Introduction and the architecture of FPGA
Evolution of Integrated Circuits

- < 1960 individual transistors
- 1960s – 1970s:
  - SSI, MSI, LSI (10,000 transistors)
- 1980s:
  - Programmable Logic Devices (PLAs, PALs)
  - 16-bit, 32-bit processors (> 1,000,000 transistors)
- 1990s:
  - Full custom chips
  - Gate arrays (semi-custom chips)
  - Field-Programmable Gate Arrays
  - > 100,000,000 transistors
- 2010: > 1,000,000,000 transistors
  - Intel Quad-core Itanium (2 billion)
Intel i4004 (1971)
2,300 transistors

Intel i486 (1988)
1.2 million transistors

P4 (2000)
55 million transistors

Core2 Duo (2006)
291 million transistors
Stratix IV GX FPGA

2.5 billion transistors
Why Teach Students about FPGAs?

- Field-Programmable Gate Arrays are programmable hardware chips
  - Can be used to implement any digital hardware circuit

- Digital hardware is found is almost all modern products
  - Consumer produces, like audio and video players
  - Telephone and television equipment
  - Cars, airplanes, trains
  - Medical equipment, and industrial control

- FPGAs are … everywhere
Industries that use FPGAs

- Consumer
- Automotive
- Test, Measurement & Medical
- Communications Broadcast
- Military & Industrial
- Computer & Storage
Traditional FPGA Applications
Consumer Applications

Set-Top Boxes

Touch Panels

DVD Players

Printers

Handheld Media Players

LCD, Plasma Displays/TVs

Consumer Music

Camcorders
What are FPGAs and why should we use them?
FPGA

- **Field Programmable** Gate Array
  - A large set of *programmable* logic elements
  - Connected with *programmable switches*

- FPGA: re-programmable hardware
FPGA Architecture: Logic Element

- Lookup table (LUT) implements any 4-input logic function

- Actual LE is significantly more complex
FPGA Architecture

- Logic array block (LAB)
- Logic element
- Programmable switch
- Flip-flop
- IO cells
- Select

© 2011 Altera Corporation — Confidential
Stratix IV GX (more than 10 million gates)

Adaptive Logic Modules

RAM Blocks (M9K & M144K)
Digital Signal Processing
General I/O

Programmable I/O
Clock Management

Delay Locked Loops (DLLs)

Phase Locked Loops (PLLs)
Serial Interfaces

High Speed Serial Interfaces
Introducing Arria II GX FPGAs

- High functionality
  - Optimized logic, memory, and digital signal processing (DSP) ratios for 3-Gbps applications
  - Up to 16 transceivers @ 3.75 Gbps

- Lowest power 3-Gbps FPGA
  - 40-nm process with 0.9V core voltage
  - <100 mW per transceiver channel
  - Integrated power optimization tool

- Design with ease
  - Built-in PCI Express hard IP
  - Single design environment
  - Jump-start design with protocol IP packs, reference designs and development kits

Lowest power, cost optimized for 3 Gbps
# Arria II GX Family

<table>
<thead>
<tr>
<th>Device</th>
<th>Equiv. LEs</th>
<th>Maximum Transceiver Channels&lt;sup&gt;(1)&lt;/sup&gt;</th>
<th>PCI Express Hard IP Blocks</th>
<th>Total Memory MBits</th>
<th>18 X 18 Multipliers</th>
<th>Max. LVDS I/O&lt;sup&gt;(1)(2)&lt;/sup&gt;</th>
<th>Phase-locked loops (PLLs)&lt;sup&gt;(1)&lt;/sup&gt;</th>
</tr>
</thead>
<tbody>
<tr>
<td>EP2AGX20</td>
<td>16K</td>
<td>4</td>
<td>1</td>
<td>0.98</td>
<td>56</td>
<td>48</td>
<td>4</td>
</tr>
<tr>
<td>EP2AGX30</td>
<td>27K</td>
<td>4</td>
<td>1</td>
<td>1.6</td>
<td>144</td>
<td>48</td>
<td>4</td>
</tr>
<tr>
<td>EP2AGX45</td>
<td>45K</td>
<td>8</td>
<td>1</td>
<td>3.4</td>
<td>228</td>
<td>56</td>
<td>4</td>
</tr>
<tr>
<td>EP2AGX65</td>
<td>63K</td>
<td>8</td>
<td>1</td>
<td>5.2</td>
<td>312</td>
<td>56</td>
<td>4</td>
</tr>
<tr>
<td>EP2AGX95</td>
<td>94K</td>
<td>12</td>
<td>1</td>
<td>6.7</td>
<td>448</td>
<td>64</td>
<td>6</td>
</tr>
<tr>
<td>EP2AGX125</td>
<td>124K</td>
<td>12</td>
<td>1</td>
<td>8.1</td>
<td>576</td>
<td>64</td>
<td>6</td>
</tr>
<tr>
<td>EP2AGX190</td>
<td>190K</td>
<td>16</td>
<td>1</td>
<td>9.9</td>
<td>656</td>
<td>96</td>
<td>6</td>
</tr>
<tr>
<td>EP2AGX260</td>
<td>256K</td>
<td>16</td>
<td>1</td>
<td>11.8</td>
<td>736</td>
<td>96</td>
<td>6</td>
</tr>
</tbody>
</table>

**Notes:**
1) Number of transceiver channels and LVDS I/O determined by packaging.
2) Maximum LVDS I/O denotes channels with built-in SERDES (up to 1 Gbps). See Handbook for details.
Arria II GX Transceiver Block Architecture

- Up to 4 transceiver blocks
- Left side of device only
- Dynamic reconfiguration support

CCU = Clock Control Unit; CMU = Clock Management Unit

Arria II GX device with 8 transceiver channels (2 transceiver blocks)
Why use FPGAs?

Problem:
- Create an application to process a lot of data quickly
- How?

Alternatives:
- Processor?
  - Easy to write code, low performance, power hungry
- Gate Array (ASIC)?
  - Very high performance, low power, very hard to design, expensive to manufacture
- Field-Programmable Gate Array?
  - no manufacturing needed (just program), easier to design for than ASIC, high performance, lower power than a processor
Increasing Chip Development Cost

Total Development Cost (M$)

- 0.18 µm
- 0.15 µm
- 0.13 µm
- 90 nm
- 65 nm
- 45 nm

- Design / Verification & Layout
- Software
- Test & Product Engineering
- Masks & Wafers

© 2011 Altera Corporation—Confidential

23
Getting a Product Out

Quartus II CAD Tools

Design

Test

Arria II GX

Yes

OK?

No
Quartus II Design Flow

Design Entry

Timing Constraints

Synthesis

Placement and Routing

Timing and Power Analyzer

Timing, Area, Power Optimization

Optimized Design ✔️
Design Flow: Quartus II

Synthesis
3rd Party or Altera

Placement & Routing

Physical Synthesis

Timing & Power Analysis

Assembler

Report

Verilog, VHDL

// Begin: Write Control
always @ (posedge wrbusy_int)
begin
write0 <= 1'b1;
write1 <= 1'b0;
writex <= 1'b0;
end
always @ (negedge wrbusy_int)
begin
write0 <= 1'b0;
end
always @ (posedge write0_done)
begin
write1 <= 1'b1;
// End: Write Control
always @ (posedge wrbusy_int)
begin
write0 <= 1'b1;
write1 <= 1'b0;
writex <= 1'b0;
end
always @ (negedge wrbusy_int)
begin
write0 <= 1'b0;
end
always @ (posedge write0_done)
begin
write1 <= 1'b1;

Teaching with Altera CAD Tools and Educational Platform
Altera Quartus II CAD Tools

Schematic, Verilog HDL, VHDL

Host Computer

USB Cable

CAD tools
How Students can use an FPGA

- Create design
  - Schematic, or Hardware Description Language (Verilog HDL, VHDL)
- Compile with FPGA CAD tools
- Simulate, debug
- Program into an FPGA lab board
- If a processor is included, write application code
- Test, debug
Teaching Materials Needed

- FPGA lab board
  - with all the right pedagogical features

- CAD tools – Quartus II
  - with tutorials to learn how to use the software

- Lab experiments
  - that fit into a modern curriculum
  - that cover the fundamentals and are fun/challenging/interesting
Developing Teaching Materials

Materials that help with teaching Digital Logic and Computer Organization

What is our approach?
- Examine the material covered in a course
- Create exercises that enhance student learning
  - Creative and Interesting examples
  - Progressive learning
    - Build knowledge base with easy examples first
    - Extend examples to allow students to enhance their understanding

Materials are designed for our Educational Boards (DE2/DE1)
DE2

- Ideal for undergraduate courses
  - Adopted in many Universities around the World

- Cyclone II 35/70 FPGA
  - 35k or 70k LUTs
  - More space than undergraduate students need for their projects

- User interface
  - Switches
  - 7-segment displays
  - LEDs
  - 16 x 2 Character LCD Display

- Advanced peripherals
  - Memory (SRAM, SDRAM, Flash)
  - USB
  - Video/Audio in/out
  - Ethernet
  - SD Card slot
Great for undergraduate courses
- For student budget, so they can buy one for personal use

Cyclone II 20 FPGA
- 20k LUTs

User interface
- Switches
- 7-segment displays
- LEDs

Advanced peripherals
- memory
- USB
- Video/Audio in/out
- Ethernet
- SD Card slot
Digital Logic Exercises
Outline

- Creating projects in Quartus II
- Targeting a project for a DE2 Board
- Downloading a circuit onto a DE2 board
- Compiling and debugging
Step 1: Start Quartus II
Step 2: Create a New Project

- Click File Menu
- Select New Project Wizard
- This will open a new window where project information can be specified
Project Name and Directory

New Project Wizard: Directory, Name, Top-Level Entity [page 1 of 5]

What is the working directory for this project?
D:\Course\Digital Logic\simple

What is the name of this project?
simple

What is the name of the top-level design entity for this project? This name is case sensitive and must exactly match the entity name in the design file.
simple

Use Existing Project Settings...
Add Source Files to Project

**New Project Wizard: Add Files [page 2 of 5]**

Select the design files you want to include in the project. Click Add All to add all design files in the project directory to the project. Note: you can always add design files to the project later.

<table>
<thead>
<tr>
<th>File name</th>
<th>Type</th>
<th>Library</th>
<th>Design entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>simple.v</td>
<td>Verilog HDL File</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Specify the path names of any non-default libraries.  

[User Libraries... button]
Specify FPGA Device

- Select the FPGA device on the board
  - Cyclone II Family
  - For DE2 – EP2C35F672C6
  - For DE1 – EP2C20F484C7
Additional EDA Tools

- Specify Tools, in addition to Quartus II, that you will use.
- These are unnecessary for small student designs:
  - Leave all entries as <None>
  - Press Next
New Project Summary

When you click Finish, the project will be created with the following settings:

- **Project directory:** D:/Course/Digital Logic/simple/
- **Project name:** simple
- **Top level design entity:** simple
- **Number of files added:** 1
- **Number of user libraries added:** 0
- **Device assignments:**
  - **Family name:** Cyclone II
  - **Device:** EP2C20F484C7
- **EDA tools:**
  - **Design entry/synthesis:** <None>
  - **Simulator:** <None>
  - **Timing analysis:** <None>
- **Operating conditions:**
  - **Core voltage:** 1.2V
  - **Junction temperature range:** 0-85 °C
Simple Project

8-bit Shift Register

Input
Reset_N
Clock

SW[0]  KEY[0]  KEY[1]

LEDG[7]  ...  LEDG[0]
Step 3: Open Source File
Step 4: Assign Pins to connect switches/lights to inputs and outputs of your circuit

- Click **Assignments**, then **Import Assignments**...

- Import file
  - DE1_pin_assignments.csv

- Imports locations for predefined port names, such as SW, LEDG, KEY, and others
  - Can be done manually for custom port names
Step 5: Compile Design

- **Verilog, VHDL**
- **Synthesis**
- **Placement & Routing**
- **Timing & Power Analysis**
- **Assembler**
- **Report**
Step 6: Examine Compilation Report
Step 7: Program the DE1 Board
Step 8: See your design work on the board

- Reset the shift register using KEY[1]
- SW[0] is the input to the shift register
- Press the KEY[0] to clock the circuit

![Diagram showing 8-bit Shift Register with inputs SW[0], KEY[1], KEY[0], and output Green LEDs]
Next Example

- Open the **Digital Logic** Directory
- Go into **stopper** subdirectory
- Double-click on stopper.qpf to open an existing project
Example 2: Stopper

- Shift the contents of a register once every second
  - The circuit is clocked using a 50MHz clock

- Press KEY[0] to start or stop the shift register
  - FSM examines if the key was pressed

- Purpose:
  - Look at FSM implementation in Quartus II
  - Finite State Machine Viewer
Circuit Diagram

- **KEY[0]**
- **FSM**
- **Clock Divider**
  - Fast
  - Slow
- **10-bit Shift Register**
  - enable
  - Clock
- **Red LEDs**
Step 1: Open Stopper Project

```
module stopper(CLOCK_50, KEY, LEDR);
  input  CLOCK_50;
  input [1:0]  KEY;
  output [9:0]  LEDR;
  reg [9:0]  shift_reg;
  reg [20:0]  counter;
  reg  go;
  wire  reset_n, enable, toggle_go;

  /* Glue logic */
  assign reset_n = KEY[1];
  assign enable = 6'counter;

  /* Control FSM */
  pulse_FSM FSM(.Clock(CLOCK_50), .Resetn(reset_n), .i_pulse(~KEY[0]),
                  .o_pulse(toogle_go));

  always @(posedge CLOCK_50 or negedge reset_n)
  begin
    if (!reset_n)
      begin
        go <= 1'bg;
        counter <= 21'd0;
        shift_reg <= 10'd1;
      end
    else
      begin
        counter <= counter + 1'bi;
      end
  end
```

© 2011 Altera Corporation — Confidential
Step 2: Compile and Program

- Compile the design
- Program the design onto the DE2 board

How does it work?
- Press KEY[0] to start/stop the circuit
- Press KEY[1] to reset the circuit
Step 3: Examine the FSM Source Code
Step 4: FSM Viewer

- Open the FSM Viewer
  - Click Tools
  - Expand Netlist Viewers
  - Click State Machine Viewer
Examine State Machine
Next Example

- Open the Digital Logic Directory
- Go into seg_shift subdirectory
- Double-click on seg_shift.qpf to open an existing project
Example 3: Segment Shifter

- Combine a few shift registers and a few instances of the FSM from Example 2
- Circuit will scroll lights left/right, both on lights and on 7-Segment displays

Purpose:
- Look at building larger circuits
- RTL Viewer
Step 1: Open seg_shift Project
Step 2: Compile and Program

- Compile the design
- Program the design onto the DE2 board

How does it work?
- Press KEY[0] to reset the circuit
- SW[0] to shift right, SW[1] to shift left
  - If SW[2] is high 1’s are shifted into the register
  - If SW[3] is high 0’s are shifted into the register
- Press KEY[1] to speed up the circuit and KEY[2] to slow it down
Step 3: See the Circuit in RTL Viewer

- Start the RTL Viewer
  - Click Tools
  - Expand the Netlist Viewers list
  - Click RTL Viewer
Examine the Circuit
SignalTap II Embedded Logic Analyzer

- A soft logic analyzer
  - Instantiate as a module in your design
- Connects to the board on which a design is running
- Collects data when a trigger event occurs
- Displays data on your computer

How does it work?
SignalTap II Operation

USB-Blaster cable

FPGA

KEY[0]  KEY[1]  FSM

Clock

enable

8-bit Shift Register

Clock

Green LEDs

SignalTap Module
Setup SignalTap II

Specify Hardware Connection

Specify Clock to Latch data on

Specify Signals to Watch
Adding signals
Recompile Project

- For changes to take effect recompile project
- Once recompiled, download it to the board
- Note: The circuit will be larger than before
  - Memory is used to store captured data

### Flow Summary

<table>
<thead>
<tr>
<th>Category</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flow Status</td>
<td>Successful</td>
</tr>
<tr>
<td>Quatrus II Version</td>
<td>8.0 Build 215 05/29/2008 SJ Full Version</td>
</tr>
<tr>
<td>Revision Name</td>
<td>signaltap</td>
</tr>
<tr>
<td>Top-Level Entity Name</td>
<td>signaltap</td>
</tr>
<tr>
<td>Family</td>
<td>Cyclone II</td>
</tr>
<tr>
<td>Device</td>
<td>EP2C35F672D6</td>
</tr>
<tr>
<td>Timing Models</td>
<td>Final</td>
</tr>
<tr>
<td>Met timing requirements</td>
<td>Yes</td>
</tr>
<tr>
<td>Total logic elements</td>
<td>568 / 33,216 (2%)</td>
</tr>
<tr>
<td>Total combinatorial functions</td>
<td>368 / 33,216 (1%)</td>
</tr>
<tr>
<td>Dedicated logic registers</td>
<td>454 / 33,216 (1.4%)</td>
</tr>
<tr>
<td>Total registers</td>
<td>154</td>
</tr>
<tr>
<td>Total pins</td>
<td>15 / 475 (3%)</td>
</tr>
<tr>
<td>Total virtual pins</td>
<td>14</td>
</tr>
<tr>
<td>Total memory bits</td>
<td>1,792 / 433,840 (&lt;1%)</td>
</tr>
<tr>
<td>Embedded Multiplier 9-bit elements</td>
<td>0 / 70 (0%)</td>
</tr>
<tr>
<td>Total FLLs</td>
<td>0 / 4 (0%)</td>
</tr>
</tbody>
</table>
Setup Event Trigger

Click here to begin capture
Trigger the event and Analyze the results
Summary

- Learned how to
  - Use Quartus II CAD Software
  - Compile projects in Quartus II
  - Target design onto DE2 board
  - View results of compilation
  - Use SignalTapII

- Where do we go from here?
Concluding Remarks