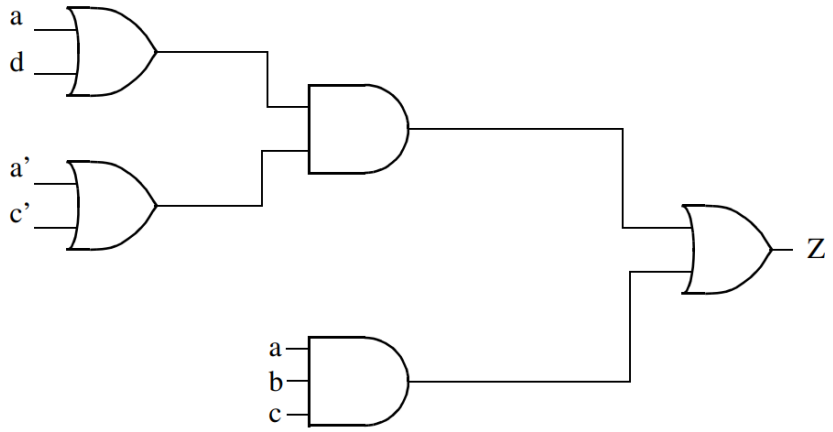
Problem 1 Retry:

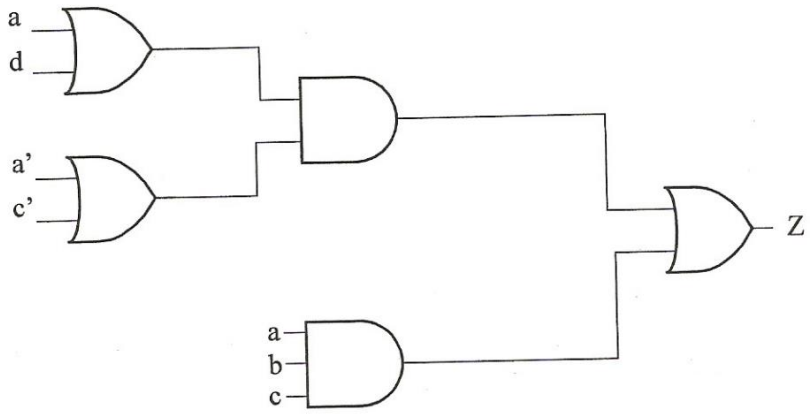
1.

For the network shown below, find any/all static 1-hazards. For any 1-hazard found specify the conditions which will cause the hazard to appear at the output (i.e. specify the logic values of the variables which are constant and the variable which is changing)



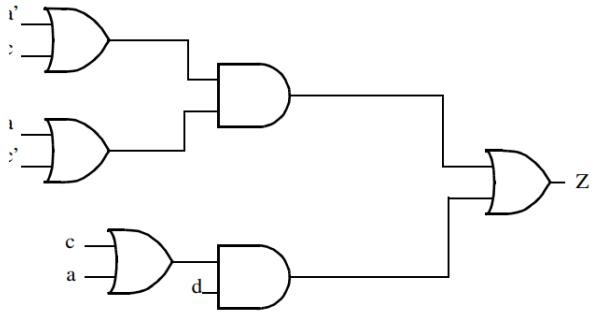
2.

For the network shown below, find any/all static 0-hazards. For any 0-hazard found specify the conditions which will cause the hazard to appear at the output (i.e. specify the logic values of the variables which are constant and the variable which is changing) [10 points].



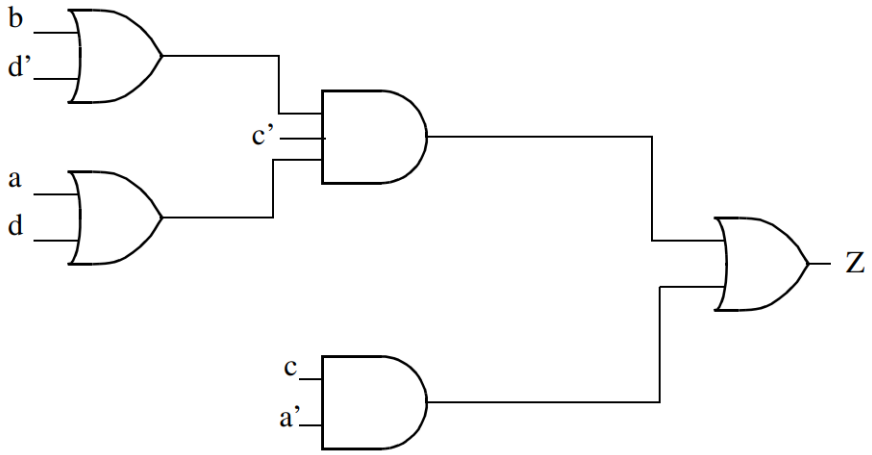
3.

For the network shown below, using the k-map methods outline in the class, find all static 0-hazards (or state that there are no static 0-hazards present). For any 0-hazard found specify the conditions which will cause the hazard to appear at the output (i.e. specify the logic values of the variables which are constant and the variable which is changing).



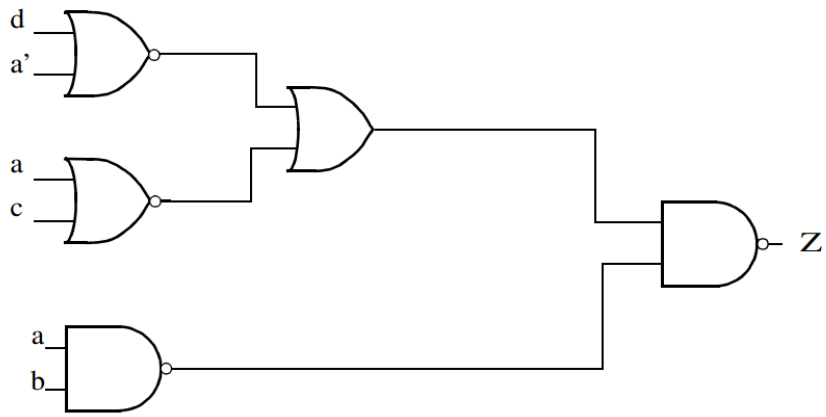
4.

For the network shown below, find all static 0-hazards. For each 0-hazard found specify the conditions which will cause the hazard to appear at the output (i.e. specify the logic values of the variables which are constant and clearly specify the variable that is assumed to be changing). If there are no 0-hazards found, use a K'map to show why this is the case.



5.

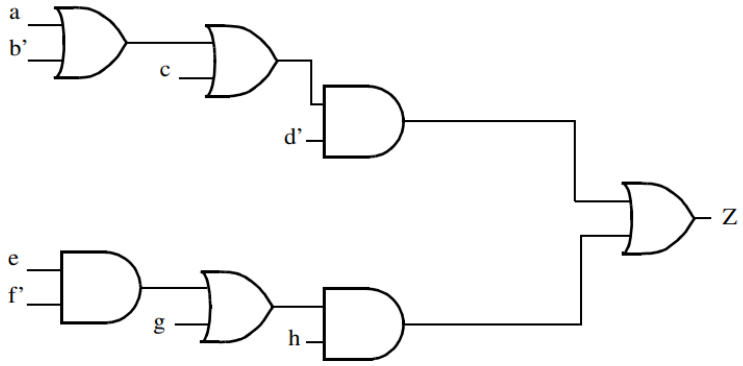
For the network shown below, find any/all static 1-hazards. For any 1-hazard found specify the conditions which will cause the hazard to appear at the output (i.e. specify the logic values of the variables which are constant and the variable which is changing).



Problem 3 Retry:

1.

Draw an equivalent multi-level network that is composed entirely of multiple two-input **NAND** gates. Assume that all input variables (a through h) are available in their true and complemented form.



2.

Use a single rising edge triggered JK Flip-Flop with no additional components (other than connecting wires) to create a rising edge triggered T flip flop (see Fig. 1-13 of the textbook for an example).

3.

A 2-bit Gray Code accumulator counts 00=>01=>11=>10=>00 and continues repeatedly. Using 2 rising-edge triggered DQ Flip Flops and a single 2-input XOR gate, draw a circuit that continually generates this pattern.

4.

A 3-bit down-counter counts 111=>110=>101=>100=>011=>010=>001=>000=>111 and continues repeatedly. By first creating a truth table or K-map to determine next state (based on current state as the input), create a 3-bit down counter circuit with the use of only NAND gates and 3 rising-edge triggered Flip Flops.

5.

A ring oscillator is a circuit with an odd number of inverters attached to each other in a daisy chain loop (output of the final inverter is tied to the input of the first inverter). Draw a ring oscillator comprising 9 inverters in one daisy chain feedback loop. If each inverter has an inertial delay of exactly 200 picoseconds, and neglecting propagation delay on the wires between inverters (i.e. 0 additional delay), at what frequency will the oscillator operate (in MHz)? If a 10^{th} inverter is added to the loop, how does the circuit behave?

Problem 4 Retry:

1. Write a short Verilog description of a transparent D Latch with D and G inputs and Q output.

module DLATCH (**input** D, G, **output** Q)

2. Write a short Verilog description of a rising-edge triggered T Flip-Flop (see Fig. 1-13 on pp. 16 of the textbook for definition), with both Q and QN outputs (QN is the inverse of Q). In addition to its normal T and CLK inputs, add an active-low asynchronous SETN input that sets Q high when SETN is low (regardless of clock edge).

module TFF (**input** T, CLK, SETN, **output** Q, QN)

3. Write a short Verilog description of a falling-edge triggered JK Flip-Flop, with J and K inputs, and an active-high asynchronous clear input CLR that sets the output Q immediately to 0.

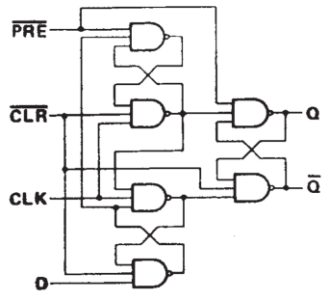
module JKFF (**input** J, K, CLK, CLR, **output** Q)

4. The following truth table and circuit diagram is from the Texas Instruments SN7474 datasheet, showing one basic design for a DQ flip-flop with active-low preset (PREN) and active-low clear (CLRN). Note that all gates drawn are NAND gates!

FUNCTION TABLE

INPUTS				OUTPUTS	
$\overline{PRE}$	$\overline{CLR}$	CLK	D	Q	$\overline{Q}$
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H†	H†
H	H	↑	H	H	L
H	H	↑	L	L	H
H	H	L	X	Q_0	$\overline{Q}_0$

logic diagram (positive logic)

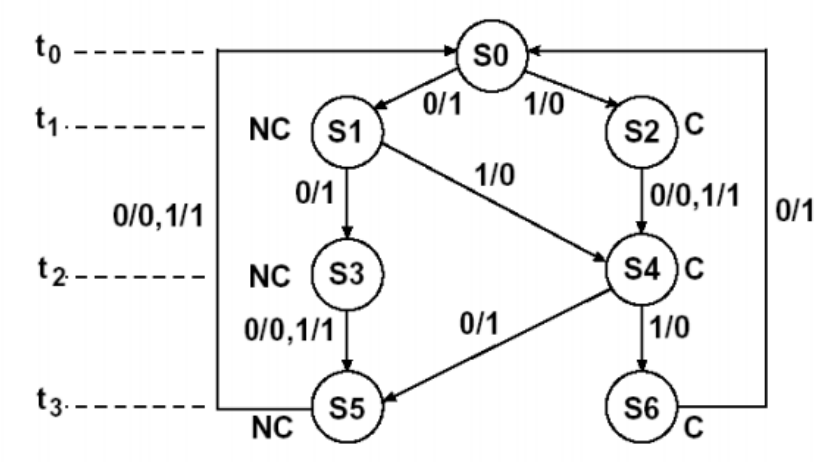


a) Are PREN and CLRN asynchronous or synchronous inputs?
b) Write a short Verilog description of the SN7474 circuit as shown in the diagram above ***USING ONLY assign OPERATORS (NO always @(…) procedures)***

module SN7474 (**input** PREN, CLRN, CLK, D, **output** Q, QN)

Problem 5 Retry:

1. Using two always @() procedures, write a Verilog model of the following state graph shown below. For state encoding, use 3'd0 for S0, 3'd1 for S1, … 3'd6 for S6.



module fsm_prob5_1 (**input** X, CLK, **output** Z) **begin**

reg [2:0] state;

reg [2:0] next_state;

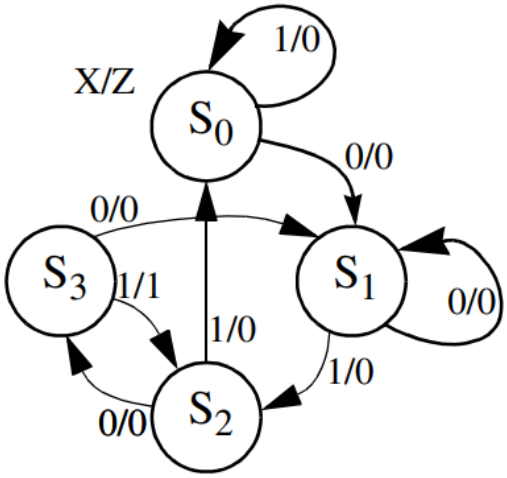
always @(X, state**) begin**

end

always @(posedge CLK) **begin**

end

2. Using two always @() procedures, write a Verilog model of the following state graph shown below. For state encoding, use 2'd0 for S0, 2'd1 for S1, 2'd2 for S2, and 2'd3 for S3.



module fsm_prob5_2 (**input** X, CLK, **output** Z) **begin**

reg [1:0] state;

reg [1:0] next_state;

always @(X, state**) begin**

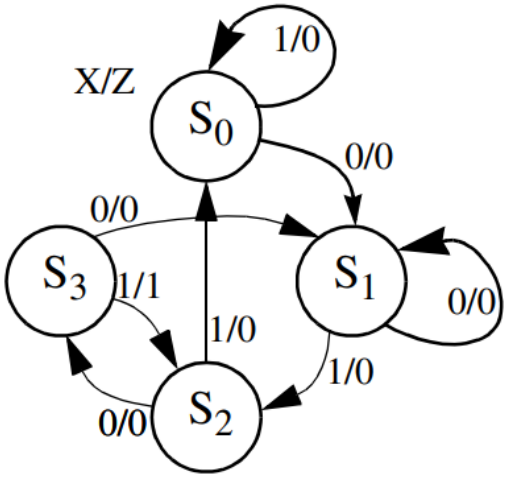
end

always @(posedge CLK) **begin**

end

end

2. Using two always @() procedures, write a Verilog model of the following state graph shown below. For state encoding, use 2'd0 for S0, 2'd1 for S1, 2'd2 for S2, and 2'd3 for S3.



module fsm_prob5_2 (**input** X, CLK, **output** Z) **begin**

reg [1:0] state;

reg [1:0] next_state;

always @(X, state**) begin**

end

always @(posedge CLK) **begin**

end

end

3. Create a valid Verilog model for a 3-to-8 decoder, with a 3-bit input SEL and an 8-bit output ONEHOT. The 3-bit input SEL enables a 1 on the output bit in the ONEHOT output vector at bit position SEL, with all other bits set to 0.

module three2eight_dec (**input** [2:0] SEL, **output** [7:0]ONEHOT) **begin**

end

4. Create a valid Verilog 5-bit priority encoder, where the highest bit position on the 5-bit input VEC[4:0] that contains a 1 is identified as the 3-bit output SEL[2:0]. For example, under the input condition VEC[4:0] = 5'b0**1**011, the SEL output would be 3'd3 (since bit position 3 is the highest one containing a 1).

module prio_enc (**input** [4:0] VEC, **output** [2:0] SEL) **begin**

end

5. Write a valid Verilog model for a 8-bit input, 1-bit output parallel to serial shift register with the following requirements:

The CLK input is a rising edge trigger for all the registers in the design.

This shift register has a P_IN parallel data input bus, an LD input that loads the P_IN data into 8-bits of registers (in other words, DQ flip-flops) contained in this module.

The serial output data S_OUT must contain the least significant bit (LSB) of the newly loaded parallel word on every clock cycle that immediately follows LD being high.

When the SHIFT input is high (and the LD input is also low), the contents of the 8 shift registers shift from MSB towards LSB, causing the S_OUT data to update every cycle. E.g. If LD is high, followed by 7 clock cycles of SHIFT being high (while LD is low), all 8 bits of P_IN that were loaded upon the LD cycle will appear on S_OUT in the order of LSB first and MSB last:

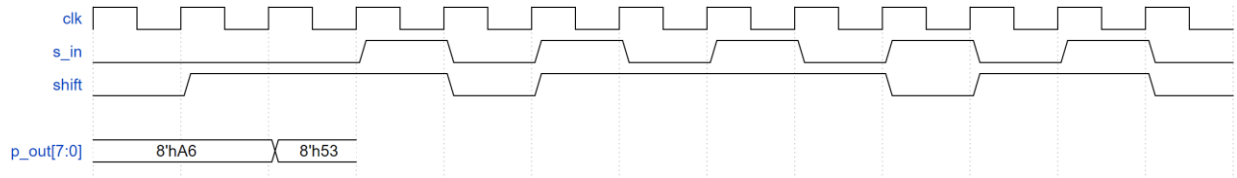
module serializer (**input** [7:0] P_IN, CLK, LD, SHIFT, **output** S_OUT) **begin**

end

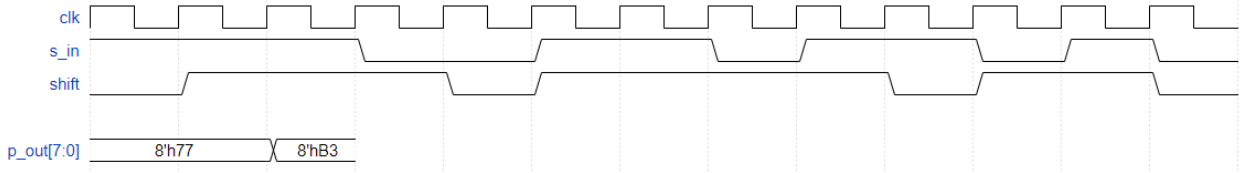
Problem 6 Retry:

INSTRUCTIONS: In this problem set, all state machines are intended to behave as Moore state machines; the p_out[7:0] and accum[7:0] outputs must change in value one clock cycle after the inputs change, as shown in the first 3 provided values in the timing diagrams.

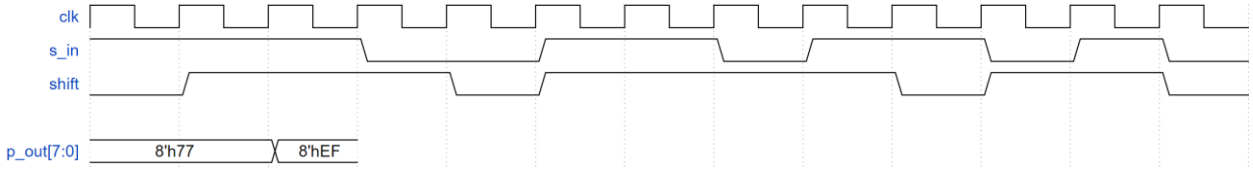
1. Complete the following timing diagram of a serial-in/parallel-out shift register, where serial data shifts in bit-by-bit from the **s_in** input into the most-significant bit side (bit 7; newest/most-recent) towards the least-significant bit side (bit 0; oldest/least-recent). Data only shifts when the **shift** input is high; when shift is low **p_out**[7:0] does not change in value. You can either complete the **p_out**[7:0] signal values in hexadecimal or binary.



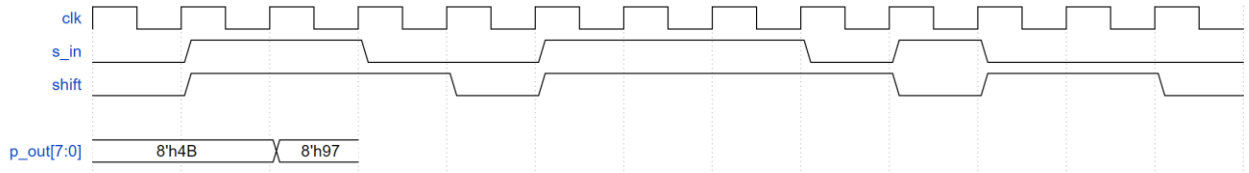
2. Complete the following timing diagram of a serial-in/parallel-out shift register, where serial data shifts in bit-by-bit from the **s_in** input into the most-significant bit side (bit 7; newest/most-recent) towards the least-significant bit side (bit 0; oldest/least-recent). Data only shifts when the **shift** input is high; when shift is low **p_out**[7:0] does not change in value. You can either complete the **p_out**[7:0] signal values in hexadecimal or binary.



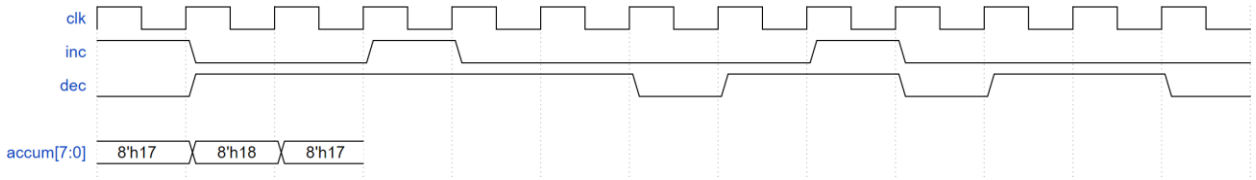
3. **NOTE: THIS PROBLEM STATEMENT DIFFERS FROM #1 & #2…** Complete the following timing diagram of a serial-in/parallel-out shift register, where serial data shifts in bit-by-bit from the **s_in** input into the ***least-significant*** bit side (bit 0; newest/most-recent) towards the ***most-significant*** bit side (bit 0; oldest/least-recent). Data only shifts when the **shift** input is high; when shift is low **p_out**[7:0] does not change in value. You can either complete the **p_out**[7:0] signal values in hexadecimal or binary.



4. **NOTE: THIS PROBLEM STATEMENT DIFFERS FROM #1 & #2…** Complete the following timing diagram of a serial-in/parallel-out shift register, where serial data shifts in bit-by-bit from the **s_in** input into the ***least-significant*** bit side (bit 0; newest/most-recent) towards the ***most-significant*** bit side (bit 0; oldest/least-recent). Data only shifts when the **shift** input is high; when shift is low **p_out**[7:0] does not change in value. You can either complete the **p_out**[7:0] signal values in hexadecimal or binary.



5. **NOTE: THIS PROBLEM DIFFERS FROM PROBLEMS 1-4:**
Complete the following timing diagram of an 8-bit accumulator with 1-bit increment (**inc**) and 1-bit decrement (**dec**) inputs. The output **accum**[7:0] increases by 1 when *only* the **inc** input is high, and decreases by 2 when *only* the **dec** input is high. If neither is high or if both are high, the value of **accum**[7:0] remains the same.

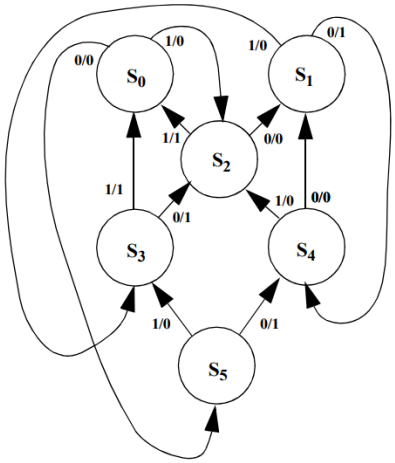


Problem 7 Retry:

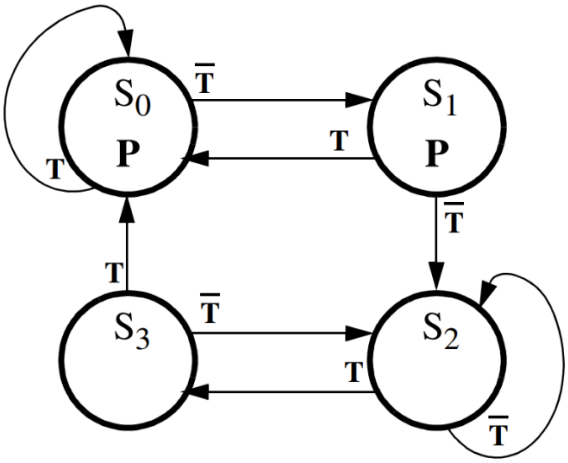
INSTRUCTIOS: FOR THE FOLLOWING 4 STATE MACHINES, DO THE FOLLOWING:

1) **Create an equivalent Algorithmic State Chart representation, using rectangular boxes to show states, diamonds to show input/decision blocks, and ovals to represent conditional outputs. See the Module/Slides for Algorithmic State Machine Representation for examples.**
2) **Write a behavioral Verilog HDL model that implements the state transition graph. In your model include a synchronous active high reset input signal that will always place the design in state S0 on the active edge of the next clock pulse regardless of the current state of the network.**

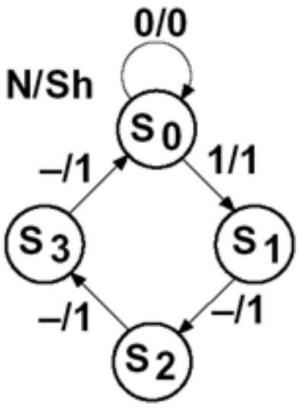
A:



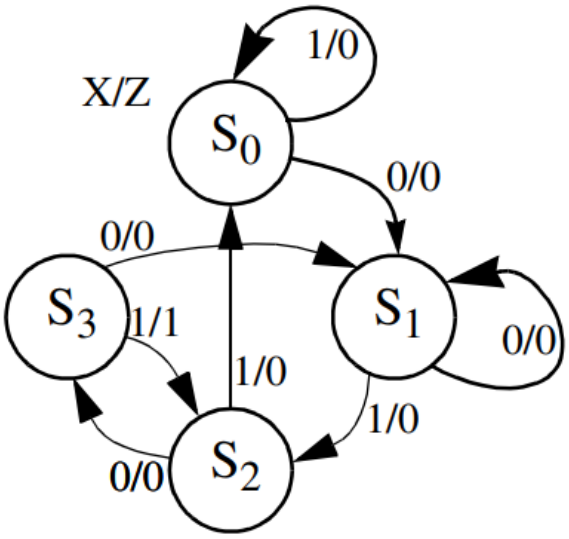
B:



C:



D:



Problem 8 Retry: Reduce the following state tables to the minimum number of states:

1.

present state	next state X=0	1	present output X=0	1
a	d	f	1	0
b	e	c	1	0
c	a	g	0	1
d	e	c	0	1
e	a	g	1	1
f	e	c	0	1
g	d	f	1	0

2.

present state	next state X=0	1	present output
S_0	S_5	S_7	0
S_1	S_3	S_2	1
S_2	S_4	S_3	0
S_3	S_1	S_4	1
S_4	S_2	S_1	0
S_5	S_5	S_0	1
S_6	S_0	S_7	0
S_7	S_1	S_6	1

3.

Present State	**Next State**		**Output**	
	X=0	**X=1**	**X=0**	**X=1**
$\mathbf{S_0}$	S_4	S_1	1	0
$\mathbf{S_1}$	S_6	S_5	1	0
$\mathbf{S_2}$	S_4	S_1	1	0
$\mathbf{S_3}$	S_4	S_1	0	1
$\mathbf{S_4}$	S_6	S_5	1	1
$\mathbf{S_5}$	S_0	S_2	0	1
$\mathbf{S_6}$	S_0	S_2	0	1

4.

Current State	Next State		Output	
	X=0	X=1	X=0	X=1
S_0	S_5	S_2	0	0
S_1	S_4	S_3	1	0
S_2	S_1	S_0	0	1
S_3	S_2	S_0	1	1
S_4	S_1	S_2	0	0
S_5	S_4	S_3	1	0

5.

Current State	Next State		Output, Z	
	X=0	X=1	X=0	X=1
S_0	S_1	S_2	1	0
S_1	S_5	S_3	0	0
S_2	S_3	S_4	1	1
S_3	S_5	S_4	0	1
S_4	S_0	S_3	0	0
S_5	S_1	S_2	1	0

Problem 10 Retry:

1. Develop a Verilog model for a 4-bit barrel shifter, which produces a 4-bit output that performs "Rotate Shift Left" or "Circular Shift Left". The 4-bit input vector shall be named I[3:0], the number of bit positions to shift is SH[1:0], and the output is O[3:0].

Note that a barrel shifter (or rotate/circular shift operation) shifts any MSbits back around to the LSbit position. For example, if I[3:0]=4'b0111 and SH[1:0]=3, then O[3:0]=4'b1011.

module BSHIFT (**input** [3:0] I, **input** [1:0] SH, **output** [3:0] O)

2. Develop a Verilog model for a 6-to-1 multiplexer where the general inputs represent a 6-bit bus and the output is a single signal. When S[2:0] is 6, choose I[2] as the output, and when S[2:0] is 7, choose I[3] as the output. If using a case statement, please use a default clause; if using an if/else statement, assign O in all clauses including a final "else" clause to avoid creating a latch. Label the inputs I[5:0], S[2:0], and the output O.

module MUX6TO1 (**input** [5:0] I, [2:0] S, **output** O)

3. Develop a Verilog model for a 2-bit synchronous static random access memory with the following specifications:

Inputs: CLK, WDAT, WE, WA, RA

Outputs: RDAT

Storage: Two Flip-Flops, named RAM

a) Whenever WE is high at the rising edge of CLK, one bit of data (WDAT) is written to the flip-flop in RAM at the address specified by the WA input. I.e. RAM[0] is assigned the value WD when WE is high and WA is 0, and RAM[1] is assigned the value WD when WE and WA are both high.
b) The output RDAT is always immediately assigned the value of the bit of RAM selected by RA. You may either read RDAT out with one cycle of delay or with zero delay (i.e. with or without an extra flip-flop in the path after the RAM flip-flops are selected down by RA).

module SRAM2 (**input** CLK, WDAT, WE, WA, RA, **output** RD)

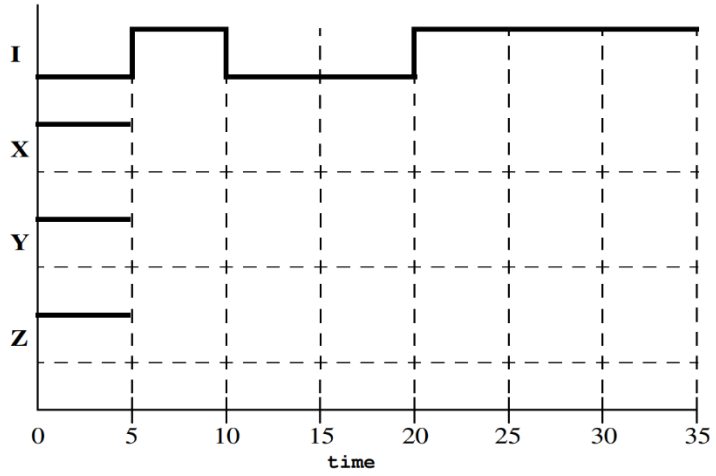
reg [1:0] RAM;

Problem 11 Retry (first part only):

1.

Complete the Timing Diagram for the simulation of the following Verilog Model where input I is driven in the manner that is shown on the diagram.

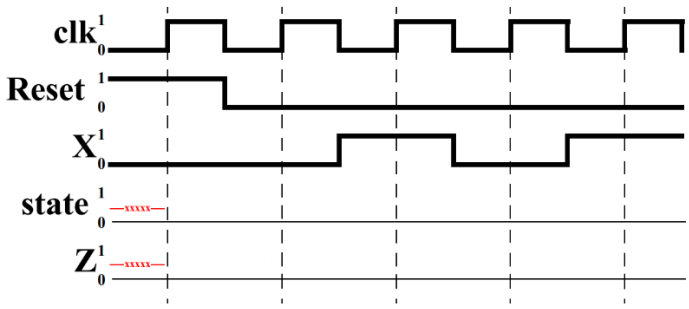
```
module v_model(input I, output X,Y, output reg Z);
   assign #10 X = ~I;
   not #(10) G1 (Y,I);
   always @(I) Z <= #10 ~I;
endmodule;
```



2.

Complete the following timing diagram for the output signal, **Z** and the internal input signal, **state**. Assume that a basic functional RTL Simulation is to be performed.

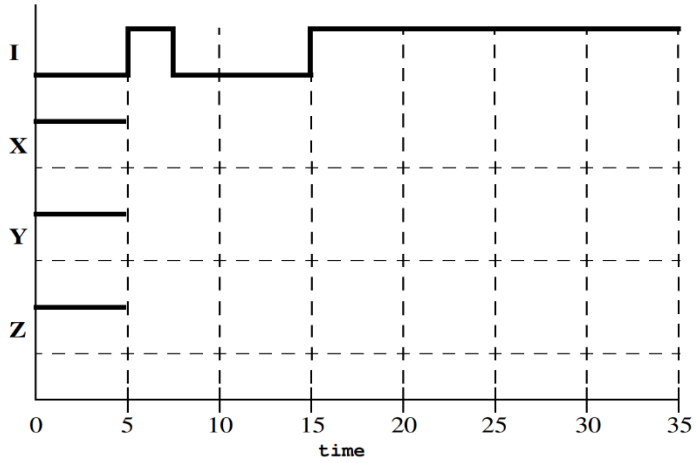
```
module fsm_network (input clk, X, Reset, output reg Z);
    reg state, next_state;

    always @ (state, X)
        case (state)
            0 : if (X) begin Z=1; next_state=1; end
                else  begin Z=0; next_state=0; end

            1 : if (X) begin Z=0; next_state=1; end
                else  begin Z=1; next_state=1; end
        endcase

    always @ (posedge clk)
        if (Reset) state=0;
        else state=next_state;

endmodule
```



3.

Complete the Timing Diagram for the simulation of the following Verilog Model where input I is driven in the manner that is shown on the diagram.

```
module v_model(input I, output reg X, output Y,Z);
   always @(I) X <= #5 ~I;
   not #(5) G1 (Y,X);
   assign #5 Z = ~Y;
endmodule;
```



4.

Complete the following timing diagram for signals A, B, and Z for the network shown below assuming that all setup and hold times for the flip-flops have been met and all propagation delays through the gates and flip-flops are negligible (i.e. zero). Also assume that the two flip-flops are clocked on the falling edge of the system clock and are in the reset state (i.e. A='0' and B='0') at the beginning of the timing diagram.