Q1: H&H 4.2.

Part a

First, we will want to write out a truth table from the if-else logic. We will use X to denote don't care, as in the case of, if a[0] is high, we don't care what any other input is, but we know y will be, y[1:0] = 11. This helps simplify our truth table substantially.

inputs				outputs	
a[0]	a[1]	a[2]	a[3]	y[0]	y[1]
1	X	X	X	1	1
0	1	X	X	0	1
0	0	1	X	1	0
0	0	0	1	0	0
0	0	0	0	0	0

$$2'b\underset{\substack{\uparrow\\ y[1]}}{x}\underset{\substack{\uparrow\\ y[0]}}{x}$$

this hierchy of a[0] → a[3] follows how the if, if-else statement is evaluated.

↳ this final case is the default statement, and in this case a[1:0] is 0,0, which gives our y values.

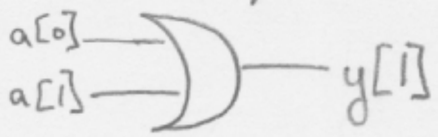
If we had gone into K-maps you could step through to prove the optimal result, but we can readily see it here,

first consider y[1], its simplified truth table is ⟹

a[0]	a[1]	a[2]	a[3]	y[1]
1	X	0	0	1
0	1	X	X	1
0	0	X	X	0

↑ ↑

We see that y[1] is only high if a[0] <u>or</u> a[1] are high, y[1] is only Low if a[0] and a[1] are low.

Therefore,  completely describes the behavior in the Verilog code for y[1].

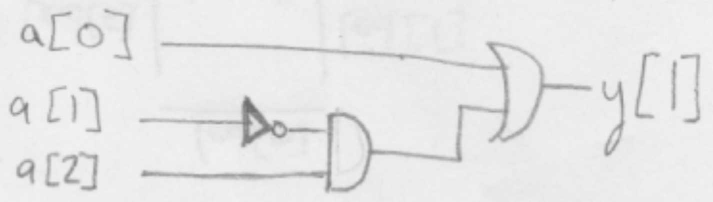
Now, doing the same for y[0],

a[0]	a[1]	a[2]	a[3]	y[0]
1	X	X	X	1
0	1	X	X	0
0	0	1	X	1
0	0	0	X	0

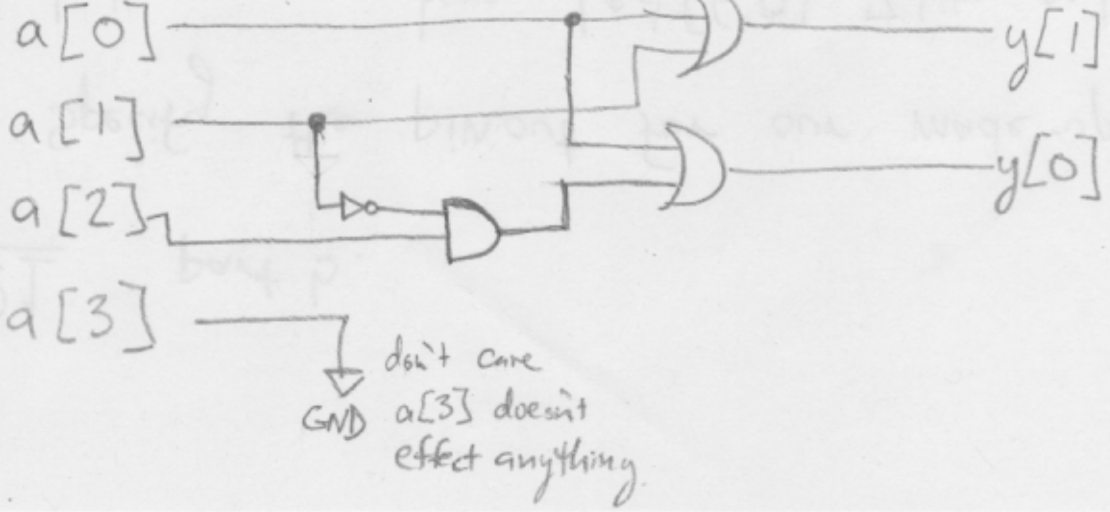
We see y[0] is 1 only when a[0] is high <u>or</u> when a[2] is high <u>and</u> a[1] is low.

We have to specify the and when a[1] is low because if a[1] is high then y[0] should be low and we want to reflect that in our logic. We don't have to specify for a[0] or a[3] in this case because we don't care, as they don't alter the high y[0] when a[2] is high.

Then,



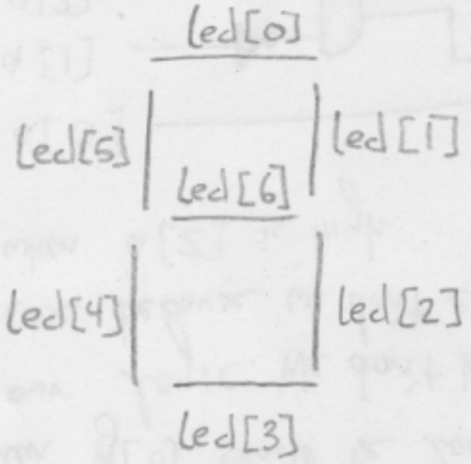
So combining it all gives the verilog if statement,



Q1 part b.

Specify the pinout for our made-up 7 segment display

Let's say we have led[6:0], 7 bits, each corresponding to a single segment, where setting it high turns on the segment, and low turns it off.



Now we assign values in a lookup case statement for our 4-bit input, a[3:0], representing the input 0-F, to make unique representations,

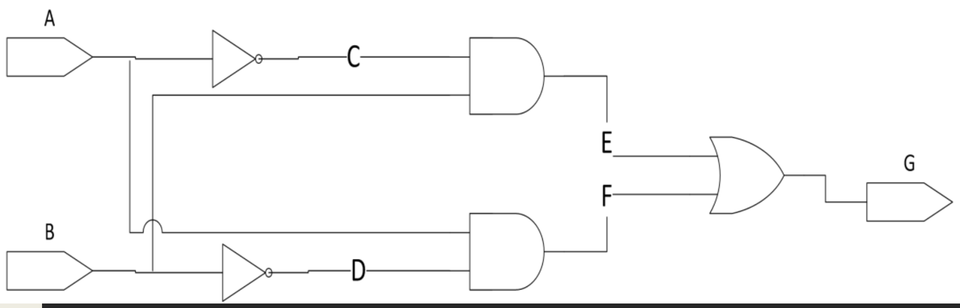
use lower case b, c, d and upper case for A, E, F

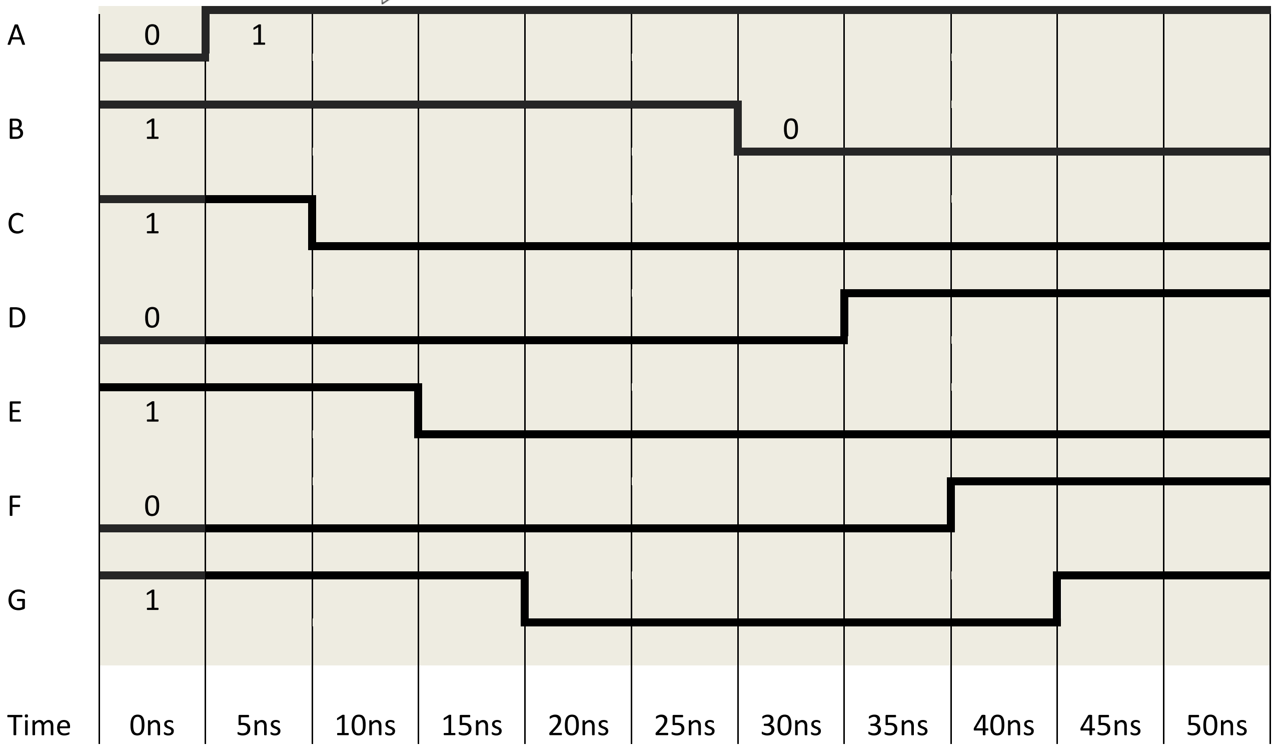
Verilog next page.

Q1b)

```
module display(
       input [3:0]in,
       output [6:0]out);

       //for pins,
       //a[0] is top  a[1] is top right   a[2] is bottom right
       //a[3] is bottom a[4] is bottom left   a[5] is top left
       //a[6] is center

       //b'1 is high for pin
       //b'0 is low

       //Decoder will implement as combo logic
       assign out = (in == 4'h0) ? 7'b011_1111 :
                         (in == 4'h1) ? 7'b000_0110 :
                            (in == 4'h2) ? 7'b101_1011 :
                            (in == 4'h3) ? 7'b100_1111 :
                            (in == 4'h4) ? 7'b110_0110 :
                            (in == 4'h5) ? 7'b110_1101 :
                            (in == 4'h6) ? 7'b111_1101 :
                            (in == 4'h7) ? 7'b000_0111 :
                            (in == 4'h8) ? 7'b111_1111 :
                            (in == 4'h9) ? 7'b110_1111 :
                            (in == 4'hA) ? 7'b111_0111 :
                            (in == 4'hB) ? 7'b111_1100 :
                            (in == 4'hC) ? 7'b010_0111 :
                            (in == 4'hD) ? 7'b101_1110 :
                            (in == 4'hE) ? 7'b111_1001 :
                            (in == 4'hF) ? 7'b111_0011 : 7'b000_0000;

       endmodule
```

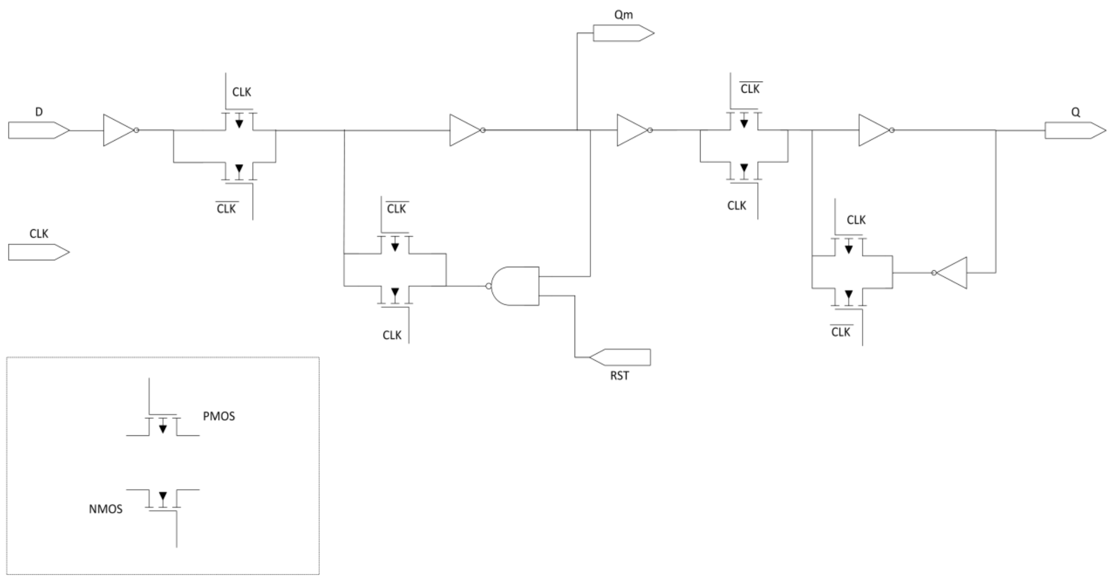
Q2:
For the circuit below, assume that all gates have the same delay (including inverters) of 5 ns, complete the timing diagrams.



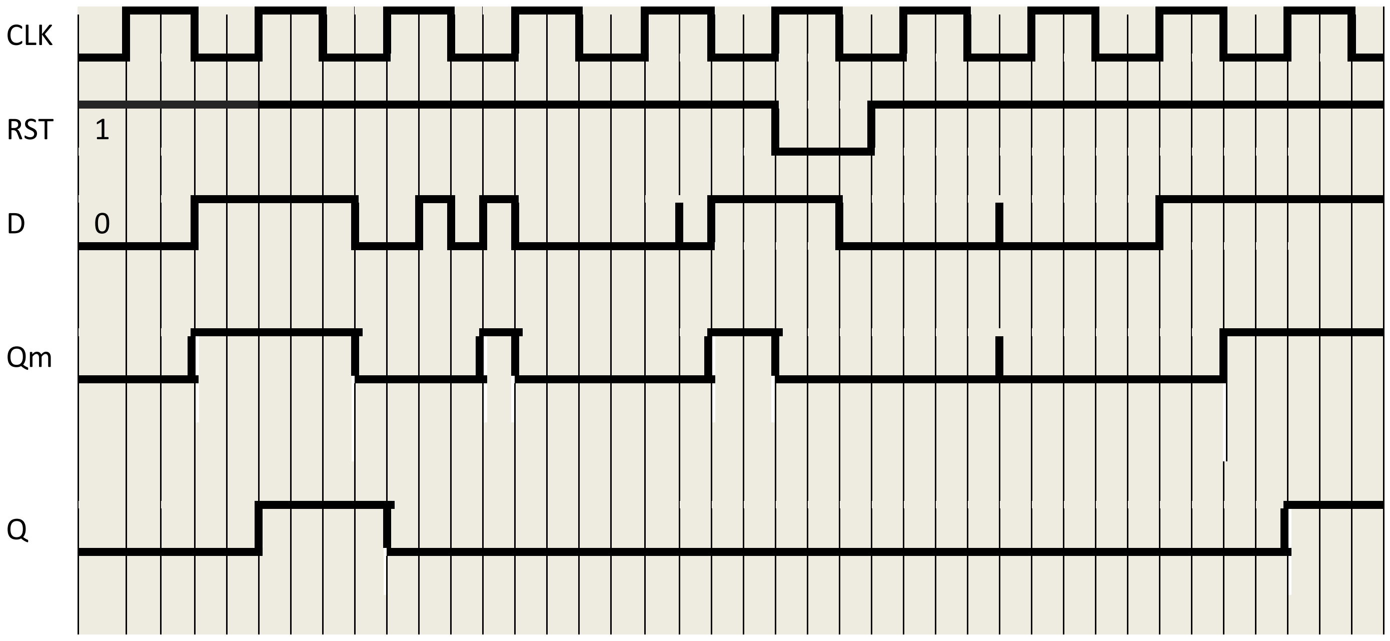


Q3: Complete the timing diagram for the circuit below for 10 clock cycles. Is the reset signal active low or active high? **Active low**

You have learned from class that this circuit is a crucial component in sequential logic design. What is it and why do you think it might be useful? **This is a DFF, a main component in building registers, state machines and ALU.**



Qm
CLK
D
Q
CLK
CLK
CLK
CLK
CLK
CLK
CLK
CLK
CLK
RST
PMOS
NMOS


CLK
RST
D
Qm
Q