EECS 270 Midterm Exam #1

Fall 2009

Name: KEY unique name: KEY

Sign the honor code:

I have neither given nor received aid on this exam nor observed anyone else doing so.

_____

Scores:

Problem #	Points
1	/11
2	/4
3	/8
4	/18
5	/8
6	/15
7	/8
8	/8
9	/20
Total	**/100**

NOTES:

1. Open book and Open notes
2. There are **14** pages total. Count them to be sure you have them all.
3. Calculators are allowed, but no PDAs, Portables, Cell phones, etc.
4. This exam is fairly long: *don't spend too much time on any one problem*.
5. You have about 120 minutes for the exam.
6. Some questions may be more difficult than others. You may want to skip around.
7. **Be sure to show work and explain what you've done when asked to do so.**

1. **Fill in the blank/multiple choice**. **[11 points]**

   a. Convert 10111 as a 5-bit 2's complement number to decimal. -9

   b. Convert 10111 as a 5-bit signed-magnitude number to decimal. -7 .

   c. Convert 11110 as a 5-bit signed-magnitude number to a 6-bit 2's complement number. 110010

   d. When A+!(!A*B*C) is converted into *canonical* sum-of-products form, it has 7 minterms. When converted into *canonical* product-of-sums form it has 1 maxterms.

   e. A truth table with 5 inputs and 3 outputs has 32 rows.

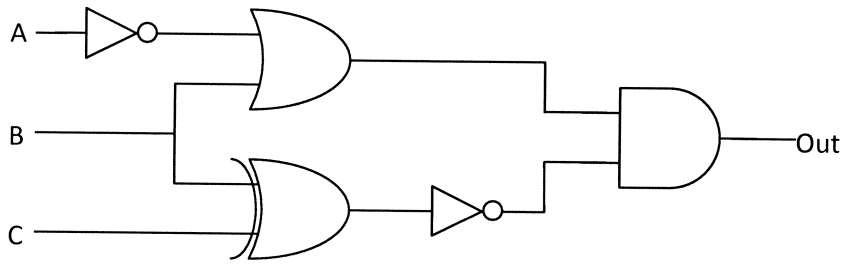
   f. A 32 to 1 multiplexor has 5 selection bits.

   g. !(A⊕B) is the equivalent of **(A*B)+** $\bar{A} \cdot \bar{B}$.

   h. A clock with a 100MHz frequency has a .01 µS period.

   i. The smallest (most negative) decimal number that can be represented by an 8-bit 2's complement number is -128. The largest (most positive) decimal number that can be represented by an 8-bit 2's complement number is 127.

2. **Truth tables**. Fill in the truth table for the output of the following circuit.
[4 points, -1 per wrong/blank box, minimum 0**]**



A	B	C	Out
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

3. **Logic**. Using the rules of logic convert the following into sum-of-products form. Show each step as in example 2.13 on page 52 of our text. **[8]**

a. !(A+B)*!(C*D)

$(\bar{A} \cdot \bar{B}) \cdot (\overline{C \cdot D})$ DeMorgan's law

$(\bar{A} \cdot \bar{B}) \cdot (\bar{C} + \bar{D})$ DeMorgan's law

$\bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}\bar{D}$ distributive prop.

b. !(A*(B+C))+!D

$\bar{A} + \overline{(B+C)} + \bar{D}$ De Morgan's Law

$\bar{A} + \bar{B}\bar{C} + \bar{D}$ De Morgan's Law

c. (!A+B)*!(B⊕C)

$(\bar{A} + B)(\bar{B}\bar{C} + BC)$ definition of XNOR

$(\bar{A} + B)\bar{B}\bar{C} + (\bar{A} + B)BC$ distributive prop.

$\bar{A}\bar{B}\bar{C} + B\bar{B}\bar{C} + (\bar{A}+B)BC$ distributive prop

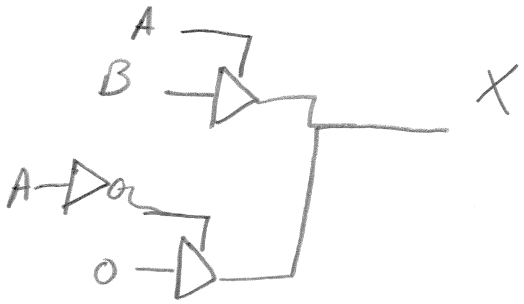
$\bar{A}\bar{B}\bar{C} + (\bar{A}+B)BC$ complement

$\bar{A}\bar{B}\bar{C} + \bar{A}BC + BBC$ distributive prop

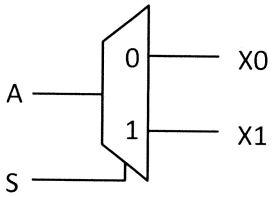
$\bar{A}\bar{B}\bar{C} + \bar{A}BC + BC$ identity

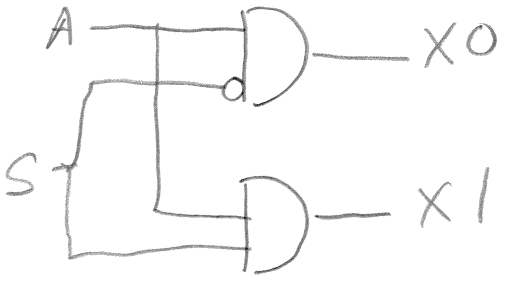
4. **Short design**. In each problem you will be asked to design a given device using a limited number of some set of components. *We will give little, if any, partial credit for these problems.* Draw your answers neatly. **[18 points, 6 points each]**

   a. Using no more than 5 components, build a two-input AND gate using only tri-state buffers and inverters. You can freely connect inputs to "1" and "0" as needed. Label the inputs as "A" and "B". Label the output as "X".

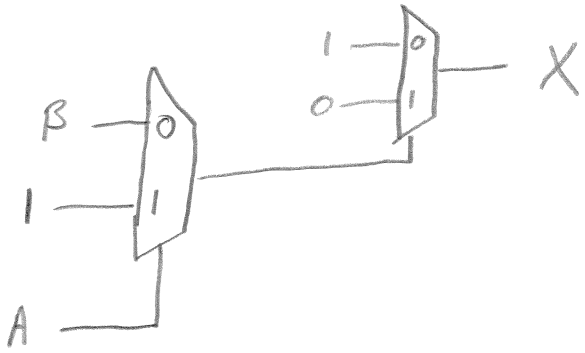


   b. Design a 1 to 2 demux using only AND, OR and NOT gates. Your answer must have 5 or fewer gates to receive credit. Your inputs and outputs should use the same labeling scheme as found in the figure below.





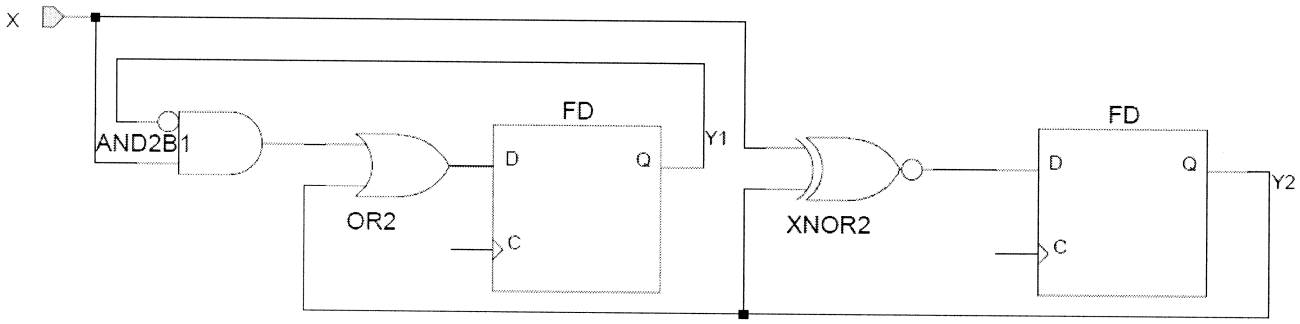
c. Say you are working in a system where the only component you have available is a 2-to-1 MUX. Show how to build a 2-input NOR gate from those MUXes. You can freely connect inputs to "1" and "0" as needed. Label the inputs as "A" and "B". Label the output as "X". *Your solution must use 3 or fewer of the MUXes to receive credit.*



5. **State Machine Analysis**. Write the state table for the synchronous sequential circuit shown here. **[8]**

State Assignment

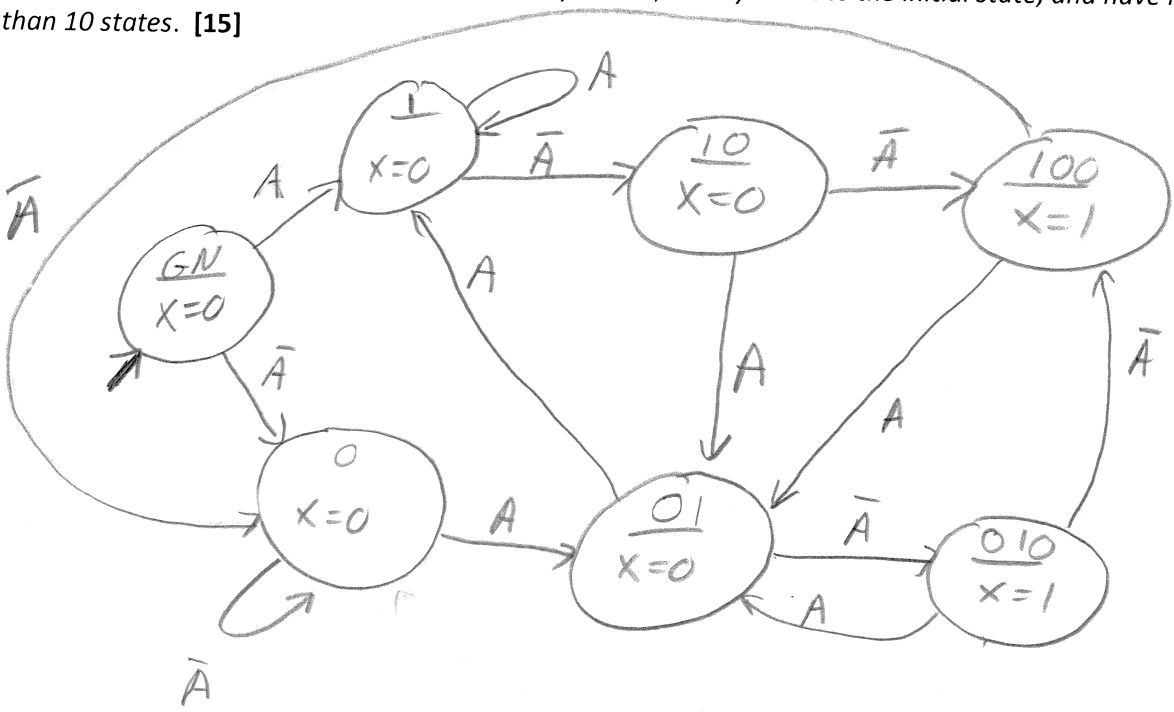
Y1	Y2	State Label
0	0	A
0	1	B
1	0	C
1	1	D



Present State	X=0	X=1
A	B	C
B	C	D
C	B	A
D	C	D

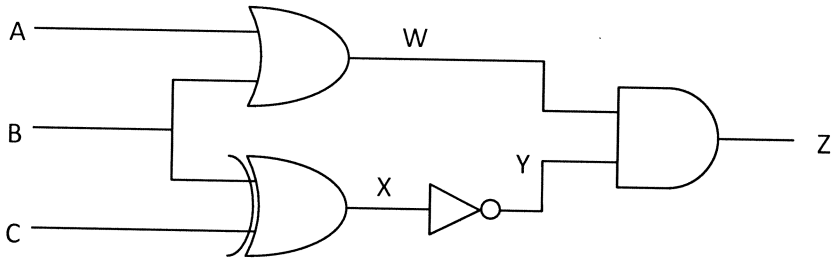
Next State

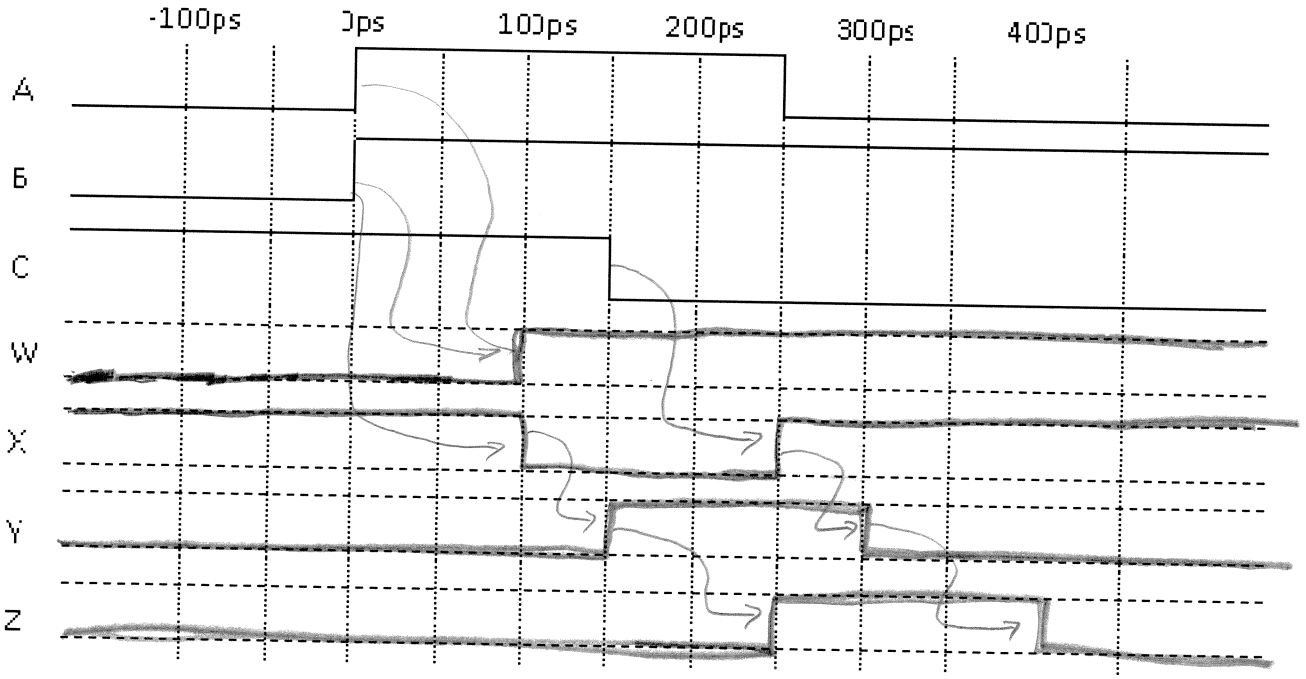
6. **State Machine Design**. Draw a Moore-type state transition diagram (not the circuit) for the following problem. Say we have one input, A, and one output, X. X should be high iff the last three values of "A" were "100" or "010". *Your answer needs to be neatly drawn, clearly indicate the initial state, and have no more than 10 states*. **[15]**



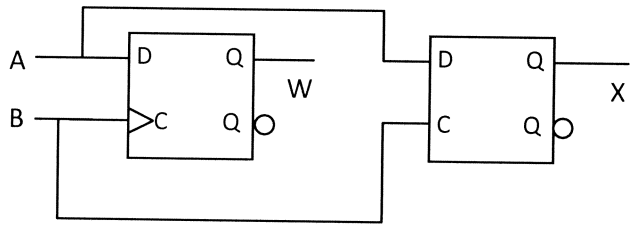
GN is initial state

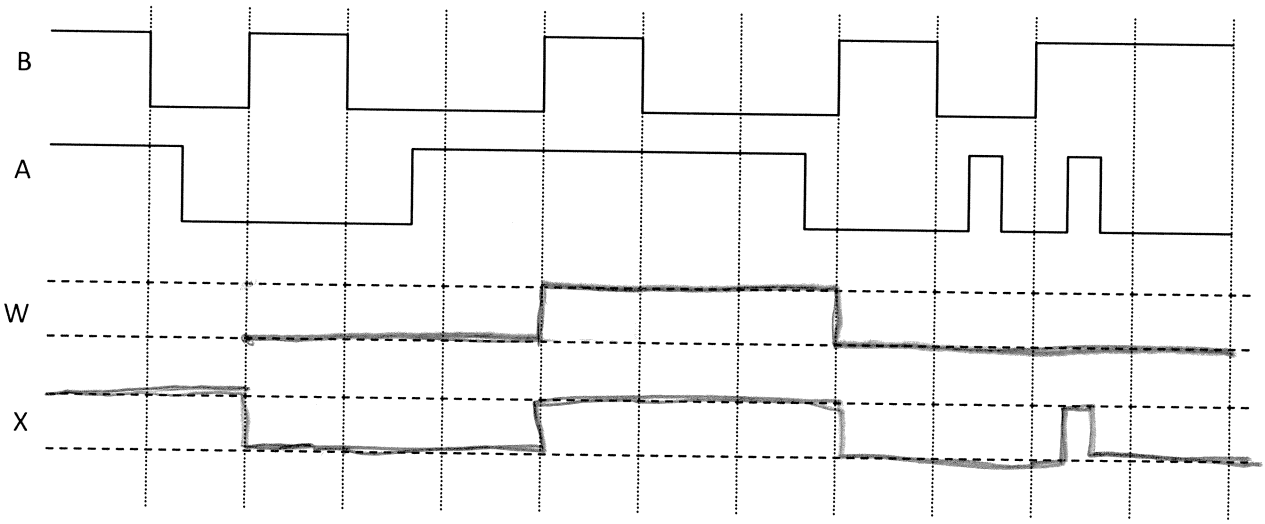
7. **Combinational Timing**. In this circuit NOT gates have a delay of 50ps, while all other gates have a delay of 100ps. Starting at time 0, show how the output and intermediate signals react; include causality arrows. *You need only fill in those times between 0ps and 450ps*. You are to assume the input values have been held constant before time 0 for a long time. **[8]**

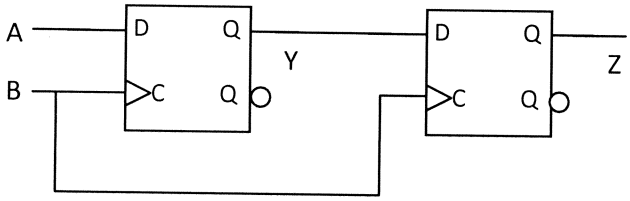


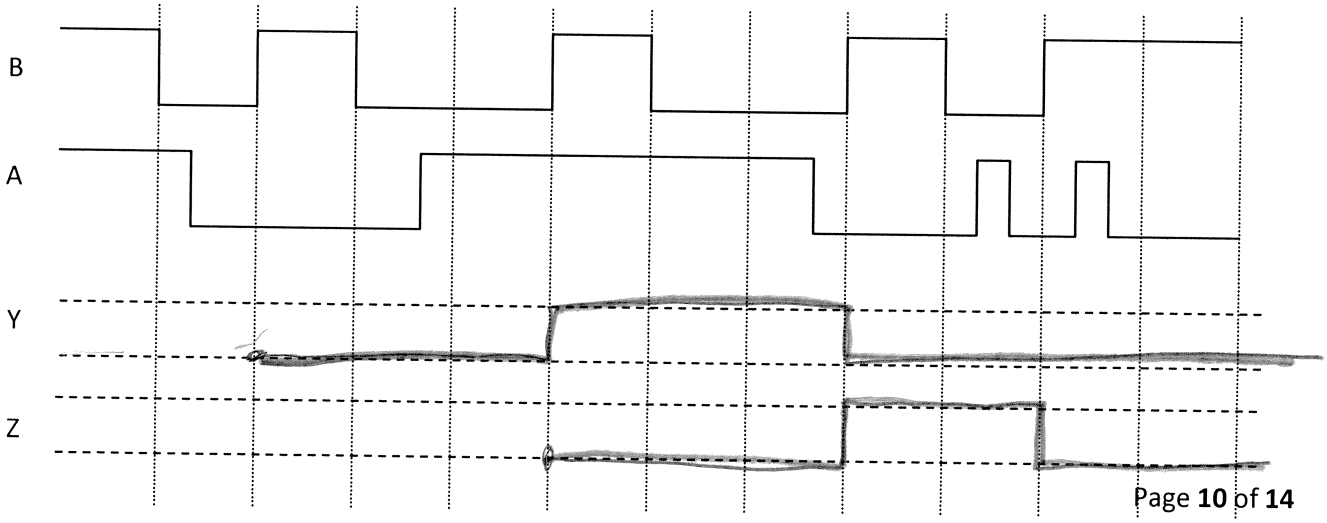


8. **Sequential Circuit Timing**. Fill in the timing diagrams for each of the following circuits. Assume that the circuit delays are small relative to the timescale given and that all setup and hold times are met. *If you can't determine a given value at a given time leave that part blank.* **[8]**



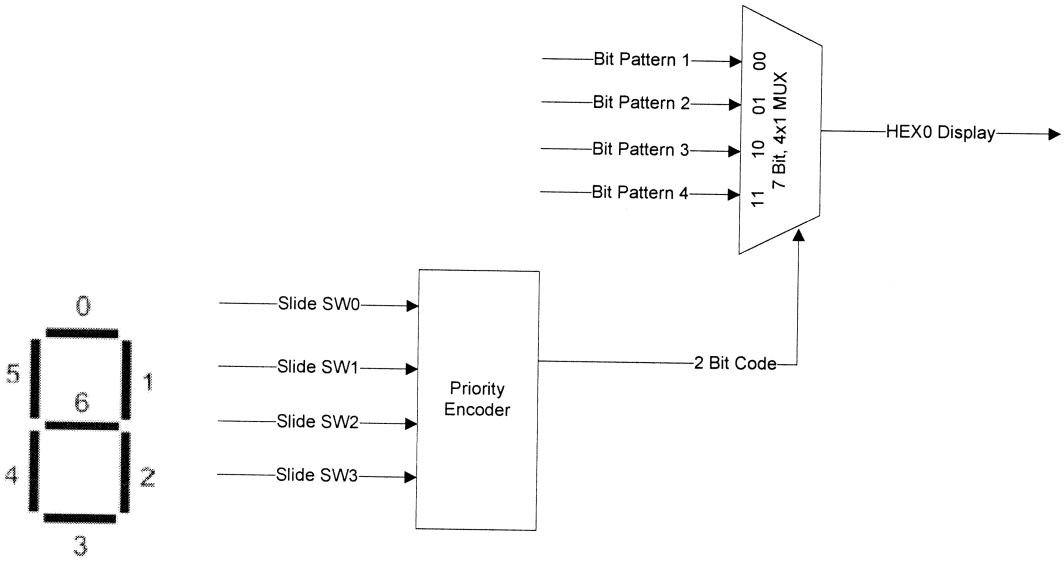






9. **Verilog**. This problem spans 4 pages (including this one).

You've been asked to write a bit of Verilog code for use in the 270 lab. Your code is to read the slide switches 0-3 and output a number 0, 1, 2, or 3 on the HEX0 seven-segment display. The output should be of the highest number switch that is in the "on" position. For example, if switch 3 is on, the number 3 should be displayed on the HEX display regardless of the switch 2-0 settings. A high level functional diagram follows, as does a figure showing how the HEX0 segments are assigned.



For you reference, the priority encoder truth table follows:

I3	I2	I1	I0	Y1	Y0
1	x	x	x	1	1
0	1	x	x	1	0
0	0	1	x	0	1
0	0	0	1	0	0

Note: x means that the value can be either 0 or 1. I3 is the highest priority and I0 is the lowest.

The following the Verilog code implements the switch priority encoding and display multiplexing. Complete the code by choosing the best answer. **[20 points, -3 per wrong or blank answer, minimum 0]**

Notes:

- Recall that a complex multiplexer can be implemented as a combination of simpler multiplexers.
- For your convenience, the high level diagram is provided on the following pages.

```
module priority_encoder (___1___ , Display);
input [3:0]A;  //connect to slide switches 3-0 respectively
output [6:0] Display;  //connect to hex display H0[6:0] respectively
wire [6:0] Display;
wire [1:0] EP;

//hex display bit patterns
`define N1 'b1111001 //1
`define N2 'b0100100 //2
`define N3 'b___2___ //3
`define N4 'b0011001 //4

//priority encode switch inputs: SW3 highest, SW0 lowest
assign EP[0]= ~A[3] & ~A[2] & A[1] | A[3];
assign EP[1]= ____3____;

//select HEX bit patterns according to priority
mux7bit_4x1  inst1(_____4_____ );

endmodule
```

Blank 1

a) SW[3:0]
b) A
c) I[3:0]
d) All of the above would work

Blank 2

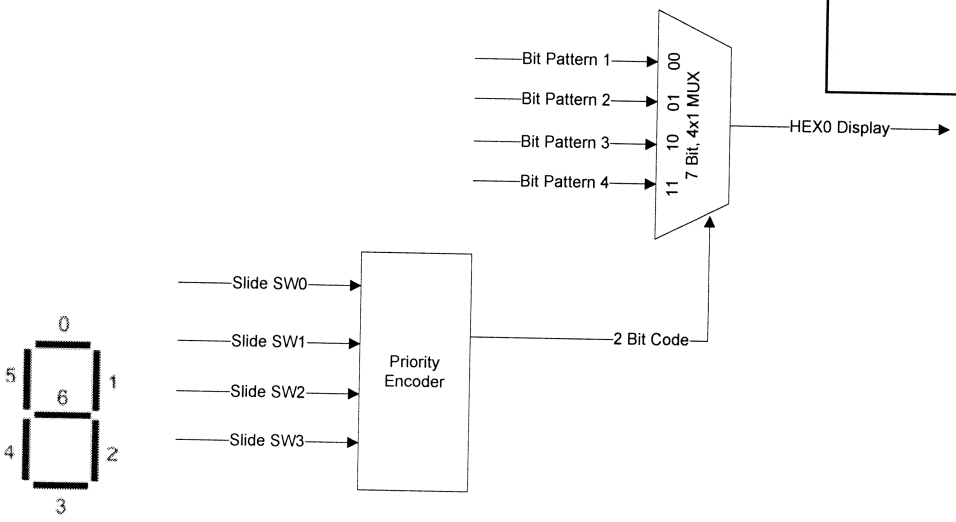
a) 0000110
b) 1001111
c) 0110000
d) 0011000

Blank 3

a) ~A[3] & A[2] | A[3]
b) ~A[3] & A[2] &~A[1]&~A0[1] | A[3] & ~A[2] &~A[1]&~A0[1]
c) A[3] & ~A[2] | A[2] | ~A[1]
d) None of the above

Blank 4

a) EP, `N1, `N2, `N3, `N4, Display
b) select, A, B, C, D, Y
c) A, SW0, SW1, SW2, SW3, HEX0)
d) None of the above



```
//7 bit 4 to 1 mux
module ____5____ (select, A, B, C, D, Y);
input [1:0] select;
input [6:0] A, B, C, D;
output [6:0]Y;
wire [6:0] Y, Y0, Y1;
wire sel1, sel2, sel3;

 //decode mux select lines
assign sel1 = _________6_________;
assign sel2 = ~select[0] & select[1];
assign sel3 = select[0] & select[1];

//make 4 to 1 mux with 3, 2 to 1 muxs
mux7bit_2x1  inst1 (____7____);
mux7bit_2x1  inst2  (sel2, Y0, C, Y1);
mux7bit_2x1  inst3  (____8____);

endmodule
```

Blank 5

a) mux7bit_4x1
b) mux7bit_4x1 inst1
c) inst1
d) none of the above

Blank 6

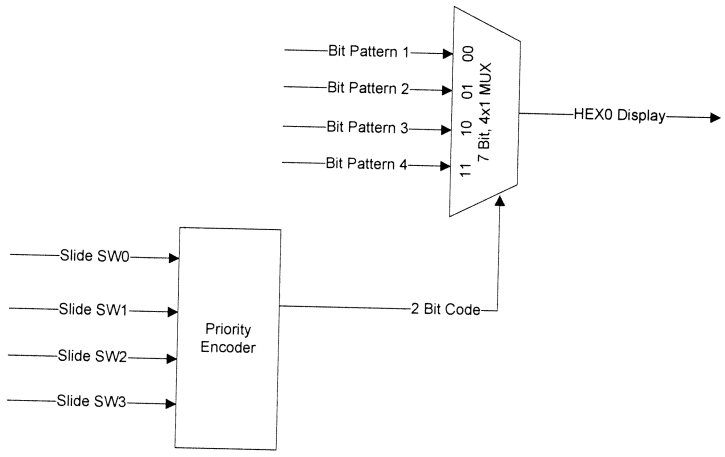
a) select[1]
b) ~select[0] & ~select[1]
c) select[0] & ~select[1]
d) none of the above

Blank 7

a) sel1, A, B, Y1
b) sel1, A, B, Y0
c) sel1, B, A, Y0
d) none of the above

Blank 8

a) sel3, Y1, B, Y
b) sel3, Y1, C, Y
c) sel3, Y1, D, Y
d) none of the above



```
//7 bit 2 to 1 mux
module mux7bit_2x1(select, A, B, Y);
input select;
input [6:0] A, B;
output [6:0]Y;
wire [6:0]Y;

_____9_____

endmodule
```

Blank 9

a) assign Y[0] = select & A[0] | B[0];
assign Y[1] = select & A[1] | B[1];
assign Y[2] = select & A[2] | B[2];
assign Y[3] = select & A[3] | B[3];
assign Y[4] = select & A[4] | B[4];
assign Y[5] = select & A[5] | B[5];
assign Y[6] = select & A[6] | B[6];

b) assign Y[0] = ~select & A[0] | select & B[0];
assign Y[1] = ~select & A[1] | select & B[1];
assign Y[2] = ~select & A[2] | select & B[2];
assign Y[3] = ~select & A[3] | select & B[3];
assign Y[4] = ~select & A[4] | select & B[4];
assign Y[5] = ~select & A[5] | select & B[5];
assign Y[6] = ~select & A[6] | select & B[6];

c) if (select) then assign Y = A else assign Y = B

d) All of the above would work.