



CSE140L: Components and Design Techniques for Digital Systems Lab

Final Review

Mohsen Imani

Example of True/False questions

- Everybody should submit his/her CAPE evaluation because that is an extremely important feedback for the university and for instructors to improve the quality of teaching **T F**

Example of True/False questions

- 1) Verilog is a HDL **T F**
- 2) The LHS of a procedural assignment should always be a reg **T F**
- 3) Verilog is used to describe a behavior that is synthesized into real hardware **T F**
- 4) A reg variable is a register **T F**
- 5) Dynamic power consumption does not depend on operating voltage **T F**
- 6) The TinyCPU lab homework was an example of combinational circuit **T F**
- 7) Mixing blocking and non-blocking assignment in the same procedure is highly recommended **T F**
- 8) Wire and reg are data types in Verilog **T F**
- 9) \$display is a keyword that automatically synthesize a display for your digital circuit **T F**
- 10) Xilinx is a major FPGA vendor **T F**
- 11) If “`timescale 1s/1ms”, then #100 is a delay of 100ms **T F**

System Tasks

- The \$ sign denotes Verilog system tasks, there are a large number of these, most useful being:
 - \$display(“The value of a is %b”, a);
 - Used in procedural blocks for text output.
 - The %b is the value format (binary, in this case...)
 - \$monitor
 - Similar to display, but executes every time one of its parameter changes
 - \$finish;
 - Used to finish the simulation.
 - Use when your stimulus and response testing is done.
 - \$stop;
 - Similar to \$finish, but doesn't exit simulation.

Examples of Multiple Choice Questions

- 1) A FSM with 6 states requires at least:
 - a) 2-bit states
 - b) 3-bit states
 - c) 4-bit states
 - d) None of the above

- 2) The value of a displayed by the following code at time 30 is
 - `#20 a <= 10;`
 - `#5 a <= 20;`
 - a) 10
 - b) 20
 - c) 5
 - d) None of the above

Procedural Assignment

Example: Blocking Vs Nonblocking Assignments

```
1 module block_nonblock();
2 reg a, b, c, d, e, f;
3
4 // Blocking assignments
5 initial begin
6     a = #10 1'b1; // The simulator assigns 1 to a at time 10
7     b = #20 1'b0; // The simulator assigns 0 to b at time 30
8     c = #40 1'b1; // The simulator assigns 1 to c at time 70
9 end
10
11 // Nonblocking assignments
12 initial begin
13     d <= #10 1'b1; // The simulator assigns 1 to d at time 10
14     e <= #20 1'b0; // The simulator assigns 0 to e at time 20
15     f <= #40 1'b1; // The simulator assigns 1 to f at time 40
16 end
17
18 endmodule
```

Examples of Multiple Choice Questions

3) The value of a displayed by the following code at time 30 is

```
#20 a = 10;  
#5  a = 20;
```

- a) 10
- b) 20
- c) 5
- d) None of the above

1) How many transistors an AND gate has?

- a) 2
- b) 4
- c) 6
- d) 8

Examples of Multiple Choice Questions

4) What's the keyword in Verilog testbenches to indicate that interrupts the simulation with the possibility of resuming it?

- a. \$finish
- b. \$monitor
- c. \$stop
- d. \$fish
- e. None of the above

5) Compared to the carry-lookahead adder, the ripple-carry adder is:

- a. Slower
- b. Faster
- c. Smaller
- d. Bigger
- e. a & c
- f. a & d
- g. b & c
- h. b & d

Example: code analysis

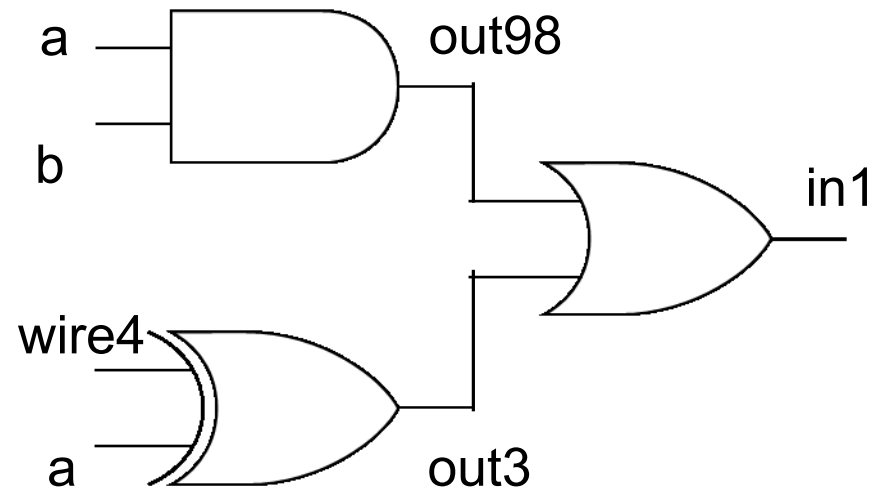
Which mistakes are contained in the following code?

```
Module my_multiplier(a, out, in);  
input [3:0] a;  
input clk;  
output in;  
reg in;  
  
always@(negedge clk) begin  
    if (out == 1)  
        in <= a;  
    else  
        in <= in+1;  
end
```

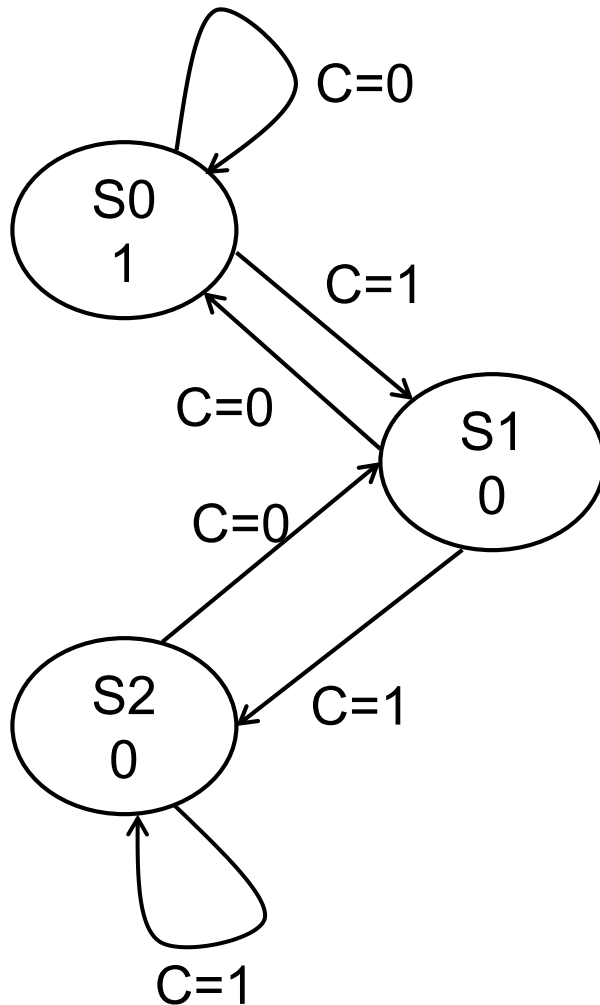
- 1) out is missing from port declaration (it should be an input port)
- 2) clk should be in the port list
- 3) reg in should be reg [3:0] in
- 4) endmodule is missing
- 5) This design needs a reset

Example: from code to circuit

```
module my_circuit(a, b, in1, wire4);  
input a,b, wire4;  
output in1;  
reg in1;  
wire out3;  
reg out98;  
  
always begin  
    out98 <= a & b;  
    in1 <= out98 | out3;  
end  
assign out3 = a ^ wire4;  
  
endmodule
```



Example: from state diagram to code



```

module my_fsm(C, clk, reset, out);
input C, clk, reset;
output out;
reg out;
reg [1:0] state;
reg [1:0] next_state;

```

```

parameter S0 = 2'b00;
parameter S1 = 2'b01;
parameter S2 = 2'b10;

```

```

always @(posedge clk) begin
if (reset)

```

```

state = S0;

```

```

else

```

```

state = next_state;

```

```

end

```

```

Always@(C or state) begin
case(state)
S0:
if (in == 0)
next_state = S0;
else
next_state = S1;

```

```

S1:
if (in == 0)
next_state = S0;
else
next_state = S2;

```

```

S2:
if (in == 0)
next_state = S1;
else
next_state = S2;
endcase

```

```

end

```

```

always@(state)
case(state)

```

```

S0: out = 1;
S1: out = 0;
S2: out = 0;

```

```

endcase

```

```

end
endmodule

```

Summary of CSE140L – SU217

- Transistors and circuit delays
- Basics of Verilog: operators, syntax
- Quartus IDE + Modelsim simulation (HW1)
- Implementation of Verilog modules (HW2)
- Implementation of testbenches (HW2,3,5)
- Re-using modules to implement larger modules (instantiation + explicit module connection) (HW3,4)
- Implementation of a large project using smaller modules as building blocks (HW4)
- Implementation of Mealy and Moore FSM (HW5)

Summary of CSE140L – SU217



- Digital circuits can be described using HDLs and synthesized automatically using IDEs such as Quartus
- HDLs are different from programming languages
- You can (and actually should) use and re-use components that you already implemented to build larger components

CAPE

Please submit your CAPE evaluations!



Good Luck for the Final Exam!