1 Lab Report Grading

1. A lab report is required for each lab, from every individual. Reports have to be typed.
2. The format for the lab reports will be:
   - A title page of the course title, lab #, lab title, date, your name
   - Conclusion: Brief conclusion on the lab and what your learned from it
   - Required Materials for each lab
3. Late lab report will be Accepted with the penalty of 50 points per day.

2 Lab Schedule

- Verilog Introduction – Lab1 (a) - Logic level Modeling
- Verilog Introduction – Lab1 (b) - Behavioral Modeling
- Verilog Introduction – Lab2 (a) - Switch Level Modeling
- Verilog Introduction – Lab2 (b) - User-Defined Primitives
- Verilog Introduction – Lab3 – Finite State machine
- The project – Design of MIPS computer system
- The progress report
- The final report

3 Using Verilog

By now you should have UNIX accounts on the EESUN machines. If you still do not have the accounts, please inform your TA. To run verilog, you need to do the following:

- To invoke Verilog-XL simulator, type the command *verilog file1.v file2.v* .....

- To invoke Verilog-XL simulator with SimControl, type the command *verilog +gui file1.v file2.v .....*

4 Logic level modeling

```
****** Example 1: Ripple Carry counter Top Block *******

-------------------------------------------------
module ripple_carry_counter(q, clk, reset);

output [3:0] q;
input clk, reset;

T_FF tff0(q[0], clk, reset);
T_FF tff1(q[1], q[0], reset);
T_FF tff2(q[2], q[1], reset);
T_FF tff3(q[3], q[2], reset);

endmodule
-------------------------------------------------

****** Example 2: Flip-flop T_FF  *******

-------------------------------------------------
module T_FF(q, clk, reset);

output q;
input clk, reset;
wire d;

D_FF dff0(q, d, clk, reset);
not n1(d, q); // not is a verilog-provided primitive

end module
-------------------------------------------------

****** Example 3: Flip_flop D_FF  *******
```

```
-------------------------------------------------
module D_FF(q, d, clk, reset);

output q;
input d, clk, reset;
reg q;

// lots of construction here....

always @(posedge reset or negedge clk)
if (reset)
  q = 1'b0;
else
  q = d;

endmodule
-------------------------------------------------
```

****** Example 4: Stimulus Block *******

```
-------------------------------------------------
module stimulus;

reg clk;
reg reset;
wire [3:0] q;

// instantiate the design block

ripple_carry_counter r1(q, clk, reset);

// control the clock signal that drives the design block. Cycle time = 10

initial
  clk = 1b'0 ; // set clk to 0
always
 #5 clk=~clk ; // toggle clk every 5 time units
```

```
// control the reset signal that drives the design block
// reset is asserted from 0 to 20 and from 200 to 220

initial
begin

 reset = 1'b1;
 #15 reset = 1'b0;
 #180 reset = 1'b1;
 #10 reset = 1'b0;
 #20 $finish; // teriminate the simulation
end

// Monitor the outputs

initial
 $monitor($time, ``Output  q = %d'', q);
endmodule
```

5 The Problems:logic level

1. Design a 4-to-1 multiplexer with 2-select signals. Assume that signals s1 and s0 do not get the value x or z. plot the diagram and the truth table for the MUX

2. Simulate a 4-bit ripple carry adder using verilog gate level primitives

   (a) Using zero time delay for the gates.

   (b) Choose any non-zero time delay for the gates.

   (c) find out how to use contiuous assignment to create 4-bit adder

3. Simulate a 4-bit full adder with carry lookahead using verilog gate level primitives. Comparing the results with Problem 2 in the conclusion (using the same time delay values for the gates in case (b) of Prob. 2)

6 Behavioral Modeling

Verilog provides designers the ability to describe design functionality in an algorithmic manner. In other words, the designer describes the behaviors of the circuit. Behavioral Verilog constructs are similar to C language constructs in many ways.

- always and initial
- case, casex, casez
- if and else
- blocking and nonblocking procedural assignments

7 blocking and nonblocking

```
****Example 1--- Blocking Assignment  ****

-------------------------------------------------------
 reg x, y,z;
 reg [15:0] reg_a, reg_b;
 integer count;

 // All behavior statement must be inside an initial or always block

 initial
 begin
  x = 0; y = 0; z = 1; // scalar assignments
  count = 0;
  reg_a = 16'b0; reg_b =reg_a; // initialize variable

  #15 reg_a[2] = 1'b1 ; // Bit select assignment with delay
  #10 reg_b[15:13] = {x, y, z}

  count = count +1 ;
 end
-------------------------------------------------------
```

```
****Example 2--- Nonblocking Assignment  ****

-------------------------------------------------------
 reg x, y,z;
 reg [15:0] reg_a, reg_b;
 integer count;

 // All behavior statement must be inside an initial or always block

 initial
 begin
  x = 0; y = 0; z = 1; // scalar assignments
  count = 0;
  reg_a = 16'b0; reg_b =reg_a; // initialize variable

   reg_a[2] <= #15 1'b1 ; // Bit select assignment with delay
   reg_b[15:13] <=  #10 {x, y, z}

  count = count +1 ;
 end
-------------------------------------------------------
```

8 Multiway Branching

```
*** Example 3  Case statement with x and z *****
-----------------------------------------------------------
module  demux1_to_4 (out0, out1, out2, out3, in ,s1, s2);

//port declarations

output out0, out1, out2, out3;
reg out0, out1, out2, out3;
input in;
input s1, s0;

always @(s1 or s0 or in)
```

```
case ({s1, s0}) // switch based on control signals
 2'b00 : begin out0 =in; out1 = 1'bz ; out2 = 1'bz ; out3 = 1'bz ; end
 2'b01 : begin out0 =1'bz; out1 = in ; out2 = 1'bz ; out3 = 1'bz ; end
 2'b10 : begin out0 =1'bz; out1 = 1'bz ; out2 = in ; out3 = 1'bz ; end
 2'b11 : begin out0 =1'bz; out1 = 1'bz ; out2 = 1'bz ; out3 = in ; end

// Account for unknown signals on selct. If any select signal is x
// then outputs are x. If any select signal is z, outputs are z.
// If one is x and the other is z, x gets priority.

 2'bx0, 2'bx1, 2'bxz, 2'bxx, 2'b0x, 2'b1x, 2'bzx :
  begin
 out0 = 1'bx ; out1 = 1'bx; out2 = 1'bx; out3 = 1'bx;
  end

 2'bz0, 2'bz1, 2'bzz, ,2'b0z, 2'b1z  :
  begin
 out0 = 1'bz ; out1 = 1'bz; out2 = 1'bz; out3 = 1'bz;
  end
endcase

endmodule
-----------------------------------------------------------------


*** Example 4 casex use *******
-----------------------------------------------------------------
// casex treats all x and z values in the case item or
// the case expression as don't care

 reg [3:0] encoding ;
 integer state;

  casex (encoding)  // logic value x represents a don't care bit
  4'b1xxx : next_state = 3;
  4'bx1xx : next_state = 2;
  4'bxx1x : next_state = 1;
  4'bxxx1 : next_state = 0;
```

```
  default : next_state = 0;
  endcase
-----------------------------------------------------------------
```

9 The Problems: behavioral modeling

1. Desing an 8-function ALU that takes 4-bit input a and b and a 3-bit input signal select, and 5-bit output out. The ALU implements the following functions base on 3-bit input signal select. Ignore any overflow or underflow bits.

Select Signal	Function
3'b000	out = a
3'b001	out = a +b
3'b010	out = a -b
3'b011	out = a /b
3'b100	out = a %b (remainder)
3'b101	out = a << 1
3'b110	out = a >> 1
3'b111	out = (a>b) (magnitude compare)

   You must simulate all of eight functions in stimulus block.

2. Design an 8-bit counter by using a ''forever'' loop, named block, and disabling of named block. The counter starts counting at count = 5 and finishes at count = 67. The count is incremented at positive edge of clock. The clock has a time period of 10 and 50% duty cycle. The counter counts through the loop only once and then is disabled.

   Required Materials:

   1. Your programs with complete comments, results and conclusions
   2. For the first problem, The diagrams of your design for all hierachy levels with net names as used in your verilog files marked on it (this can be hand drawn)

3. Write both programs with Verilog HDL behavioral modeling
4. Due days: Sep 21, 2006