EECS150 - Digital Design
Lecture 3 - Field Programmable Gate Arrays (FPGAs)

January 25, 2005
John Wawrzynek

Outline

• What are FPGAs?
• Why use FPGAs (a short history lesson).
• FPGA variations
• Internal logic blocks.
• 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.

Simplified version of FPGA internal architecture:

Die Photos: Virtex FPGA vs. Pentium IV

• FPGA Vertex chip looks remarkably structured
  – Very dense, very regular structure

• “Full-Custom” Pentium chip somewhat more random in structure
  – Large on-chip memories (caches) are visible
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 (used as “glue logic”).

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 (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), relative to ASICs.
    • 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

• Newer FPGAs even combine microprocessor cores, special multiplier circuits, memory blocks, and configurable logic on a single chip.
**FPGA as CSOC**

Xilinx Virtex-II Pro 100+ (year 2003)
- ~100K logic blocks, each with 4-LUT and Flip-flop (8 Million “system” gates)
- 1 MBytes SRAM bits
- 444 18x18bit dedicated multipliers
- 20 10-Gbit/s serial communication links
- ~1000 user I/Os (most with LVDS 600 Mb/s signaling)
- 2 embedded “hard” PowerPC cores

Xilinx Technology Drives Performance in BMW Williams Formula One Race Car at 2003 Grand Prix

**Xilinx Technology Drives Performance in BMW Williams Formula One Race Car at 2003 Grand Prix**

Sep 26, 2003 18:08 ET

Xilinx-Enabled WilliamsF1 BMW FW25 Challenges Ferrari for the Win

FPGA also get used in many (not as interesting) products:
- network routers,
- set-top boxes,
- printers, etc.

Xilinx-enabled WilliamsF1 (PRNewsFoto)

SAN JOSE, Calif., Sept. 26 /PRNewswire/ -- At the Indianapolis 2003 Grand Prix today, Xilinx, Inc. (NASDAQ:XLNX) programmable chips will play a mission-critical role in the performance of the BMW WilliamsF1 Team. The BMW WilliamsF1 Team is the leading contender in the 2003 Formula One Series, currently heading the FIA Formula One Constructors' Championship. After a closely fought race in Monza, Italy, between Ferrari, BMW Williams and McLaren, the outcome of today's U.S. Grand Prix race could hold the key to the entire 2003 FIA Formula One Championship. The season concludes with a final race in Suzuka, Japan on October 12.

The teams and drivers who remain in contention are acutely conscious that wins and losses will come at the margins of performance, and no one can afford anything less than a faultless race. The BMW WilliamsF1 Team selected high performance programmable chips from Xilinx as the leading technology in the Vehicle Control Module (VCM) of the team's 2003 Challenger -- the WilliamsF1 BMW FW25.

The Xilinx-enabled VCM is used to control essential vehicle components such as the gearbox, differential, traction control, launch control and telemetry. With leading edge technology -- greater on-chip functionality and lower power consumption, Xilinx FPGAs have ensured that the VCM unit is reduced in size and weight, ultimately enhancing the performance of the car.
Berkeley Research Project

- FPGA based computing machine provides computational performance similar to supercomputers.
- Provides all computing needs for radio astronomy telescope array.
- Also used to accelerate simulation/emulation of other hardware systems.
- Module from “BEE2”
- Scalable from 5-1000 FPGAs.

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, …)
- 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

Idealized FPGA Logic Block

- 4-input look up table (LUT)
  - implements combinational logic functions
- Register
  - optionally stores output of LUT
Annoucements

• Homework #1 due Friday (associated reading is linked).
• Quiz #1 Thursday during lecture.
• Please do the reading (the earlier the better).
• Homework is an important part of the class:
  – It goes beyond what is covered in class
  – Work on it seriously
  – Discussion is a good place to get hints about homework.
• Unlike some of our lower division classes we will not necessarily tell you everything you need to know. Some of it will come from readings and homework.

Background for Next Slide

• A MUX or multiplexor is a combinational logic circuit that chooses between $2^N$ inputs under the control of N control signals.

• A latch is a 1-bit memory (similar to a flip-flop).
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!

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>store in 3rd latch</td>
<td></td>
</tr>
<tr>
<td>0011</td>
<td></td>
<td></td>
<td></td>
<td>store in 4th latch</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?
FPGA Generic Design Flow

- Design Entry:
  - Create your design files using:
    - schematic editor or
    - hardware description language (Verilog, VHDL) and “logic synthesis”
- 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:
  - 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 Virtex-E Floorplan

Virtex-E Configurable Logic Block (CLB)

1 CLB = 2 "logic slices"

1 slice: 2 LUTs, 2 FFs
Details of Virtex-E Slice

Xilinx FPGAs (interconnect detail)

lots of wires...

lots of delay
Virtex-E Input/Output block (IOB) detail

Virtex-E Family of Parts

Table 1: Virtex-E Field-Programmable Gate Array Family Members

<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>XCV50E</td>
<td>71,693</td>
<td>20,736</td>
<td>16 x 24</td>
<td>1,728</td>
<td>83</td>
<td>176</td>
<td>65,536</td>
<td>24,576</td>
</tr>
<tr>
<td>XCV100E</td>
<td>128,236</td>
<td>32,400</td>
<td>20 x 30</td>
<td>2,700</td>
<td>83</td>
<td>196</td>
<td>81,920</td>
<td>38,400</td>
</tr>
<tr>
<td>XCV200E</td>
<td>306,393</td>
<td>63,504</td>
<td>28 x 42</td>
<td>5,292</td>
<td>119</td>
<td>284</td>
<td>114,688</td>
<td>75,264</td>
</tr>
<tr>
<td>XCV300E</td>
<td>411,955</td>
<td>82,944</td>
<td>32 x 48</td>
<td>6,912</td>
<td>137</td>
<td>316</td>
<td>131,072</td>
<td>98,304</td>
</tr>
<tr>
<td>XCV400E</td>
<td>569,952</td>
<td>129,600</td>
<td>40 x 60</td>
<td>10,800</td>
<td>183</td>
<td>404</td>
<td>163,840</td>
<td>153,600</td>
</tr>
<tr>
<td>XCV600E</td>
<td>985,882</td>
<td>186,624</td>
<td>48 x 72</td>
<td>15,552</td>
<td>247</td>
<td>512</td>
<td>294,912</td>
<td>221,184</td>
</tr>
<tr>
<td>XCV1000E</td>
<td>1,569,178</td>
<td>331,776</td>
<td>64 x 96</td>
<td>27,648</td>
<td>281</td>
<td>660</td>
<td>393,216</td>
<td>393,216</td>
</tr>
<tr>
<td>XCV1600E</td>
<td>2,188,742</td>
<td>419,904</td>
<td>72 x 108</td>
<td>34,992</td>
<td>344</td>
<td>724</td>
<td>699,824</td>
<td>497,664</td>
</tr>
<tr>
<td>XCV2000E</td>
<td>2,541,952</td>
<td>518,400</td>
<td>80 x 120</td>
<td>43,200</td>
<td>344</td>
<td>804</td>
<td>855,364</td>
<td>614,400</td>
</tr>
<tr>
<td>XCV2600E</td>
<td>3,263,755</td>
<td>685,584</td>
<td>92 x 138</td>
<td>57,132</td>
<td>344</td>
<td>804</td>
<td>753,664</td>
<td>812,544</td>
</tr>
<tr>
<td>XCV3200E</td>
<td>4,074,387</td>
<td>876,096</td>
<td>104 x 156</td>
<td>73,008</td>
<td>344</td>
<td>804</td>
<td>851,968</td>
<td>1,038,336</td>
</tr>
</tbody>
</table>

Spring 2005
EECS150 - Lec03-FPGA
Page 25

Spring 2005
EECS150 - Lec03-FPGA
Page 26
Our FPGAs

• How they differ from idealized array:
  – In addition to their use as general logic “gates”, LUTs can alternatively be used as general purpose “random access” memory (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.

Summary

• Logic design process influenced by available technology AND economic drivers
  – Volume, Time to Market, Costs, Power
• FPGA offer a valuable new sweet spot
  – Low TTM, medium cost, tremendous flexibility (during and after design is done - field upgrades are possible).
• Fundamentally tied to powerful CAD tools
• Build everything (simple or complex) from one set of building blocks
  – LUTs + FF + routing + storage + IOs
• FPGA = Field programmable gate array.
  But where are the gates?!