

DIGITAL DESIGN AND COMPUTER ARCHITECTURE (252-0028-00L), SPRING 2021
OPTIONAL HW 2: SEQUENTIAL LOGIC AND VERILOG

Instructor: Prof. Onur Mutlu

TAs: Juan Gomez-Luna, Jisung Park, Hasan Hassan, Mohammed Alser, Lois Orosa, Minesh Patel,
Jawad Haj-Yahya, Haiyu Mao, Behzad Salami, Jeremie Kim, Giray Yaglicki, Can Firtina,
Geraldo De Oliveira Junior, Rahul Bera, Konstantinos Kanellopoulos, Nika Mansouri, Gagandeep Singh

Released: Monday, March 16, 2021

1 1 Verilog (I)

Please answer the following three questions about Verilog.

- (a) Does the following code result in a D Flip-Flop with a synchronous active-low reset? Please explain your answer.

```
1  module mem (input clk, input reset, input [1:0] d, output reg [1:0] q);
2  always @ (posedge clk or negedge reset)
3      begin
4          if (!reset) q <= 0;
5          else q <= d;
6      end
7  endmodule
```

- (b) Does the following code result in a sequential circuit or a combinational circuit? Please explain your answer.

```
1  module Mask (input [1:0] data_in, input mask, output reg [1:0] data_out);
2  always @ (*)
3      begin
4          data_out[1] = data_in[1];
5          if (mask)
6              data_out[0] = 0;
7      end
8  endmodule
```

(c) [10 points] Is the following code syntactically correct? If not, please explain the mistake(s) and how to fix it/them.

```
1  module fulladd(input a, b, c, output reg s, c_out);
2      assign s = a^b;
3      assign c_out = (a & b) | (b & c) & (c & a);
4  endmodule
5
6  module top ( input wire [5:0] instr, input wire op, output z);
7
8      reg[1:0] r1, r2;
9      wire [3:0] w1, w2;
10
11     fulladd FA1 (.a(instr[0]), .b(instr[1]), .c(instr[2]),
12                .c_out(r1[1]), .z(r1[0]));
13     fulladd FA2 (.a(instr[3]), .b(instr[4]), .c(instr[5]),
14                .z(r2[0]), .c_out(r2[1]));
15
16     assign z = r1 | op;
17     assign w1 = r1 + 1;
18     assign w2 = r2 << 1;
19     assign op = r1 ^ r2;
20
21 endmodule
```

2 2 Verilog (II)

Please answer the following four questions about Verilog.

(a) Does the following code result in a D Flip-Flop with asynchronous reset? Please explain why.

```
1  module dff (input clk, input reset, input [3:0] d, output reg [3:0] q);
2  always @ (posedge clk)
3  begin
4      if (reset == 0) q <= 0;
5      else q <= d;
6  end
7  endmodule
```

(b) Does the following code result in a sequential circuit or a combinational circuit? Explain why.

```
1  module concat (input clk, input data_in1, input data_in2,
2                  output reg [1:0] data_out);
3  always @ (posedge clk, data_in1, data_in2)
4  if (data_in1 > data_in2)
5  data_out = {data_in1, data_in2};
6  else
7  data_out = {data_in2, data_in1};
8  endmodule
```

(c) Is the following code syntactically correct? If not, please explain the mistake(s) and how to fix it/them.

```
1 module Inn3r ( input [3:0] d, input op, output s);
2   assign s = op ? (d[1:0] - d[3:2]) :
3                 (d[3:2] + d[1:0]);
4 endmodule
5
6 module top ( input wire [6:0] instr, input wire op, output reg z);
7
8   reg[1:0] r1, r2, r3;
9   wire [3:0] w1, w2;
10
11   Inn3r i0 (.instr(instr[1:0]), .op(instr[7]), .z(r1) );
12   Inn3r i1 (.instr(instr[3:2]), .op(instr[0]), .z(r2) );
13
14   assign z = r1 | r2;
15   assign w1 = r1 + 1;
16   assign w2 = r2 << 1;
17
18   top t (.instr({w1, w2, w1==w2}), .op(z), .z(r3));
19
20   assign op = r1 ^ r2 ^ r3;
21
22 endmodule
```

- (d) Does the following code correctly implement a counter that counts down from 10 to 1 (e.g., 10, 9, 8, ..., 2, 1, 10, 9, ...)? If so, say "Correct". If not, correct the code with minimal modification.

```
1 module the_final_count_down (clk, count);
2   wire clk;
3   reg[3:0] count = 10;
4   reg[3:0] count_next;
5
6   always @ * begin
7     count_next <= count;
8     if(count != 1)
9       count_next <= count_next - 1;
10    else
11      count_next <= 1;
12  end
13
14
15  always@(posedge clk)
16    count = count_next;
17 endmodule
```

Answer with concise explanation:

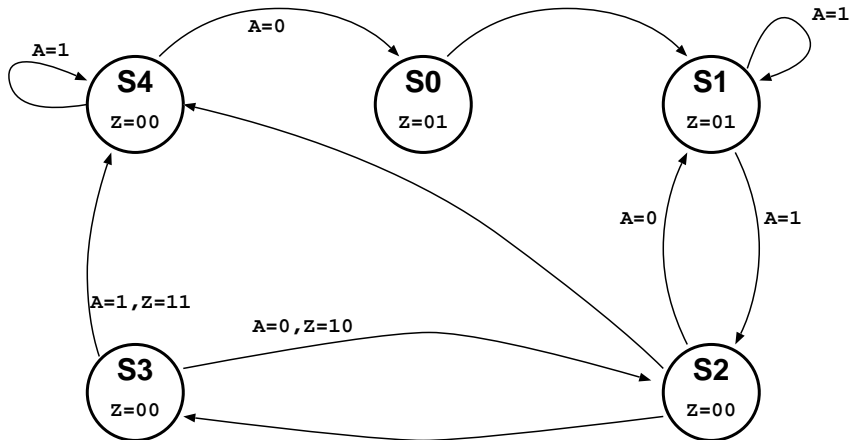
(e) Which of the combinational logic blocks does the following verilog code implement?

```
1 module mystery(input select, input enable, output result);  
2     wire [3:0] result;  
3     wire [1:0] select;  
4     wire enable;  
5  
6     assign result = enable << (select);  
7 endmodule
```

3 3 Finite State Machines (I)

This question has three parts.

- (a) [20 points] An engineer has designed a deterministic finite state machine with a one-bit input (A) and a two-bit output (Z). He started the design by drawing the following state transition diagram:



Although the exact functionality of the FSM is not known to you, there are **at least three mistakes** in this diagram. Please list **all** the mistakes.

- (b) [25 points] After learning from his mistakes, your colleague has proceeded to write the following Verilog code for a much better (and **different**) FSM. The code has been verified for syntax errors and found to be OK.

```
module fsm (input CLK, RST, A, output [1:0] Z);

    reg [2:0] nextState, presentState;

    parameter start    = 3'b000;
    parameter flash1   = 3'b010;
    parameter flash2   = 3'b011;
    parameter prepare  = 3'b100;
    parameter recovery  = 3'b110;
    parameter error    = 3'b111;

    always @ (posedge CLK, posedge RST)
        if (RST) presentState <= start;
        else     presentState <= nextState;

    assign Z = (presentState == recovery) ? 2'b11 :
               (presentState == error)    ? 2'b11 :
               (presentState == flash1)   ? 2'b01 :
               (presentState == flash2)   ? 2'b10 : 2'b00;

    always @ (presentState, A)
        case (presentState)
            start      : nextState <= prepare;
            prepare    : if (A) nextState <= flash1;
            flash1     : if (A) nextState <= flash2;
                       else nextState <= recovery;
            flash2     : if (A) nextState <= flash1;
                       else nextState <= recovery;
            recovery   : if (A) nextState <= prepare;
                       else nextState <= error;
            error      : if (~A) nextState <= start;
            default    : nextState <= presentState;
        endcase
endmodule
```


Draw a proper state transition diagram that corresponds to the FSM described in this Verilog code.

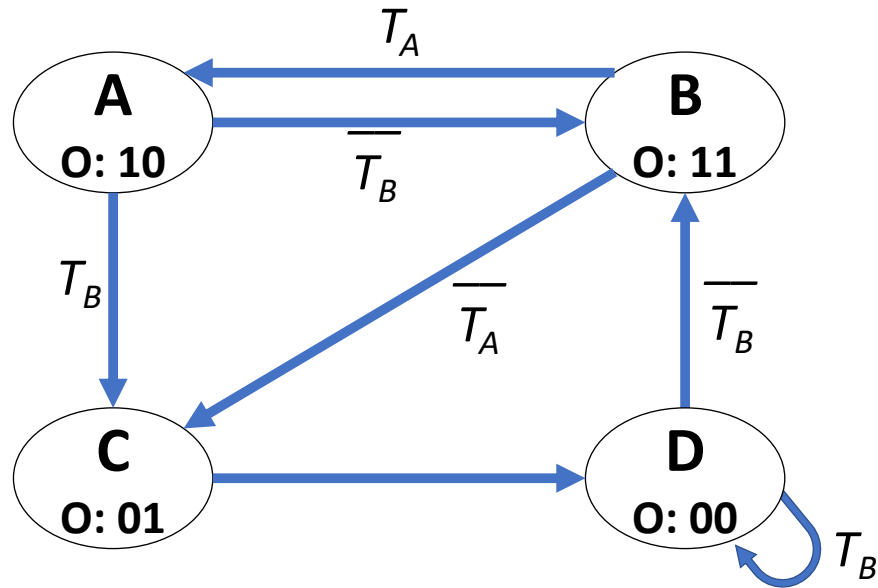


(c) [5 points] Is the FSM described by the previous Verilog code a Moore or a Mealy FSM? Why?



4 4 Finite State Machines (II)

You are given the following FSM with two one-bit input signals (T_A and T_B) and one two-bit output signal (O). You need to implement this FSM, but you are unsure about how you should encode the states. Answer the following questions to get a better sense of the FSM and how the three different types of state encoding we discussed in the lecture (i.e., one-hot, binary, output) will affect the implementation.



- (a) [2 points] There is one critical component of an FSM that is *missing* in this diagram. Please write what is missing in the answer box below.

- (b) [2 points] What kind of an FSM is this?

(c) [6 points] List one major advantage of each type of state encoding below.

- One-hot encoding
- Binary encoding
- Output encoding

(d) [10 points] Fully describe the FSM with equations given that the states are encoded with **one-hot** encoding. Assign state encodings such that numerical values of states increase monotonically for states A through D while using the **minimum** possible number of bits to represent the states with one-hot encoding. Indicate the values you assign to each state and simplify all equations:

- (e) [10 points] Fully describe the FSM with equations given that the states are encoded with **binary** encoding. Assign state encodings such that numerical values of states increase monotonically for states A through D while using the **minimum** possible number of bits to represent the states with binary encoding. Indicate the values you assign to each state and simplify all equations:

- (f) [10 points] Fully describe the FSM with equations given that the states are encoded with **output** encoding. Use the **minimum** possible number of bits to represent the states with output encoding. Indicate the values you assign to each state and simplify all equations:

(g) [10 points] Assume the following conditions:

- We can only implement our FSM with 2-input AND gates, 2-input OR gates, and D flip-flops.
- 2-input AND gates and 2-input OR gates occupy the *same* area.
- D flip-flops occupy 3x the area of 2-input AND gates.

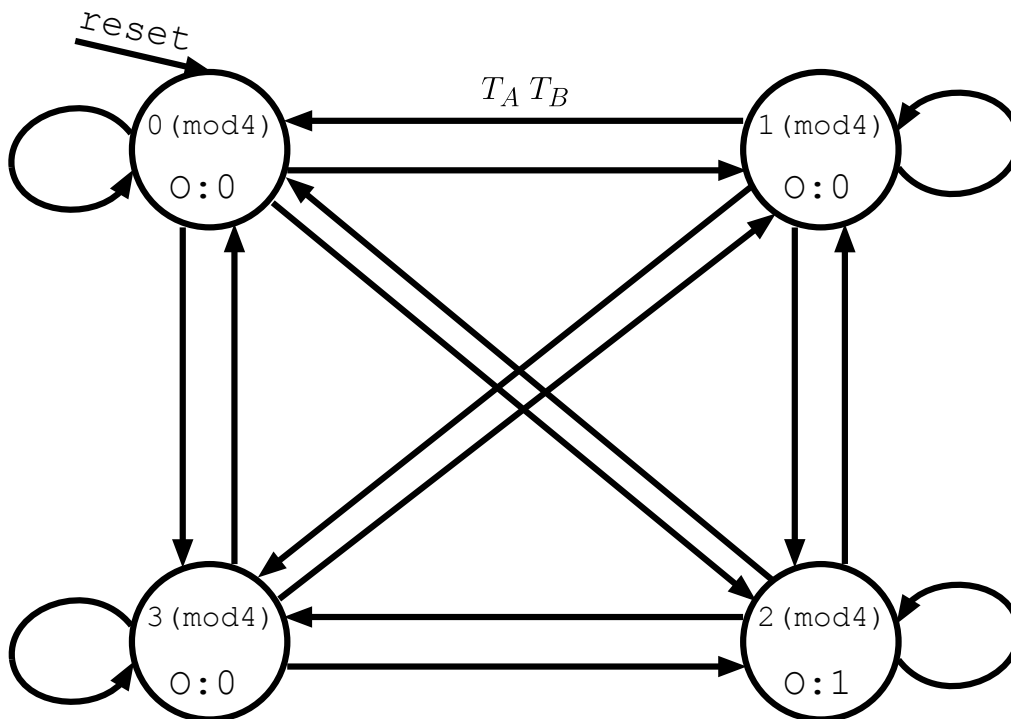
Which state-encoding do you choose to implement in order to minimize the total area of this FSM?

5 5 Finite State Machines (III)

You are given two *one-bit* input signals (T_A and T_B) and one *one-bit* output signal (O) for the following modular equation: $2N(T_A) + N(T_B) \equiv 2 \pmod{4}$. In this modular equation, $N(T_A)$ and $N(T_B)$ represent the **total number of times** the inputs T_A and T_B are high (i.e., logic 1) at each positive clock edge, respectively. The one-bit output signal, O , is set to 1 when the modular equation is satisfied (i.e., $2N(T_A) + N(T_B) \equiv 2 \pmod{4}$), and 0 otherwise. An example that sets $O = 1$ at the end of the fourth cycle would be:

- (1st cycle) $T_A = 0$ ($N(T_A) = 0$), $T_B = 0$ ($N(T_B) = 0$), $2N(T_A) + N(T_B) \equiv 0 \pmod{4} \Rightarrow O = 0$
- (2nd cycle) $T_A = 1$ ($N(T_A) = 1$), $T_B = 1$ ($N(T_B) = 1$), $2N(T_A) + N(T_B) \equiv 3 \pmod{4} \Rightarrow O = 0$
- (3rd cycle) $T_A = 1$ ($N(T_A) = 2$), $T_B = 0$ ($N(T_B) = 1$), $2N(T_A) + N(T_B) \equiv 1 \pmod{4} \Rightarrow O = 0$
- (4th cycle) $T_A = 0$ ($N(T_A) = 2$), $T_B = 1$ ($N(T_B) = 2$), $2N(T_A) + N(T_B) \equiv 2 \pmod{4} \Rightarrow O = 1$

- (a) You are given a partial **Moore** machine state transition diagram that corresponds to the modular equation described above. However, the input labels of most of the transitions are still missing in this diagram. Please label the transitions with the correct inputs so that the FSM correctly implements the above specification.



- (b) Describe the FSM with Boolean equations assuming that the states are encoded with **one-hot encoding**. Assign state encodings while using the **minimum** possible number of bits to represent the states. Please indicate the values you assign to each state.

- (c) Describe the FSM with Boolean equations assuming that the states are encoded with **binary encoding** (i.e., fully encoding). Assign state encodings while using the **minimum** possible number of bits to represent the states. Please indicate the values you assign to each state.

- (d) Consider an implementation of the FSM assuming that the states are encoded with **output encoding**. What is the **minimum** number of bits required to encode the states with output encoding?