Ain Shams University
Faculty of Engineering
Computer and Systems Department
Spring 2023

CSE 313s
Selected Topics in Computer Engineering
Sheet 3

1. What is coverage? - measure of verification progress and quality (verification of verification)
   - meaningful in case of randomized testing
   - code coverage -> automated - functional coverage -> manual
2. What is the meaning of code coverage? description of the 6 code coverage types discussed in sheet 2.
3. What is a coverage driven verification?
4. What is the difference between X and Z in verilog? x: unknown state ,Z: floating (high impedance)
   Both are important, as having different values indicate several problems as discussed.
5. Which coverage has more importance, code coverage or functional coverage?
6. What is the difference between the following two declarations?

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

   W1: msb is on left most
   W2: msb is the right most
7. What is 10'had? AD is a hexadecimal number represented in 10 bits.
   if bits greater than needed zeros are added to left if less than needed leftmost bits are truncated
8. Write a Verilog behavioral description of a four-bit adder module. The adder should have three inputs, a, b, and c_{in}, and two outputs, sum and cout. Ports c_{in} and c_{out} are one bit, the other ports are four bits each.
9. 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)
10. 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.

11. The code below starts executing a t = 0. Show all changes in a, include the time of the change and the new value.

```
integer a;
initial begin
  a = 1;
  #1;
  a = 2;
  #1;
  a = 3;
  #1;
  a <= 4;
  #1;
  a <= 5;
  #1;
  #3 a = a+1;
  #1;
  a = #3 a+1;
  #1;
  a <= #3 a+1;
  #1;
  a = a+1;
end
```

<= is a non-blocking assignment which means that execution is not blocked by this statement

12. 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.

```
integer i, c;
integer a[0:999];

// ...
i = 0;
while( i < 1000 ) begin:LOOP
  if( a[i] == c ) disable LOOP;
  i = i + 1;
end
```

disable statement returns execution sequence to the end of the named block.

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

```
always@ (posedge clk)
  begin
    z = y; y = x;
    y1= x1; z1 = y1;
    z2<= y2; y2 <= x2;
    y3<= x3; z3 <= y3;
```

14. If a variable is not assigned in all possible executions of an always statement then:
    a. A don't care is inferred
    b. A latch is inferred
    c. The variable is set to 0
    d. The synthesis process will fail

15. 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: c3 c2 c3 (Assume c3 is the most significant bit)

111
110
101
100
011
010
001
000

```
module top(clk, r, c1, c2, c3);
  input r, clk;
  output c1, c2, c3;
  reg c1, c2, c3;

  assign x = 1'b1;
  my_unit u1(clk, r, x, x, c1);
  my_unit u2(clk, r, c1, x, c2);
  my_unit u3(clk, r, x, c2, c3);
endmodule

module my_unit(clk,r,a,b,c);
  input clk,r,a,b;
  output c;
  reg c;

  always@(negedge clk)
    c <= (~a&~c) | ((~b)&c);

  always@(r) c <= 1'b0;
endmodule
```

```
module top(clk, r, c1, c2, c3);
input r, clk;
output c1, c2, c3;
reg c1, c2, c3;
assign x = 1'b1;
my_unit u1(clk, r, 0, x, c1);
my_unit u2(clk, r, c1, !c1, c2);
my_unit u3(clk, r, (c1||c2), !(c1||c2), c3);
endmodule
module my_unit(clk,r,a,b,c);
input clk,r,a,b;
output c;
reg c;
always@(negedge clk)
 c <= (~a&~c) | ((~b)&c);
always@(r)
 c <= 1'b1;
endmodule
```

16. 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.
    a. Draw Moore state machine diagram with minimum number of states.
    b. Write the Verilog representation of your Moore state machine.

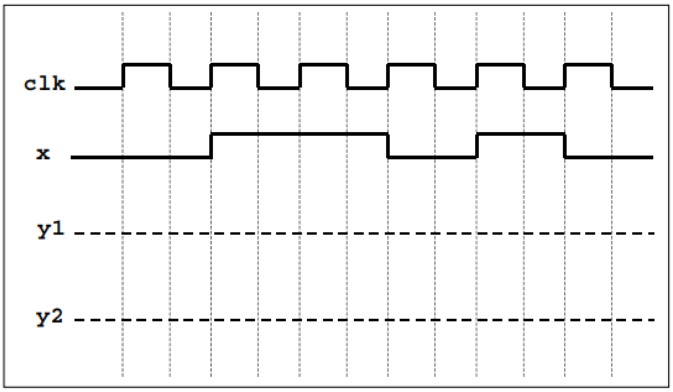
17. Use the following Verilog module to answer the question below:

```
module my_unit(clk, x, y1, y2);
  input clk, x;
  output y1, y2;
  reg y1, y2;
  wire w;

  assign w = x|(~y1);
  always@(posedge clk)
    y2 <= w;

  always@(negedge clk)
    y1 <=y2;
endmodule
```

a. Draw the output waveforms for the specified inputs.



b. Draw the logic diagram for the circuit that represents **my_unit**

18. 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)