EE109: Introduction to Embedded Systems
Spring 2023 - Midterm Exam
3/7/23, 7PM – 8:40PM

[Complete all the information in the box below.]

Name:________________________________

Student ID: ____

Email: ____________________@usc.edu

Lecture section (Circle One):

Redekopp	Redekopp	Weber	Puvvada
9:30 a.m.	11 a.m.	12:30 p.m.	2 p.m.

Ques.	Your score	Max score	Recommended Time
-		-	0 min.
1		6.5	6 min.
2		2	2 min.
3		4.5	5 min.
4		7	10 min.
5		8	10 min.
6		12	15 min.
7		10	15 min.
8		8	12 min.
9		17	25 min.
Total		75	

All work MUST be on these EXAM PAGES. No Scratch work will be graded or viewed.

1. **(6.5 pts) Number Systems**

   1.1. Convert **153** decimal to an 8-bit unsigned binary number: ______________________

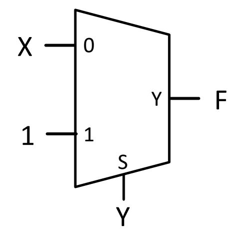
   1.2. Convert **1000111011101.110** binary to hexadecimal: ______________________

   1.3. Using a **6-bit code** to represent colors would allow for how many unique colors to be represented: ________

2. **(2 pts.) Mux Behavior**

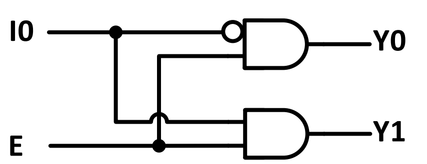
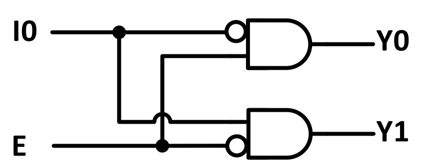
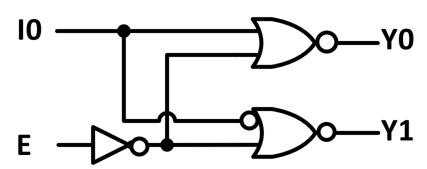
   The circuit to the right can be equivalently replaced with a single 2-input logic gate (e.g. **NAND, OR**, **XOR**…) with inputs: {X,Y} and output F. Identify that gate:

   Equivalent gate type: ______



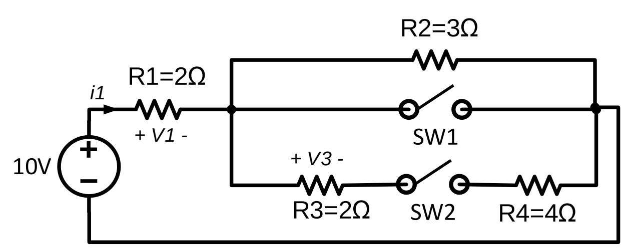
3. **(4.5 pts.) Decoder Behavior**

   Consider the three circuits below. Indicate which ones form a valid 1-to-2 decoder **<u>with</u>** enable. Hint: Use your knowledge of a decoder but apply DeMorgan's theorem to manipulate the circuits and verify if they do indeed implement a **valid** 1-to-2 decoder. (Circle the correct option for each circuit).

		
Valid / Invalid	**Valid / Invalid**	**Valid / Invalid**

4. **(7 pts.) Analog Circuits**

   Consider the resistive circuit to the right and answer the following questions.



   4.1. When SW1 and SW2 **open**, what is **V1**? ____ **V**

   4.2. When SW1 is **open** and SW2 **closed**, what is the **equivalent resistance** of all the resistors? ____ **ohms**

   4.3. When SW1 AND SW2 are **closed**, what is the **equivalent resistance** of ALL the resistors and what is **the current, i1,**?

   ____ **ohms** ***i1* =** _____ **A**

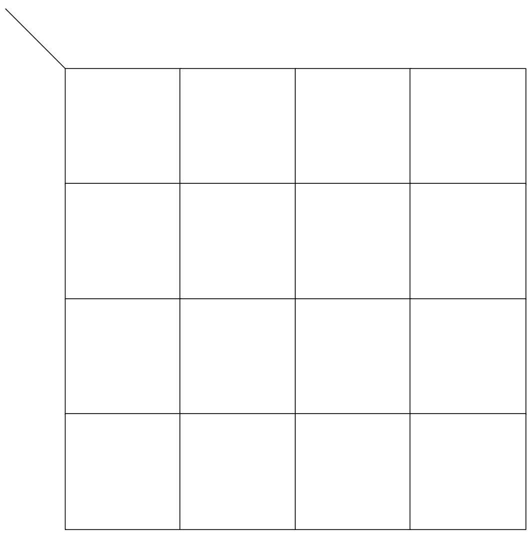
5. (8 pts.) Logic Simplification

Billy Bruin arrived at what he believes is a **minimal, POS equation** for a function, F, that he desired to implement. The equation he found was:

$$F(w, x, y, z) = (w + x + \overline{y} + z)(\overline{x} + z)(\overline{w} + z)$$

To prove or disprove that Billy Bruin's equation truly is a minimal, POS implementation, **construct a Karnaugh map in the area below** using the equation above to fill in the values. Then **group** and **translate** to show the minimal, **POS** equation yielded by your Karnaugh Map and show your answer in the blank below to see if it agrees with the equation Billy found.

5.1. **Construct, group and translate a Karnaugh map for the given equation in the space below.**



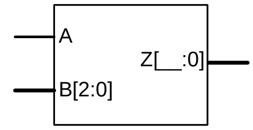
5.2. **Minimal POS equation for F that you found:**

F = ____

6. (12 pts.) Logic Function Design

Consider a circuit which takes as input a **__1-bit__** value **A,** and a **__3-bit__** __unsigned__ input **B[2:0]** (i.e. B2,B1,B0). The output of the circuit should be an __unsigned__ value, **Z** given by the description below. Then use the K-maps to find the specified minimal expressions.

```
if      (A is 0)                     then  Z =   B[2:0]
else if (A is 1 and B[2:0] < 3)      then  Z = 2*B[2:0] + 1
else                                       Z =   B[2:0] - 3
```



Note: If you consider the above description, a negative result is NOT possible.

Ex. 1: if A = 1 and B = 010 (2 decimal) then because A is 1 and $B < 3$, Z = 2*2 + 1 = 5 dec.
Ex. 2: if A = 1 and B = 110 (6 decimal) then because A is 1 and $B \nless 3$, Z = 6 – 3 = 3 dec.

6.1. What is the minimum number of output bits needed for Z? _______________

6.2. Complete the truth table for this circuit showing the Z output bits .

6.3. Use the Karnaugh maps to find the minimal **SOP expression for Z0** and minimal **POS expression for Z1**. You do not have to implement any other bits of Z.

A	B2	B1	B0	Z1	Z0
0	0	0	0		0
0	0	0	1		1
0	0	1	0		0
0	0	1	1		1
0	1	0	0		0
0	1	0	1		1
0	1	1	0		0
0	1	1	1		1
1	0	0	0		1
1	0	0	1		1
1	0	1	0		1
1	0	1	1		0
1	1	0	0		1
1	1	0	1		0
1	1	1	0		1
1	1	1	1		0

Kmap for Z0,SOP

B1B0 \ A B2	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

Kmap for Z1,POS

B1B0 \ A B2	00	01	11	10
00	0	4	12	8
01	1	5	13	9
11	3	7	15	11
10	2	6	14	10

*Fill in the minimal **SOP** expression for Z0.*

Z0 ***minimal,SOP*** = ____

*Fill in the minimal **POS** expression for Z1.*

Z1 ***minimal,POS*** = ____

7. **(10 pts.) Boolean Algebra:** Use theorems to simplify the given equation for G to its minimal **SOP** representation. You need to use DeMorgan's theorem in the first step. And at some point you MUST use T8' (whenever it seems helpful). You must show what theorems you are applying at each step (though you can apply 2 or 3 theorems per step). Write neatly and circle your final answer. We **strongly** recommend you (PLEASE!!) plan your work on the scratch paper first to determine your approach but your **final solution must be** on this page. S*inge/multi variable and DeMorgan's theorem are listed below*.

Step	**Theorem(s) or Manipulation(s) Used**
$G = \overline{A} \cdot \overline{D} + \left[\overline{\left(A + \overline{D}\right) \cdot \left(A + \left(B \cdot \overline{C}\right)\right)}\right] + A \cdot \overline{C} \cdot \left[B \cdot D + \overline{\left(\overline{B} + D\right)}\right]$	
$G = \overline{A} \cdot \overline{D} +$ ____________________ $+ A \cdot \overline{C} \cdot [$_______________$]$	**DeMorgan's Theorem**

Use only the rows needed

Single-Variable Theorems

(T1)	X + 0 = X	(T1')	X • 1 = X	(Identities)
(T2)	X + 1 = 1	(T2')	X • 0 = 0	(Null elements)
(T3)	X + X = X	(T3')	X • X = X	(Idempotency)
(T4)	(X')' = X			(Involution)
(T5)	X + X' = 1	(T5')	X • X' = 0	(Complement)

Two- and Three-Variable Theorems

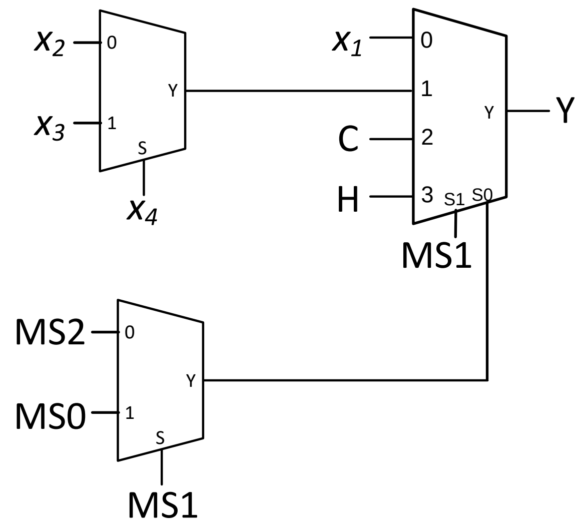
(T6)	X +Y = Y + X	(T6')	X • Y = Y • X	(Commutativity)
(T7)	(X+Y)+Z = X+(Y+Z)	(T7')	(X•Y) •Z = X• (Y•Z)	(Associativity)
(T8)	X•(Y+Z) = X•Y + X•Z	(T8')	X+(Y•Z) = (X+Y) • (X+Z)	(Distributivity)
(T9)	X + X•Y = X	(T9')	X • (X + Y) = X	(Covering)
(T10)	X•Y + X•Y' = X	(T10')	(X+Y) • (X+Y') = X	(Combining)
(T11)	X•Y+X'•Z+Y•Z = X•Y+X'Z	(T11')	(X+Y)•(X'+Z)•(Y+Z) = (X+Y)•(X'+Z)	(Consensus)

DeMorgan's Theorem

(X • Y)' = X' + Y'	(X +Y)' = X' • Y'	(DeMorgan's)

8. **(8 pts.) Tree Muxes.** – Suppose you are given **8 data inputs: A-H** that correspond to the select combinations shown in the table below. Now suppose we ONLY want to design a **5-to-1** mux to select and pass a **subset of 5 of the 8 input using the select bits MS2, MS1, MS0.** Billy Bruin's initial attempt to design the 5-to-1 mux is shown. However, Billy was unsure what to connect to some of the inputs and, instead, used placeholder variables: **x1-x4**. Assuming Billy's design follows the specification of input / select combinations in the table below and **can output 5 UNIQUE** inputs from the set A-H, help Billy by answering the following questions.

MS2	MS1	MS0	Y
0	0	0	A
0	0	1	B
0	1	0	C
0	1	1	D
1	0	0	E
1	0	1	F
1	1	0	G
1	1	1	H



8.1. Whenever **MS1** is **0**, the select bit that will pass/connect to S0 input of the 4-to-1 mux is (circle the correct answer): **MS2** / **MS0**

8.2. What input(s) (or subset of inputs if many are possible), **A-H**, could correctly be connected to the input labelled **x1** (circle all that are possible):

x1: **A B C D E F G H**

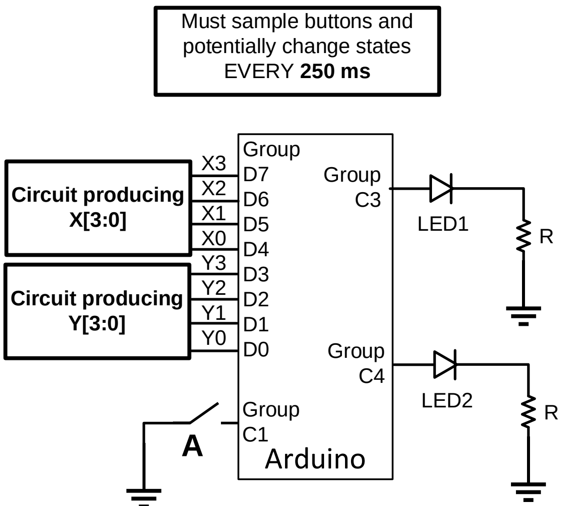
8.3. What inputs (from the options **A-H**, **MS2, MS1,** and **MS0**) MUST be connected to **x2, x3**, and **x4**.

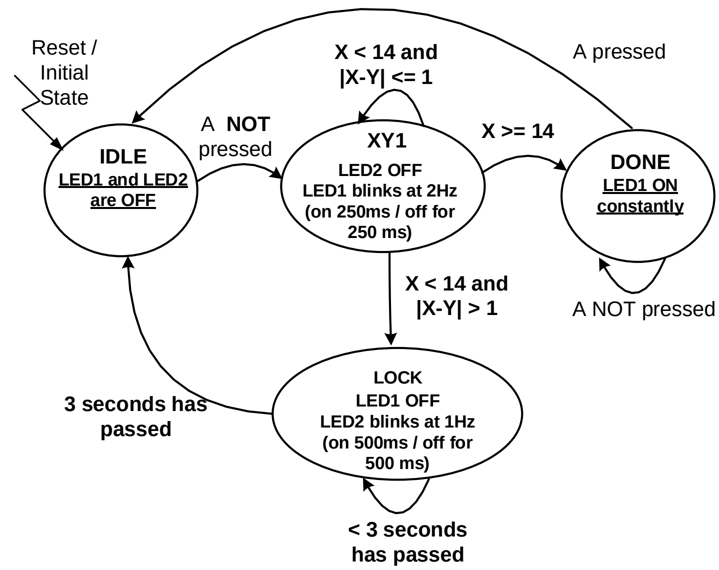
x2=____ **x3**=____ **x4**=____

8.4. Because of where Billy has connected **C** and **H** to inputs 2 and 3 of the 4-to-1 mux, which **two** inputs listed below cannot be connected anywhere. Or said differently, which **two** inputs could never be selected and passed to the output of the mux.

(Circle two): A B D E F G

9. **(17 pts) State Machines -** Complete the implementation of Arduino code (on this and the following page) to implement the following behavior which will require the use of state. Two circuits produce 4-bit unsigned numbers: **X[3:0]** and **Y[3:0]** and are connected to group **D** as shown below. A button: **A** (on group **C**, bit **1**) is connected as shown below. Two LEDs are connected: **LED1** is connected to group **C**, bit **3** and **LED2** to group **C**, bit **4**. The state should transition based on the values of the 4-bit numbers X and Y as well as the button A as shown in the state diagram below. The buttons MUST be sampled (and state updated) **every 250 ms**. The LEDs should be off, blink, or be on as described in the state diagram.





- **<u>You should NOT add any other delay statements (_delay_ms()) to the code.</u>**
- **You may not change the structure or values of the code provided in the skeleton.**
- **You need not worry about debouncing.**

Assume the following #includes and declarations should you want to use them.

```
#include <avr/io.h>
#include <util/delay.h>       // allows for _delay_ms() function
#include <stdlib.h>           // allows abs() - absolute value

const int A = 1; LED1 = 3, LED2 = 4;
enum {IDLE, XY1, DONE, LOCK};

```

```
int main() {
    char state = IDLE, cnt; // state variable and 3 sec. count
    /* Other necessary intiailization code */





    while(1) {  // this is the only loop allowed
        _delay_ms(250);  /* this is the ONLY delay statement allowed */
        char a = (___________ & __________________); // sample the A button input

        // combined next state and output logic
        if( state == IDLE ) {
            _____________________________________________; // appropriate output action
            if( __________________________________ ) { state = XY1; }
        }
        else if (state == XY1) {
            unsigned char inp = PIND;
            // extract 4-bits of x and y as separate numbers that can be compared
            char y = (inp & 0x______________);
            char x = ______________________________________________________________;
            if(x >= 14){
                state = DONE;
            }
            else if(abs(x-y) > 1){
                state = LOCK;
                cnt = ____;
            else {
                PORTC ___= (1 << LED1);   // Enter operator to flip/toggle LED1
            }
        }
        else if(state == DONE) {
            ____________________________;
            if(_____________________________ ) { state = IDLE; }
        }
        else {
            PORTC ________________________________;      // Clear the appropriate LED
            cnt++;
            if( (cnt % ______) == ______   )
              { PORTC ___= (1 << LED2); }   // Enter operator to flip/toggle LED2
            if(cnt == _____){
                state = IDLE;
        }   }
    } /* end while */
    return 0;
} /* end main */
```