
<td>6,000</td>
<td>880</td>
<td>10 x 20</td>
<td>400</td>
<td>8</td>
<td>8</td>
<td>80,000</td>
<td>80,000</td>
</tr>
</tbody>
</table>
Xilinx 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 assembled by the CAD tools operate much faster than adders built from gates and luts alone.
  – Many more wires, including tri-state capabilities.