[Introduction to Computer Architecture'21] Lab: Verilog Syntax 2

- **What is Verilog?**

  Verilog HDL is a hardware description language used to design and document electronic systems. Verilog HDL allows designers to design at various levels of abstraction. It is the most widely used HDL with a user community of more than 50,000 active designers.

- **Overall Module Structure:**

```
module name (args…);
        input …; // define input ports
        output …; // define output ports
        wire …; // internal wires
        reg …; // internal or output regs
        // the parts of the module body are
        // executed concurrently
endmodule
```

- **Assignments:**
  - **Wire:** Results in combinational logic
    1. assign variable = exp.;
  - **Reg:** Result in combinational or sequential logic
    1. Blocking: variable = exp.;
    2. non-Blocking: variable <= exp.;

- **Block Structure:**
  - Type 1: **always** // repeats until simulation is done.
  - Type 2: **initial** // executed once at beginning of simulation.

[Introduction to Computer Architecture'21] Lab: Verilog Syntax 2

- **Data Types:**
  - reg and wire are the main variable types.
  - Possible values for wire and reg variables:
    1. 0: logic 0, false
    2. 1: logic 1, true
    3. X: unknown logic value
    4. Z: High impedance state
  - Registers:
    1. Abstract model of a data storage element.
    2. A reg holds its value from one assignment to the next.
    3. Examples:
       - reg a; // a scalar register
       - reg [3:0] b; // a 4-bit vector register
  - Wires:
    1. Wire variables model physical connections.
    2. They don't hold their value(driver).
    3. Examples:
       - wire d; \\ a scalar wire
       - wire [3:0] e; \\ a 4-bit vector wire
- **Number Representation:**
  - <size>'<base><number>
  - base is d(decimal), h(hex decimal), o(octal), or b(binary)
  - Examples:
    1. 4'b1001 // a 4-bit binary number
    2. 8'h2fe4 // an 8-bit hex number
- **Relational Operators:**
  - A<B, A>B, A<=B, A>=B, A==B, A!=B
  - A===B, A!==B
  - !, &&, | |
  - ~, &, |, ^
  - {a, b[3:0]} // example of concatenation
- **Conditional:**
  - If (<expr>) begin <statement> end else begin<statement> end
  - Example:

```
if (x > y)
begin
    x = y;
end
else
begin
    y = x;
end
```

- **Testbench:**
  - All your test code will be inside an initial block!
  - If you want new temp variables you need to define those outside the procedural blocks.
  - Generate the wave window of the inputs and the outputs.
  - How to run:
    1. Go to **simulation sources folder** in your project
    2. **Create new file** as a testbench
    3. Write your own **test**
    4. Click on **Run Simulation** and see the **WAVE window**
  - Example:

```
module test();
reg x,y;
wire [3:0] z;
gates g1(x,y,z);
initial
    begin
        $monitor("a is %b, b is %b, output is %b ",x,y,z);
        x = 0;
        y = 1;
        #5
        x = 0;
        y = 0;
        #5
        x = 1;
        y = 1;
    end
```

- **How to solve Problems:**
  1. **Design** specs
     - Paper and pen
  2. **Coding**
     - Verilog syntax
  3. **Synthesize** your code
     - From the left menu: Click on Run Synthesis
  4. **Simulate** your code
     - From the left menu: Click on Run Simulation (***to see the WAVE window***)
     - From the left menu: Click on RTL Analysis then click on Schematic (***to see the RTL Design***)
  5. **Run** your program
     - Create a new module for your board
     - Run this module on your board

- **Exercise 1: Bit-wise Operators with Testbench:**

Operator	Type
&	AND
~&	NAND
|	OR
~|	NOR
^	XOR
~^	XNOR
~	NOT

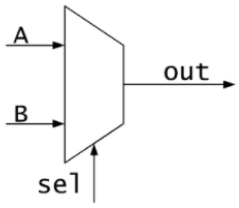
```
module gates(
    input a,
    input b,
    output [3:0] z
    );
wire a,b;
reg [3:0]z;

always @(a or b)
begin
    z[0] = a & b;
    z[1] = a | b;
    z[2] = a ^ b;
    z[3] = !(a | b);
end
endmodule
```



```
module test();
reg x,y;
wire [3:0] z;
gates g1(x,y,z);
initial
    begin
        $monitor("a is %b, b is %b, output is %b ",x,y,z);
        x = 0;
        y = 1;
        #5
        x = 0;
        y = 0;
        #5
        x = 1;
        y = 1;
    end
endmodule
```

- Make **Module test** as a top file
- **Run simulation** to see the output and the WAVE window (**your turn!**)

- **Exercise 2: 2x1 Multiplexer with Testbench:**



sel	out
0	A
1	B

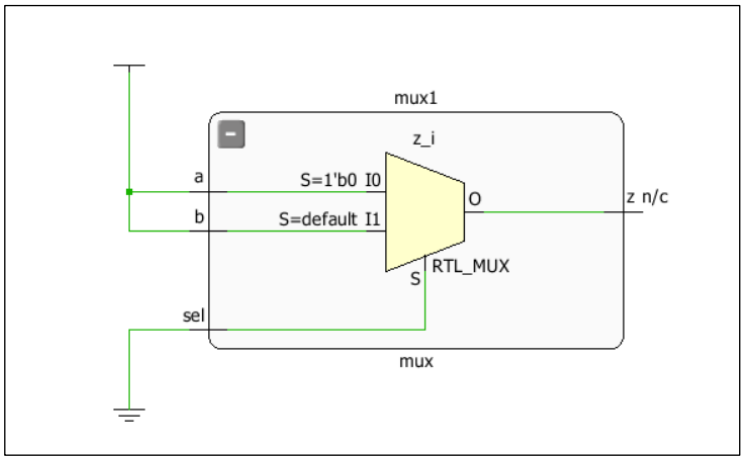
```
module mux(
    input a,
    input b,
    input sel,
    output z
    );
wire a,b,sel;
reg z;

always @(a or b or sel)
begin
    if(sel == 0)
    begin z = a; end
    else if (sel == 1)
    begin z = b; end
    else
    begin z = 1'bx; end
end
endmodule
```

Mux module

```
module test();
reg x,y,sel;
wire z;
mux mux1(x,y,sel,z);
initial
    begin
        $monitor("a = %b, b = %b, sel = %b, output is %b ", x, y, sel, z);
        x = 0;
        y = 1;
        sel = 0;
        #20
        x = 0;
        y = 0;
        sel = 1;
        #20
        x = 1;
        y = 1;
        sel = 1'bx;
    end
endmodule
```

Testbench module

WAVE window



RTL Design

Verilog + ZYBO Z7 board Help:

- Check this link for Verilog syntax: https://www.nandland.com/verilog/tutorials/index.html
- Check this link for ZYBO Z7 board info.: https://digilent.com/reference/programmable-logic/zybo/start

If you need any help regarding anything about the course, ask:

- Engr. Ahmad M. Abdel-Hafeez: akassem@nu.edu.eg
- Engr. Mohammad Rady: mrady@nu.edu.eg