



## **CSE 313s**

# **Selected Topics in Computer Engineering**

## **Sheet 3**

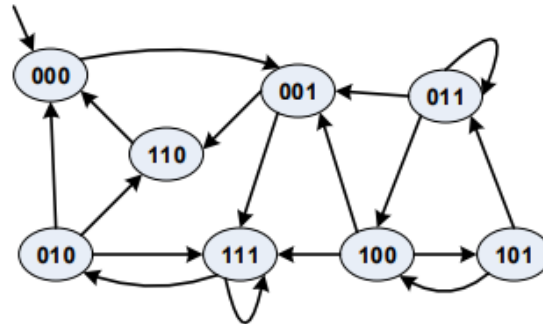
---

1. Select the disadvantage of using Model Checking
  - a. Concurrent systems cannot be analyzed using this method.
  - b. Producing a mathematical specification requires a detailed analysis of the requirements.
  - c. They require the use of specialized notations that can only be understood by domain experts.
  - d. All of the above
2. What is a BDD?
  - a. Boolean Decision Diagram
  - b. Binary Decision Diagram
  - c. Binary Decision Device
  - d. Binary Device Diagram
3. What are the two inputs to the model checker?
  - a. Implementation and specification
  - b. BDD and FSM
  - c. FSM of the design and properties
  - d. Verification and Validation
4. What is meant by the “counter example” generated by the Model Checker?
5. Which of the following techniques can be used to prove a general SAFETY PROPERTY?
  - a. Equivalence checking
  - b. Simulation
  - c. Model Checking
  - d. Emulation

**Consider the function:  $F = c.b + b.(a + d) + a'.d$  For the questions below, use variable order (from top to bottom): a,b,c,d.**

6. What are the cofactors of F w.r.t. a (i.e  $F_{a=0}$ ,  $F_{a=1}$ )? Provide a simplified expression with the minimum number of literals
7. Draw the BDD for the function F. Remember to mark the edge with ‘0’ or ‘1’ to indicate 0 and 1 cofactors. Please draw the ‘0’ edges on the left and the ‘1’ edges on the right.

8. Consider the following function:  $F(a,b,c,d) = \sum(2,3,12,14)$ . Draw the BDD to represent this function. Remember to mark the edge with '0' or '1' to indicate 0 and 1 cofactors. Please draw the '0' edges on the left and the '1' edges on the right.
9. For the following figure, perform a reachability analysis. Are there unreachable states?



10. For the previous problem, is the following property true?  
"State 101 can eventually be reached from the initial state"
11. What are the two types of assertions, and what are the differences between them?
12. What is the type of the following assertion. Explain its meaning.

```
property hash_delay_prop;
  @(posedge prop_clk) req ##5 gnt;
endproperty

hash_delay_check: assert property (hash_delay_prop);
```

13. What is the type of the following assertion. Explain its meaning.

```
assert (grant && request) begin
  $display ("Seems to be working as expected");
end
else begin
  current_time = $time;
  #1 $error("assert failed at time %0t", current_time);
end
```

14. Assume in a coverage report, you found the functional coverage = 60% while the code coverage = 100%. What could be the reasons for that? In another experiment you found functional coverage = 100% while the code coverage = 60%. What could be the reasons in that case?
15. What is coverage?
16. What is the meaning of code coverage?
17. What is a coverage driven verification?
18. What do the values x and z signify?

19. What is the difference between X and Z in verilog?
20. Which coverage has more importance, code coverage or functional coverage?
21. What is the difference between the following two declarations?

```
1 wire [7:0] w1;  
2 wire [0:7] w2;
```

22. What is 10'had?
23. Write a Verilog behavioral description of a four-bit adder module. The adder should have three inputs, *a*, *b*, and *c<sub>in</sub>*, and two outputs, *sum* and *c<sub>out</sub>*. Ports *c<sub>in</sub>* and *c<sub>out</sub>* are one bit, the other ports are four bits each.
24. Write a Verilog structural description of an eight-bit adder that uses two of the four-bit adders of the previous problem. (That is, instantiate the modules designed in the previous problem)
25. An accumulator has three inputs, *amt*, *reset*, and *clk*, and an output, *sum*. The accumulator has an internal 32-bit register which is updated as follows:
- On a positive edge of *clk* it adds *amt*, a 32-bit integer, to the register;
  - On the negative edge of *clk* it places the new *sum* on its outputs (until the next negative edge).
  - Whenever *reset* is high the register is set to zero and the output changes immediately.
- Write a Verilog behavioral description of this module.
26. The code below starts executing at *t* = 0. Show all changes in *a*, include the time of the change and the new value.

```
1 integer a;  
2 initial begin  
3     a = 1;  
4     #1;  
5     a = 2;  
6     #1;  
7     a = 3;  
8     #1;  
9     a <= 4;  
10    #1;  
11    a <= 5;  
12    #1;  
13    #3 a = a+1;  
14    #1;  
15    a = #3 a+1;  
16    #1;  
17    a <= #3 a+1;  
18    #1;  
19    a = a+1;  
20 end
```

27. The programmer expected execution to exit the loop below when either  $i$  was 1000 or  $a[i] == c$ , but that's not what happened. What goes wrong and how can it be fixed? The loop must be exited using a *disable* statement.

```

1 integer i, c;
2 integer a[0:999];
3
4 // ...
5 i = 0;
6 while( i < 1000 ) begin:LOOP
7   if( a[i] == c ) disable LOOP;
8   i = i + 1;
9 end

```

28. For the Verilog code segment below, which of the lines implement a shift register?

```

1 always@ (posedge clk)
2   begin
3     z = y; y = x;
4     y1= x1; z1 = y1;
5     z2<= y2; y2 <= x2;
6     y3<= x3; z3 <= y3;

```

29. If a variable is not assigned in all possible executions of an always statement then:
- A don't care is inferred
  - A latch is inferred
  - The variable is set to 0
  - The synthesis process will fail
30. An entry level engineer has implemented Verilog code with the objective of designing a counter that continuously counts down from 7 to 0 in binary and then back to 7 again (counts 111, 110, 101, ..., 000, 111, 110, 101, ... ). Unfortunately, the designer made a number of mistakes in the design. Please fix all the bugs using minimal changes. The counter bits are:  $c_3 c_2 c_1$  (Assume  $c_3$  is the most significant bit)

```

1 module top(clk, r, c1, c2, c3);
2   input r, clk;
3   output c1, c2, c3;
4   reg c1, c2, c3;
5
6   assign x = 1'b1;
7   my_unit u1(clk, r, x, x, c1);
8   my_unit u2(clk, r, c1, x, c2);
9   my_unit u3(clk, r, x, c2, c3);
10 endmodule
11
12 module my_unit(clk,r,a,b,c);
13   input clk,r,a,b;
14   output c;
15   reg c;
16
17   always@(negedge clk)
18     c <= (~a&~c) | ((~b)&c);
19
20   always@(r) c <= 1'b0;
21 endmodule

```

31. A recognizer has one input “X” and one output “Y”. At each clock cycle, the input “X” value is read. When a sequence of “101” is observed in the input sequence, the output, “Y”, will become 1, otherwise it will be 0.

- Draw Moore state machine diagram with minimum number of states.
- Write the Verilog representation of your Moore state machine.

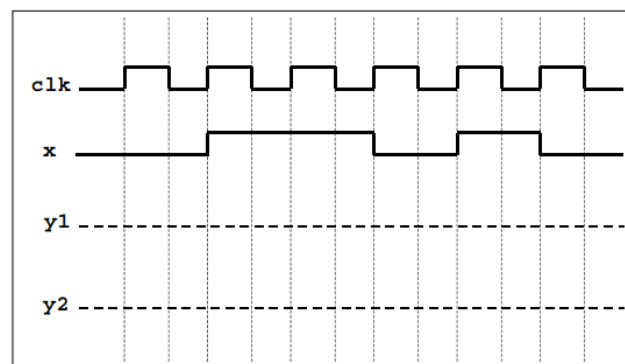
32. Use the following Verilog module to answer the question below:

```

1 module my_unit(clk, x, y1, y2);
2   input clk, x;
3   output y1, y2;
4   reg y1, y2;
5   wire w;
6
7   assign w = x | (~y1);
8   always@(posedge clk)
9     y2 <= w;
10
11  always@(negedge clk)
12    y1 <= y2;
13 endmodule

```

- Draw the output waveforms for the specified inputs.



- Draw the logic diagram for the circuit that represents **my\_unit**

33. Your job is to design a 3-bit ALU for the specifications in the table below. This unit has a two bit control lines ( $P_1 P_0$ ), to select the required operation, and 3-bit input data  $D[2:0]$ . The output lines are  $Q[2:0]$ .

$P_1 P_0$	Operation
0 0	$Q = D$
0 1	$Q = Q'$ (New Q is the complement of the current Q)
1 0	$Q = 2Q$ (New Q is twice the current Q)
1 1	$Q = 2Q + 1$ (New Q is twice the current Q plus 1)