Week 4
Announcement

• Office Hours changed

Office hours:

- Monday 10-11
- Tuesday 11-12
- Friday 12-1
Behavioral Modeling --- Continue

• **Conditional Statements**
• **Syntax**

  ```
  if (expression)
      then clause
  else
      else clause
  ```

• **Note:**
  • Else clause is optional

• **Conditional Operator**
  • Can be used in place of If...Then...Else
  • **Syntax:** *Condition ? True Expression : False Expression ;*
• Case
  – Syntax
• `case (expression)`
  alternative 1 : `statement(s) 1 ;`
  alternative 2 : `statement(s) 2 ;`
  ...
  ...
  alternative n : `statement(s) n ;`
  default : `default statement(s) ;`
endcase
  – Default
    • Implemented when none of the alternatives are true
  – Variations
    • Replace `case` with `casex` or `casez` for comparing don’t cares or high impedance states
# Operators

<table>
<thead>
<tr>
<th>Operation Type</th>
<th>Symbol</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arithmetic</td>
<td>*</td>
<td>Multiplication</td>
</tr>
<tr>
<td></td>
<td>/</td>
<td>Division</td>
</tr>
<tr>
<td></td>
<td>+</td>
<td>Addition</td>
</tr>
<tr>
<td></td>
<td>-</td>
<td>Subtraction</td>
</tr>
<tr>
<td></td>
<td>%</td>
<td>Modulus</td>
</tr>
<tr>
<td>Logical</td>
<td>!</td>
<td>Negation</td>
</tr>
<tr>
<td></td>
<td>&amp; &amp;</td>
<td>AND</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Relational</td>
<td>&gt;</td>
<td>Greater Than</td>
</tr>
<tr>
<td></td>
<td>&lt;</td>
<td>Less Than</td>
</tr>
<tr>
<td></td>
<td>&lt;=</td>
<td>Greater Than or Equal To</td>
</tr>
<tr>
<td></td>
<td>&gt;=</td>
<td>Less Than or Equal To</td>
</tr>
<tr>
<td>Equality</td>
<td>==</td>
<td>Equality</td>
</tr>
<tr>
<td></td>
<td>!=</td>
<td>Inequality</td>
</tr>
<tr>
<td></td>
<td>!==</td>
<td>Case Inequality</td>
</tr>
</tbody>
</table>

# Operators

<table>
<thead>
<tr>
<th>Operation Type</th>
<th>Symbol</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitwise</td>
<td>~</td>
<td>Bitwise Negation</td>
</tr>
<tr>
<td>Operation Type</td>
<td>&amp;</td>
<td>Bitwise AND</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bitwise OR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bitwise XOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>^</td>
<td>Bitwise XNOR</td>
</tr>
<tr>
<td>Reduction</td>
<td>&amp;</td>
<td>Reduction AND</td>
</tr>
<tr>
<td></td>
<td>~&amp;</td>
<td>Reduction NAND</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reduction OR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reduction NOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reduction XOR</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>^</td>
<td>Reduction XNOR</td>
</tr>
<tr>
<td>Shift</td>
<td>&gt;&gt;</td>
<td>Right Shift</td>
</tr>
<tr>
<td></td>
<td>&lt;&lt;</td>
<td>Left Shift</td>
</tr>
<tr>
<td>Concatenation</td>
<td>{</td>
<td>Concatenation</td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td></td>
<td>{</td>
<td>Replication</td>
</tr>
<tr>
<td></td>
<td>}</td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>:</td>
<td>Conditional</td>
</tr>
</tbody>
</table>

## Operator Precedence

<table>
<thead>
<tr>
<th>Operators</th>
<th>Symbols</th>
<th>Precedence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unary</td>
<td>+, - !~</td>
<td>Highest</td>
</tr>
<tr>
<td>Multiply, Divide, Modulus</td>
<td>* / %</td>
<td></td>
</tr>
<tr>
<td>Add, Subtract</td>
<td>+</td>
<td></td>
</tr>
<tr>
<td>Shift</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Relational</td>
<td>%</td>
<td></td>
</tr>
<tr>
<td>Equality</td>
<td>!</td>
<td></td>
</tr>
<tr>
<td>Reduction</td>
<td>&amp;., ~&amp;</td>
<td></td>
</tr>
<tr>
<td></td>
<td>^., ^</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>.,</td>
</tr>
<tr>
<td>Logical</td>
<td>&amp;&amp;</td>
<td></td>
</tr>
<tr>
<td>Conditional</td>
<td>?:</td>
<td>Lowest</td>
</tr>
</tbody>
</table>

• **Numbers**
  
  — **Syntax**
  
  • **Sized**
    
    — *Size*’*Format Number*
    
    — *Size*
      
      » Number of digits
    
    — *Format*
      
      » h (Hexadecimal)
      
      » d (Decimal)
      
      » o (Octal)
      
      » b (Binary)
    
    — Number
      
      » Number specified
  
  • **Unsized**
    
    — ’*Format Number*
  
  • **Examples**
    
    — 4’h a729
    
    — ‘d 62923
    
    — 8’b 1101zzzz
    
    — 16’h x
Loops

- **While**
  - Syntax
    ```
    while (condition)
    begin
      statement(s);
    end
    ```

- **For**
  - Syntax
    ```
    for (initial condition; termination condition; control variable change)
    begin
      statement(s);
    end
    ```

- **Repeat**
  - Repeats a given number of times based on the number, variable, or signal value given
  - Syntax
    ```
    repeat (number, variable, or signal value)
    begin
      statement(s);
    end
    ```
Loops

• Forever
  – Statement executes forever
    • Can be disabled by the keyword disable
  – Syntax
    • forever statement;
  – Example
    • forever #10 clk=~clk;
• Concatenation Operator
  – {..., ...}
  – Allows several wires to be combined together into a single multi-bit wire.
• In the next slides we will implement a full example to program the FPGA.

• Try to follow these steps to create your first project.
- Open the ISE project Navigator
- Type the project name

![New Project Wizard dialog box](image)

- **Name:** circuit
- **Location:** C:\Users\YourName\Documents\Circuit Project
- **Working Directory:** C:\Users\YourName\Documents\Circuit Project
- **Description:**

Select the type of top-level source for the project

- **Top-level source type:** Schematic
- Select these options then click next

**New Project Wizard**

**Project Settings**

Specify device and project properties.

<table>
<thead>
<tr>
<th>Property Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Category</td>
<td>All</td>
</tr>
<tr>
<td>Family</td>
<td>Spartan3E</td>
</tr>
<tr>
<td>Device</td>
<td>XC3S1000E</td>
</tr>
<tr>
<td>Package</td>
<td>CPL32</td>
</tr>
<tr>
<td>Speed</td>
<td>-4</td>
</tr>
<tr>
<td>Top-Level Source Type</td>
<td>Schematic</td>
</tr>
<tr>
<td>Synthesis Tool</td>
<td>XST (VHDL/Verilog)</td>
</tr>
<tr>
<td>Simulator</td>
<td>ISim (VHDL/Verilog)</td>
</tr>
<tr>
<td>Preferred Language</td>
<td>Verilog</td>
</tr>
<tr>
<td>Property Specification in Project File</td>
<td>Store all values</td>
</tr>
<tr>
<td>Manual Compile Order</td>
<td>□</td>
</tr>
<tr>
<td>VHDL Source Analysis Standard</td>
<td>VHDL-93</td>
</tr>
<tr>
<td>Enable Message Filtering</td>
<td>□</td>
</tr>
</tbody>
</table>

[Image of the New Project Wizard window]

- More Info
- < Back
- Next >
- Cancel
- click finish
- Select Project → new source
- Select Verilog module and write the file name, then click next
- click next
- Click Finish
- The file will be created and it will look like as below

```vhd
--timescale 1ns / 1ps

// Company:
// Engineer:
// Create Date: 23:59:38 02/05/2011
// Module Name: circuit
// Project Name:
// Target Devices:
// Tool versions:
// Description:
// Dependencies:
// Revision:
// Revision 0.01 - File Created
// Additional Comments:

module circuit(
);

endmodule
```
- Add your code shown

```plaintext
module circuit(switch, enable, led);

input switch;
input enable;
output led;
integer count;
reg led;

always @(switch)
begin
  led = switch;
end

if (enable == 1) begin
  led = switch;
end
else
begin
  led = 0;
end
```

```python
# Your code goes here
```
module circuit(switch, enable, led);
    input switch;
    input enable;
    output led;
    integer count;
    reg led;

    always @(switch)
    begin
        if (enable==1)
            begin
                led=switch;
            end
        else
            begin
                led=0;
            end
    end
endmodule
- To simulate the project select new source
- Select Verilog Test Fixture and write the file name then click next.
- Select the circuit module then click next
- click Finish to create the test file
- This is how it will look like
- Add your simulation code to the test file

```verilog
// Outputs
wire led;

// Instantiate the Unit Under Test (UUT)
circuit uut (  
    .switch(switch),  
    .enable(enable),  
    .led(led)  
);

initial begin  
    // Initialize Inputs
    switch = 0;  
    enable = 0;

    // Wait 100 ns for global reset to finish
    #100;  
    switch = 0;  
    enable = 1;

    #100;  
    switch = 1;  
    enable = 1;

    #100;  
    switch = 1;  
    enable = 0;

    #100;

    // Add stimulus here

end
```
- Select the simulation option from Design tab
- Select test1.v file and D-Click Simulate Behavioral Model to open ISIM simulator
- ISIM application, use zoom in, out to view the results
- To implement the code on the FPGA board go to implementation mode

```vhdl
-- Outputs

// Instantiate the Unit Under Test (UUT)
circuit out :
  switch(switch),
  enable(enable),
  led(led)
end;

// Initial begin
// Initialise Inputs
switch = 0;
enable = 0;

// Wait 100 ns for global reset to finish
#100;
switch = 0;
enable = 1;

#100;
switch = 1;
enable = 1;

#100;
switch = 0;
enable = 0;

#100;
switch = 1;
enable = 1;

#100;
switch = 0;
enable = 1;

// Add stimulus here
end
```
- Select circuit.v file and select I/O planning (PlanAhead-Presynthesis) to open PlanAhead application to assign the FPGA I/O ports to the code I/O
- Click yes to create UCF file and open Planahead application
- PlanAhead Application after its opened
- Select I/O ports, drag and drop each of them to Pin in the Package figure, refer to [http://www.digilentinc.com/Data/Products/BASY2/Basys2_rm.pdf](http://www.digilentinc.com/Data/Products/BASY2/Basys2_rm.pdf) to find the pin definition.
- After pin assignment
- Click save design and exit
- To Prepare the .bit File to Program the FPGA,
- Set the FPGA Startup Clock to JTAG Clock, Right click on Generate Programming File in the Processes Window, select Process Properties
Change **Startup Options** to **JTAG clock** as shown, then press OK.
Create the .bit file, Double click on Generate Programming File in the Processes Window
Make sure that every thing is compiling and building
Connect Basys2 to your computer and make sure it's correctly configured.
Open Adept2.1 (Downloaded from Digilent’s Website)
http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,66,69&Prod=ADEPT
Click on Browse (next to FPGA Option) and select the bit file from the project Directory.
Make sure the basys2 is on, click on the program, and test your fpga
Another example

• The following circuit will blink LED #0 when enabled
  • Period = 2 seconds
  • Enable = Switch #0
    • When Switch #0 is off, LED #0 is off

Pin Assignments

• Clk
  - B8
• Enable (SW0)
  - P11
• LED (LD0)
  - M5

module blinking_led(clk, led, enable);
input clk;
output led;
input enable;
reg led;
integer count;
always @(posedge clk)
begin
  if (enable==0)
  begin
    • The following circuit will blink LED #0 when enabled
      • Period = 2 seconds
      • Enable = Switch #0
        • When Switch #0 is off, LED #0 is off
  end else begin
    if (count > 50000000) begin
      count=0;
      led=~led;
    end
    count = count+1;
  end
end
endmodule
Basys Board Resources

Basys board documentation and resources can be found at the Digilent Website. Click on http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,400,790&Prod=BASYS2 to go right to these resources.

Quick Links
Reference Manual
http://www.digilentinc.com/Data/Products/BASYS2/Basys2_rm.pdf
Schematics
http://www.digilentinc.com/Data/Products/BASYS2/Basys2_sch.pdf

Adept Programming Suite

The Adept Programming Suite is required to the Basys2 board using the USB cable. It can be downloaded by clicking http://www.digilentinc.com/Products/Detail.cfm?NavPath=2,66,69&Prod=ADEPT
Required Software

- Xilinx ISE WebPack
  - Project Navigator (Main workspace for verilog)
  - ISIM (Simulator)
  - PlanAhead (to assign Pins)

- iVerilog (Simulator)

- Diligent
  - configuration and print the implementation of the fpga board
References

  - (Prentice Hall), 2003
- Donald E. Thomas and Philip R. Moorby, *The Verilog Hardware Description Language*,
- David R. Smith and Paul D. Franzon, *Verilog Styles of Digital Systems*,
  - Prentice Hall, Inc.,
  - 2000
- *Digilent BASYS 2 System Board Schematics*, Digilent, Inc., December 12, 2008