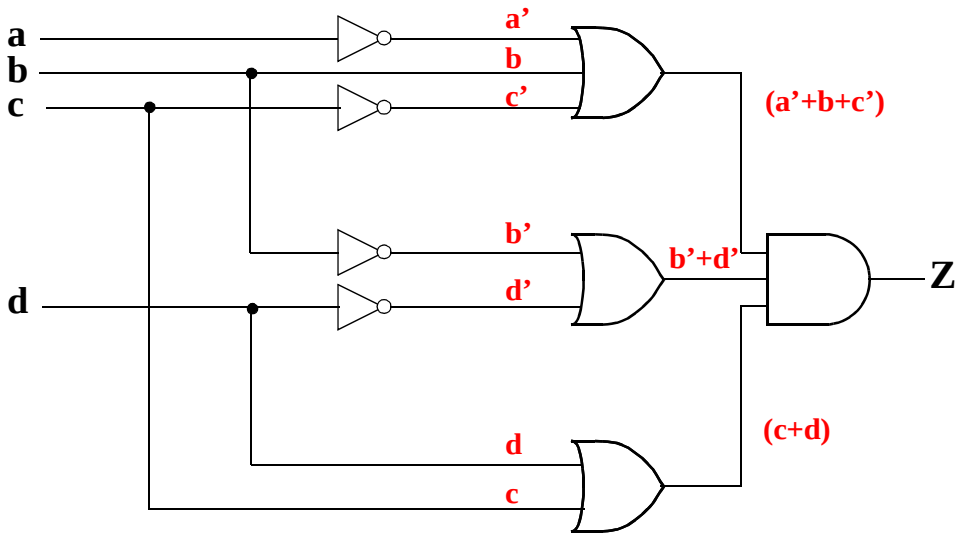
CPE 322 EXAM I Name [Solution]

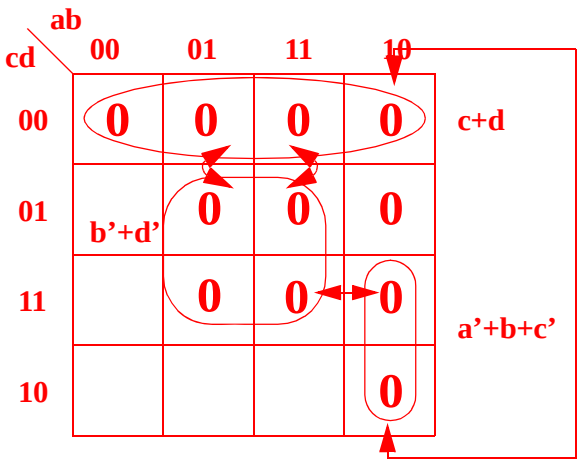
Fall Semester 2015

Work should be performed systematically and neatly with the final answer being underlined. This exam is closed book, closed notes, closed neighbor. Allowable items on desk include: exam, pencils, and pens. All other items must be removed from student's desk. Students have approximately 90 minutes (1 1/2 hours) to complete this exam. Best wishes!

1. [12.5 points] 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.



Z = (a'+b+c')(b'+d')(c+d) already in desired 0-hazard form



hazard #1
a=1, b=0, c=1, and d changes from 0 to 1 or 1 to 0

hazard #2
a=0, b=1, c=0, and d changes from 0 to 1 or 1 to 0

hazard #3
a=1, b=1, c=0, and d changes from 0 to 1 or 1 to 0

hazard #4
a=1, c=1, d=1, and b changes from 0 to 1 or 1 to 0

2. [10 points] Write a Verilog Module for a Parallel-in, Serial-out 8-bit Shift Register..

```
// The following I/O ports are defined as follows:
//   clk    -- global 1-bit clock signal -- all operations must occur on the falling edge
//   ld     --  load input, when high 8-bit data on PI port is loaded into the shift register
//   shift -- when high (and ld is low) contents are shifted out one bit at a time LSB first
//   pi     -- 8-bit parallel data input
//   q      -- 1-bit serial output
module eight_bit_shift (input clk, ld, shift, input [7:0] pi, output reg q);


   reg [7:0] Q_in; // internal value of shift register

    always @ (negedge clk)
      // if load is true load parallel input into internal buffer
      if (ld) Q_in = pi;

      // if load is false and shift is true then shift data
      // right one bit postion
      else if (shift)
         begin
           q = Q_in[0];
           Q_in = {1'b0,Q_in[7:1]};
         end // output LSB of buffer & Shift

// otherwise nothing is done at current clock cycle
endmodule
```

3. [5 points] Short Answer: What is the major difference between *blocking* and *non-blocking procedural assignment statement* in Verilog.

If a current statement contains a blocking procedural assignment then the next statement will be executed after the execution of the current statement. If a current statement contains a non-blocking procedural assignment then the next statement in effect will be executed at the same time as the current statement. Blocking procedural assignment statements use the '=' operator. Nonblocking procedural assignment statements use the '<=' operator.

Give an example of where the use of blocking procedural assignment statements would be useful.

In cases where an operation needs to be completed before another one is utilized because it needs the result of the previous operation or because its result will be overwritten by the subsequent operation. An example is shown in problem 2 above in the shift clause where

always @(negedge clk)
...
begin q=Q_in[0]; Q_in = {1'b0,Q_in[7:1]}; end

where Q_in[0] needs to be extracted before the entire value of Q_in is overwritten.

Give an example of where non-blocking procedural assignment statement would be useful.

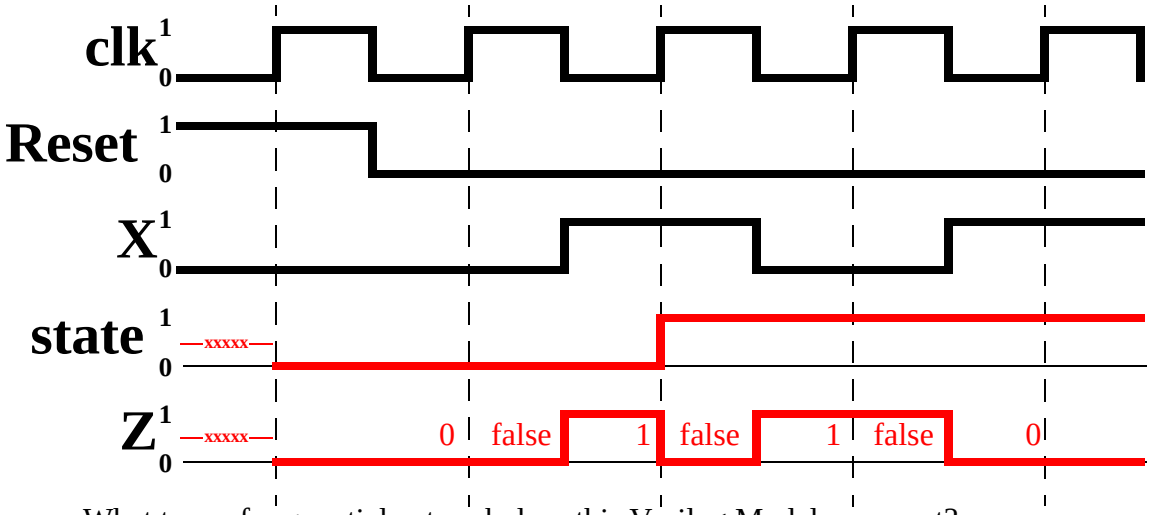
In cases where concurrency are to be modeled using a single procedural block. An example is the modeling of a small three bit shift register where Din is the input and Dout is the output .

reg Q0,Q1;
always @(negedge clk)
...
begin Q1<= Din; Q0<=Q1; Qout <=Q0; **end**

Note that all individual shifting actions occur simultaneously at the same clock pulse. The order of the statements in this case is not important because non-blocking statements are used and updates should occur at the same time using old values.

4. [12.5 points] 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
```

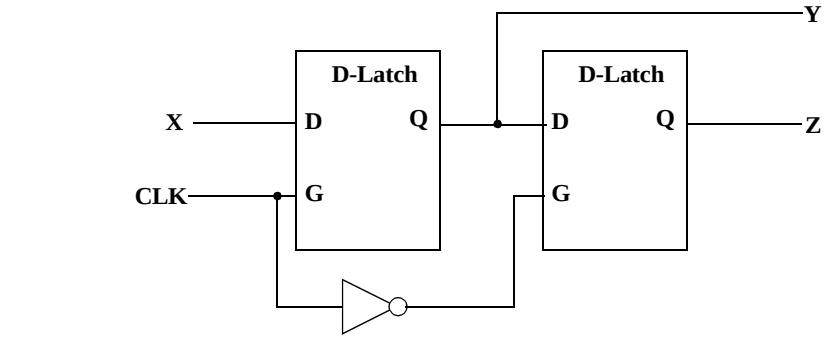


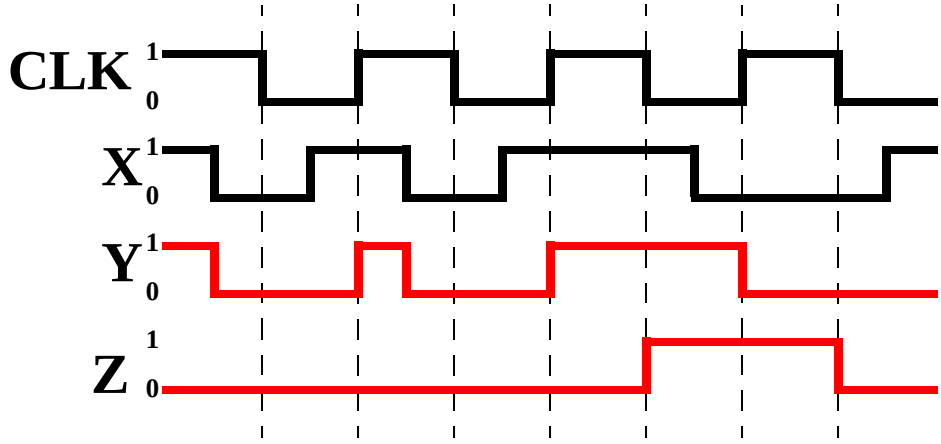
What type of sequential network does this Verilog Model represent?
Mealy FSM Sequential network.

Write down the output sequence for Z.
0 1 1 0

Are there any 'false' outputs on the timing diagram? If so clearly identify them.
Yes, see graph

5.> [10 points] (a) Complete the timing diagram for the following sequential network.





6. [5 points] Create a valid Verilog Model for a 3-to-8 Decoder where the inputs represent a 3 bit bus and the output is an 8-bit bus. Label the input **SEL** and the output **Y**.

```
// 3-to-8 Decoder shift register that loads data eight bits at a time in parallel but
module decoder_3_to_8 (input [2:0] SEL, output reg [7:0] Y);

  always @ (SEL)
    case (SEL)
      0 : Y = “00000001”;
      1:  Y = “00000010”;
      2:  Y = “00000100”;
      3:  Y = “00001000”;
      4:  Y = “00010000”;
      5:  Y = “00100000”;
      6:  Y = “01000000”;
      7:  Y = “10000000”;
      default : Y = “00000000”;
    endcase

endmodule
```

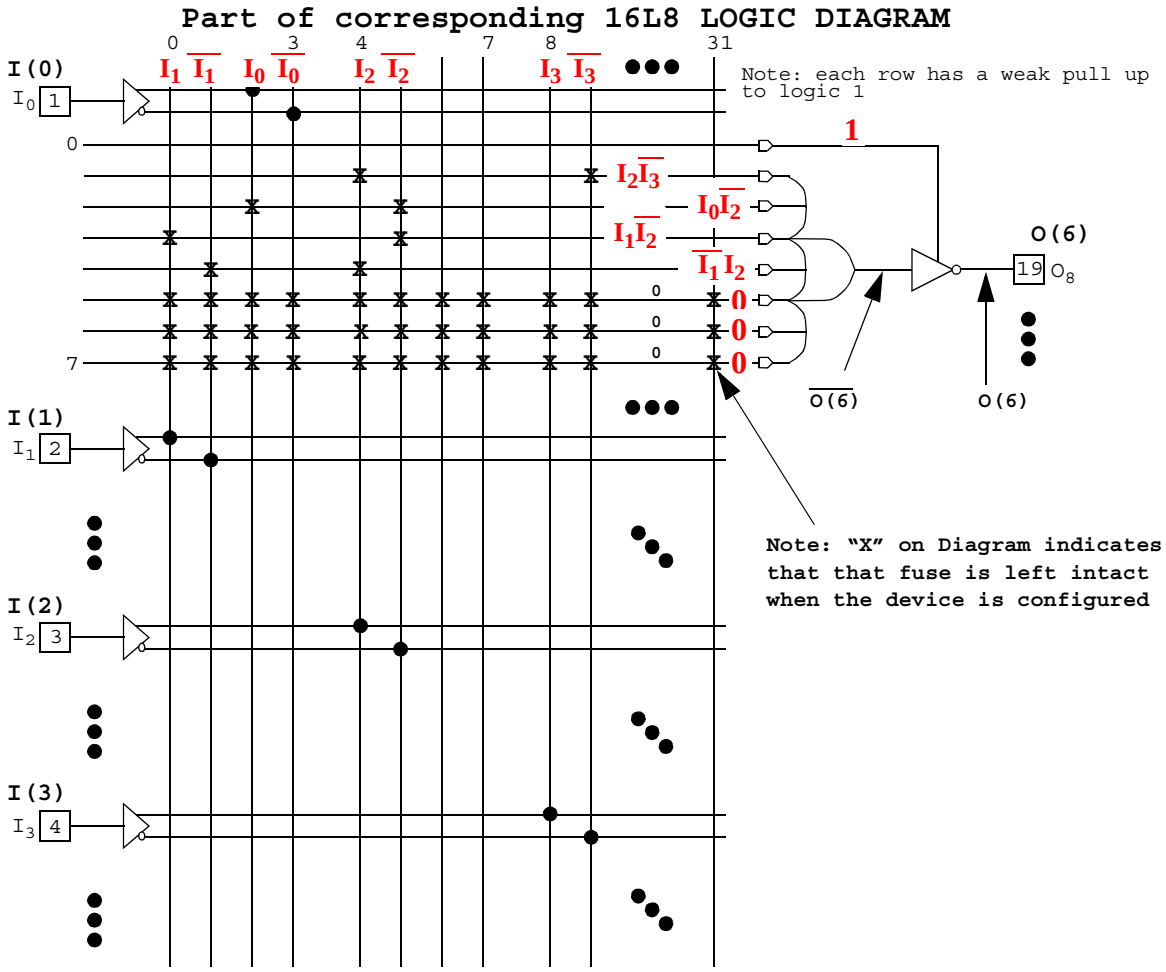
7. [12.5 points] On the 16L8 Programmable Array Logic, PAL, element shown in the figure what are the Boolean Equation for the point labeled $\overline{\mathbf{O6}}$?

$\overline{O6} = I_2\overline{I_3} + I_0\overline{I_2} + I_1\overline{I_2} + I_1\overline{I_2}$

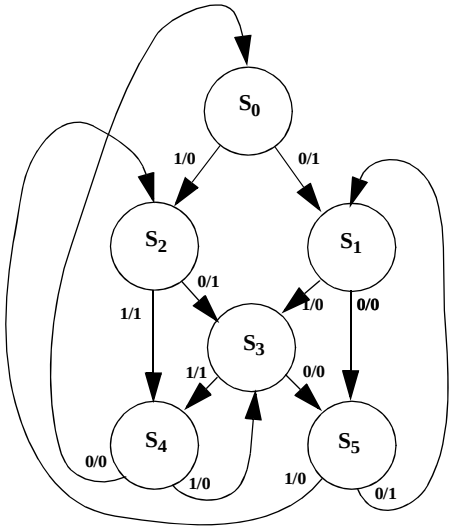
What is the simplified form of the Boolean equation (hint: use Demorgan's Law) for the output point labeled **O6**?

$O6 = (\overline{I_2}+I_3)(\overline{I_0}+I_2)(\overline{I_1}+I_2)(I_1+\overline{I_2})$

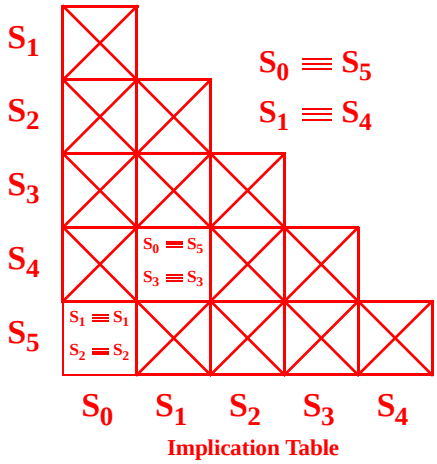
What is the purpose of the inverting tri-state buffer that drives output **O6**? (i.e. why is it there?)
The tri-state buffer is actually a tri-state inverter. It inverts the output when it is active and places the output in a high impedence state which in effect allows the output to be disconnected from the circuit when its select input is low. In this case the select input is always high so it only performs inversion for this configuration of the PAL.



8. [10 points] Reduce the following state graph to a minimum number of states clearing identifying the states that are equivalent with one another. Show the final reduced state graph or table.



Current State	Next State X=0	Next State X=1	Output, Z X=0	Output, Z 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



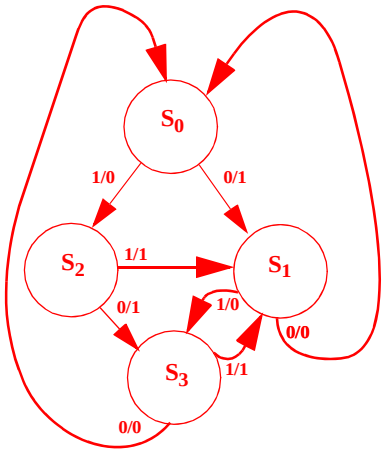
Implication Table

Replacing S_5 with State S_0 and S_4 with State S_1

Reduced State Table

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

Reduced State Graph

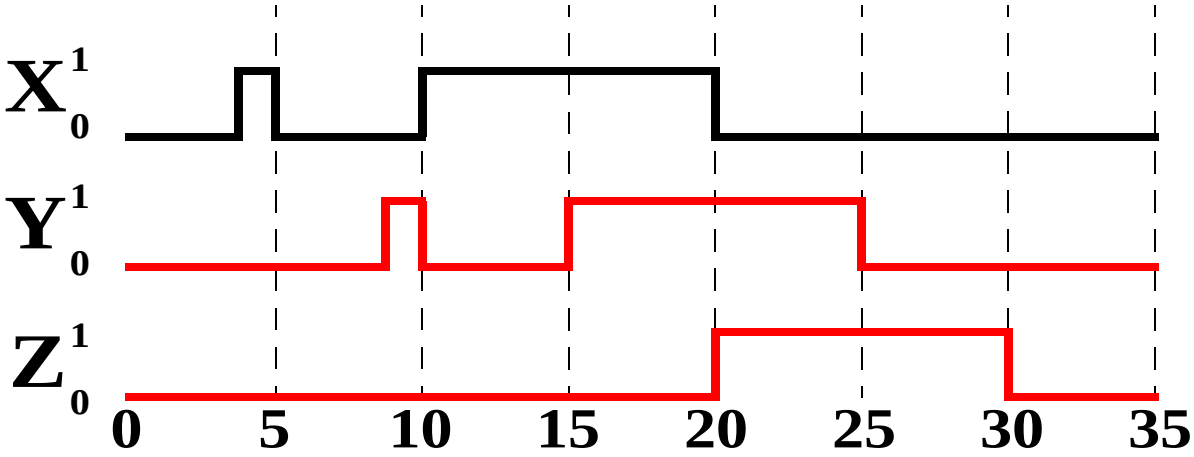


9 [10 points] Complete the timing diagram for the Verilog model for outputs Y and Z, assuming that input X has been stable at a logic 0 for at least 20 ns prior to time 0.

```
`timescale 1ns/100 ps

module delay_test (input X, output  reg Y, output Z);

  assign #5 Z =  Y;

  always @ (X)
    Y <= #5 X;

endmodule
```

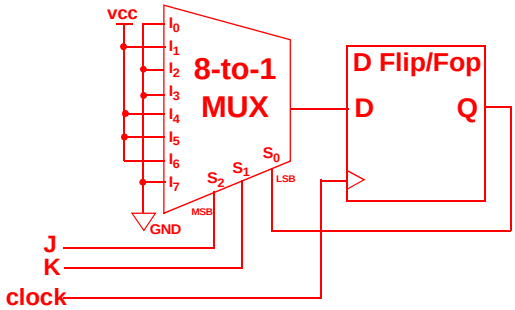
Intertial Delay Model

Transport Delay Model



10 [7.5 points] Use a single 8-to-1 MUX and a rising edge triggered D Flip-flop to create a rising edge triggered J-K Flip/Flop. Assume possitive logic where a connection to VCC will be interpreted as a logic 1 and a connection to GND will be interpreted as a logic 0.

J	K	Q	Q^+=D
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



11 [5 points] [Short Answer]

a) What is the basic definition of combinational logic?
Combinational logic -- Boolean logic where the value of the output(s) are only a function of the current value of its input(s).

b) What is the basic definition of sequential logic?
Sequential logic -- Boolean logic where the value of the outpus(s) are a function of not only the current value of the inputs but also past values of the inputs. Sequential logic has memory.