CSE320 Exam 1

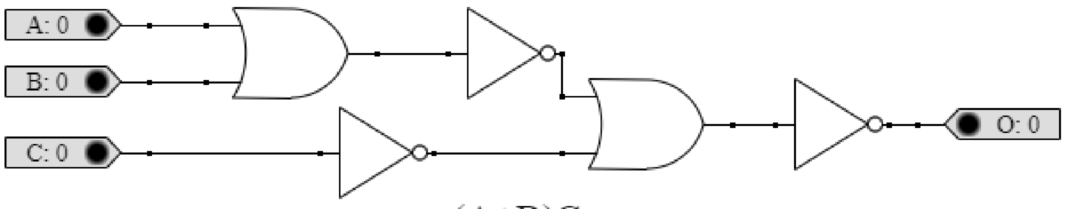
Spring, 2020 exam 153226

1.(15 pts)

a) A 3-bit binary function has how many minterms? ___8___.

1 point for this part.

b) Using only OR and NOT gates, draw a circuit for F(A,B,C)=(A+B)C



(A+B)C

Up to 5 points. If they are off by one edge somewhere, maybe -1

c) A 64-bit ARM processor executes 32-bit instructions by translating them on-the-fly to the equivalent 64-bit instructions. The fact that this mechanism can execute 32-bit instructions in a completely different way, yet have them work identically is due to ___abstraction___.

1 point for this part.

d) Given the following function in C, what is the output of each line?

```
void func_p()
{
    int nw = 123;
    int e = 0xFFFFFF16;
    int o = 0x000000A5;
    printf( "%08X\n", e | o );      _____________FFFFFFB7_____________
    printf( "%08X\n", -nw );        _____________FFFFFF85_____________
    printf( "%08X\n", e & o );      _____________00000004_____________
    printf( "%08X\n", nw );         _____________0000007B_____________
    printf( "%d\n", o );            _____________165______________
}
```

1 point per item.

e) Assuming a function in the three variables L, M, and N, express (NM'+NL'M')M' in canonical sum of products form:

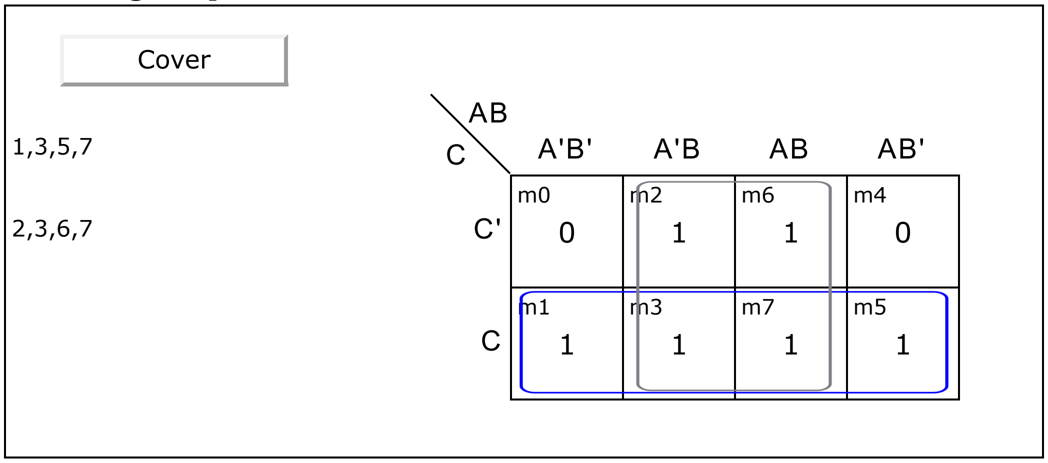
L'M'N+LM'N

3 points for this part of the question. -1 if only off by one term.

2.(20 pts) For each of the following problems, provide a minimized expression. Show all your work.

a) F(A,B,C)=m1+m2+m3+m5+m6+m7

✓ **Karnaugh Map**



Cover

1,3,5,7

2,3,6,7

C \ AB	A'B'	A'B	AB	AB'
C'	m0 0	m2 1	m6 1	m4 0
C	m1 1	m3 1	m7 1	m5 1

C+B

5 points

b) F(A,B,C) as specified by this truth table:

A	B	C	F(A,B,C)
0	0	0	1
0	0	1	X
0	1	0	1
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

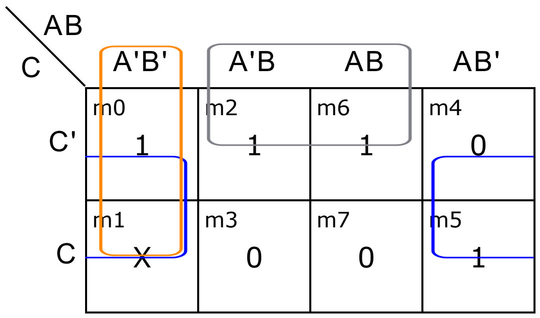
✓ **Karnaugh Map**

Cover

1,5

2,6

0,1

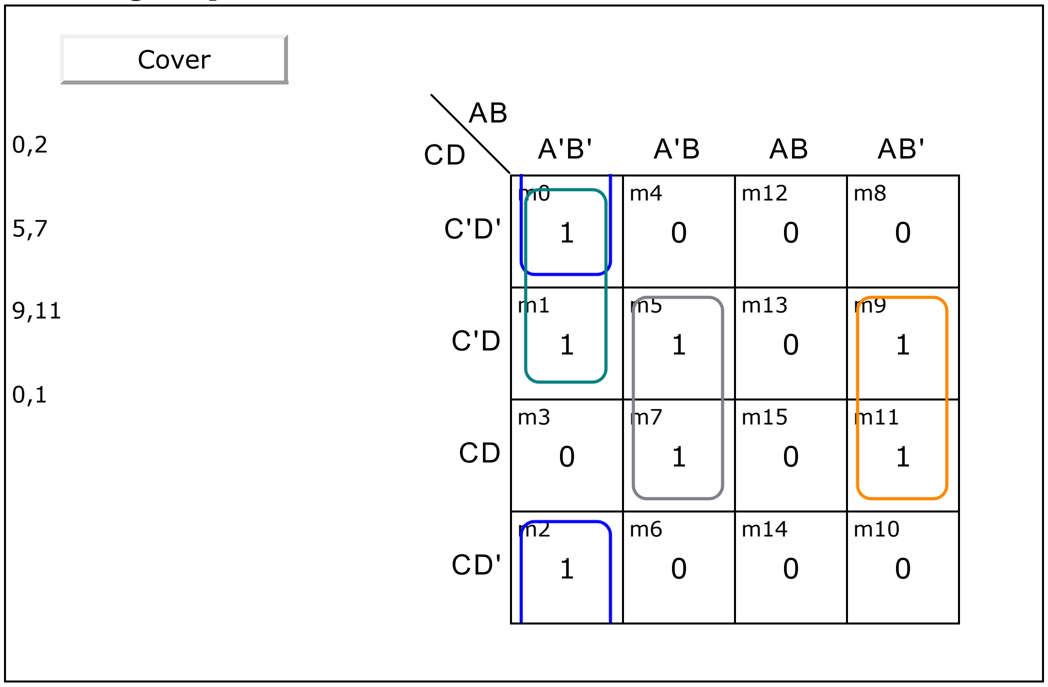


B'C+BC'+A'B'

5 points

c) F(A,B,C,D)=m0+m1+m2+m5+m7+m9+m11

☑ **Karnaugh Map**



Cover

0,2

5,7

9,11

0,1

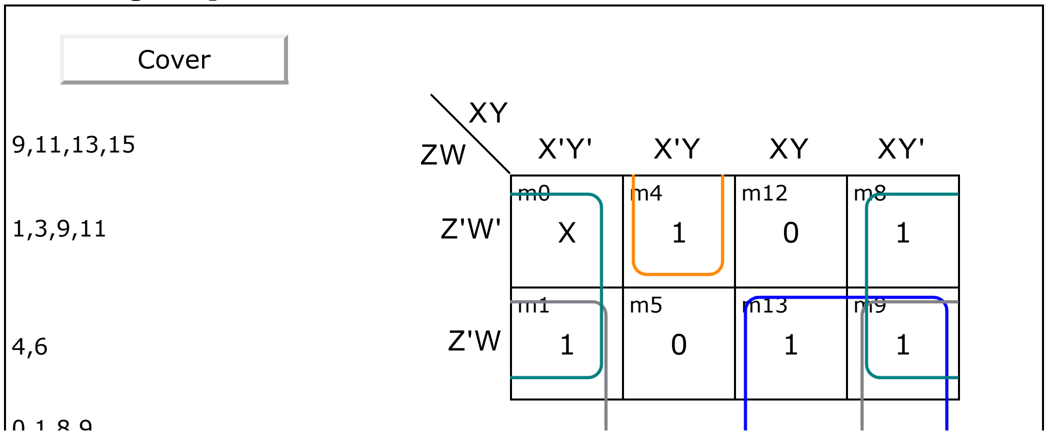
CD \ AB	A'B'	A'B	AB	AB'
C'D'	m0 1	m4 0	m12 0	m8 0
C'D	m1 1	m5 1	m13 0	m9 1
CD	m3 0	m7 1	m15 0	m11 1
CD'	m2 1	m6 0	m14 0	m10 0

A'B'D'+A'BD+AB'D+A'B'C'

5 points

d) F(X,Y,Z,W)=m1+m3+m4+m6+m8+m9+m13+m15 and don't cares are m0,m7,m10,m11,m14

☑ **Karnaugh Map**



Cover

9,11,13,15

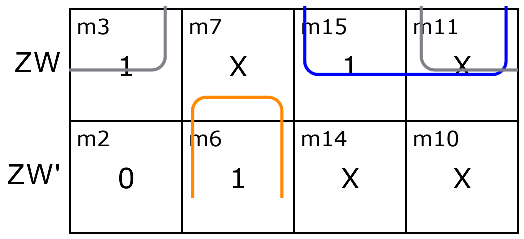
1,3,9,11

4,6

0,1,8,9

ZW \ XY	X'Y'	X'Y	XY	XY'
Z'W'	m0 X	m4 1	m12 0	m8 1
Z'W	m1 1	m5 0	m13 1	m9 1

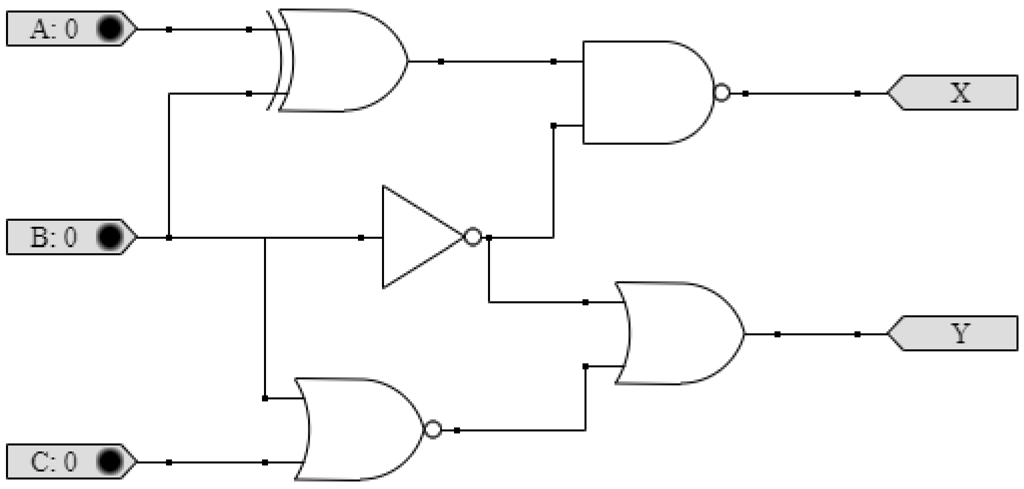
0,1,8,9



XW+Y'W+X'YW'+Y'Z'

5 points

3.(10 pts)

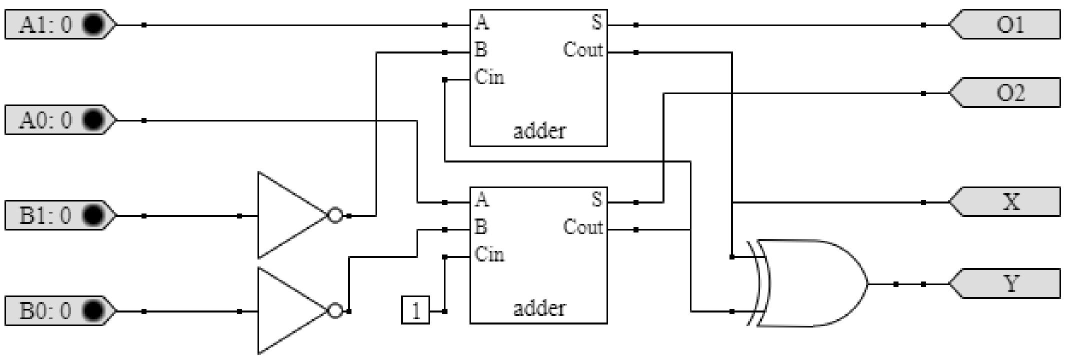


Complete the truth table for the circuit above, assuming the outputs are X and Y.

A	B	C	X	Y
0	0	0	1	1
0	0	1	1	1
0	1	0	1	0
0	1	1	1	0
1	0	0	0	1
1	0	1	0	1
1	1	0	1	0
1	1	1	1	0

-1 for each for up to 5 missed, then -1 for each two missed up to -10 points.

4.(10 pts)

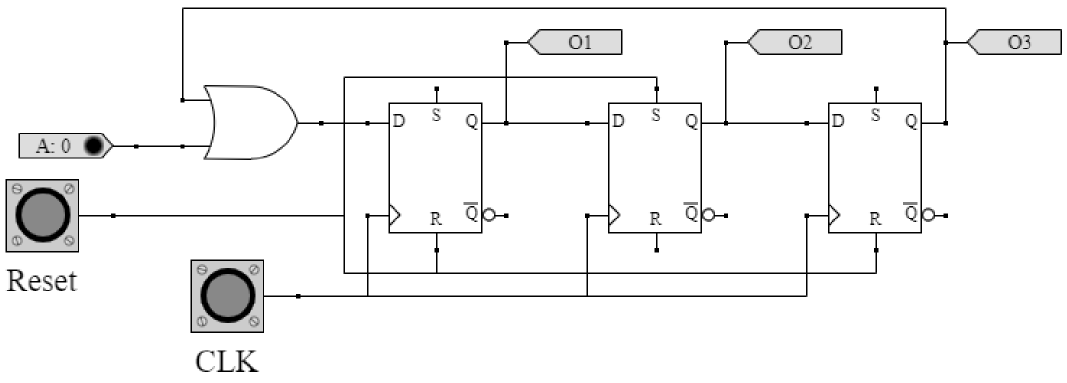


Fill in the blanks in truth table rows for the above circuit, where the component named *adder* is a full adder. You are only required to complete the rows indicated.

A1	A0	B1	B0	O1	O2	X	Y
0	0	0	1	1	1	0	0
0	1	0	1	0	0	1	0
0	1	1	0	1	1	0	1
1	1	0	1	1	0	1	0

-1 for each for up to 5 missed, then -1 for each two missed up to -10 points.

5.(15 pts) You are given the following circuit:



a) Suppose the input A is set to 0. What will be the output after each button press, assuming the button presses happen in this order:

Button Press	O1	O2	O3
RESET	0	1	0
CLK	0	0	1
CLK	1	0	0
CLK	0	1	0
CLK	0	0	1

-1 for each for up to 3 missed, then -1 for each two. Maximum -8 on (a).

b) Suppose the input A is set to 1. What will be the output after each button press, assuming the button presses happen in this order:

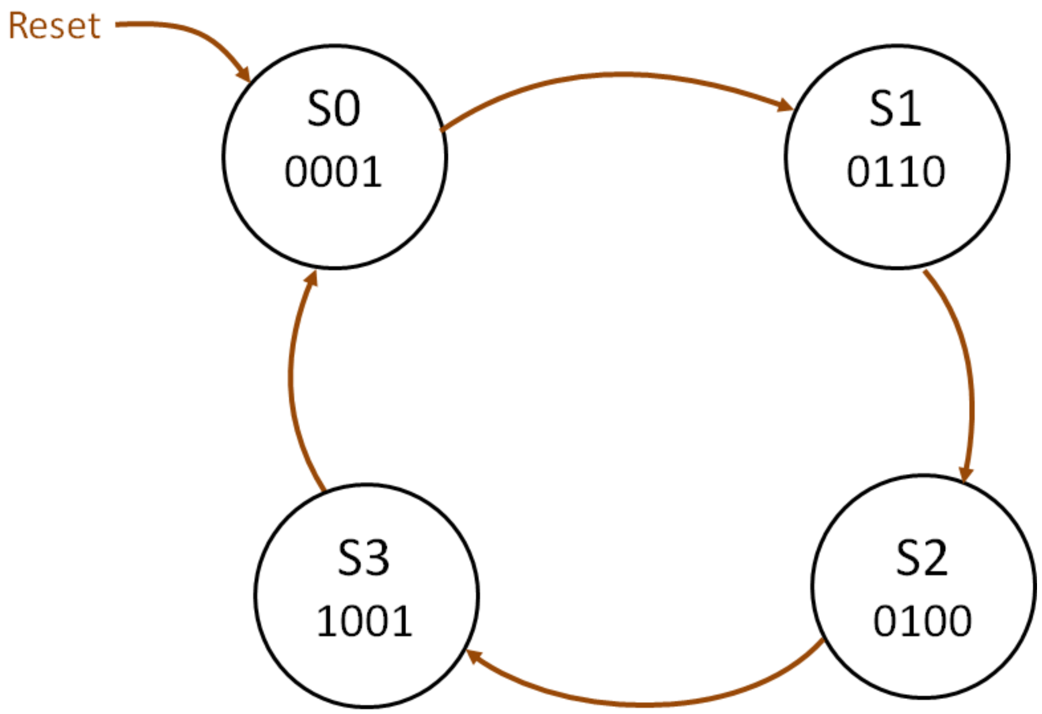
Button Press	O1	O2	O3
RESET	0	1	0
CLK	1	0	1
CLK	1	1	0
CLK	1	1	1

CLK	1	1	1

-1 for each for up to 3 missed, then -1 for each two. Maximum -7 on (b).

6.(20 pts) Assume you must create a finite state machine that outputs the sequence 0001, 0110, 0100, 1001 and then repeats. Assume the first output upon a reset is 0001. If the clock is applied six times, the output will be 0001, 0110, 0100, 1001, 0001, 0110, 0100, 1001. The value 0001 and values following that repeat!

a) Draw a state diagram for the sequence:



There are two ways to solve this problem. You can use a 4-bit state or you can use 2-bits. Since there are only four actualy states, the 2-bit solution is much simpler. This diagram assumes the states are 2-bits and not the actual output values.

Up to -5 for (a). It requires 4 states. Cannot be done with less.

b) Write the truth table for the next states in the sequence:

There is the solution assuming a 2-bit state:

S_1	S_0	S_1*	S_0*
0	0	0	1

0	1	1	0
1	0	1	1
1	1	0	0

And this is the solution assuming a 4-bit state:

S_3	S_2	S_1	S_0	S_3*	S_2*	S_1*	S_0*
0	0	0	1	0	1	1	0
0	1	1	0	0	1	0	0
0	1	0	0	1	0	0	1
1	0	0	1	0	0	0	1

Up to -5 for (b). -1 for each wrong value. If the truth table matches their state diagram, even if the diagram is incorrect, give them credit for (b).

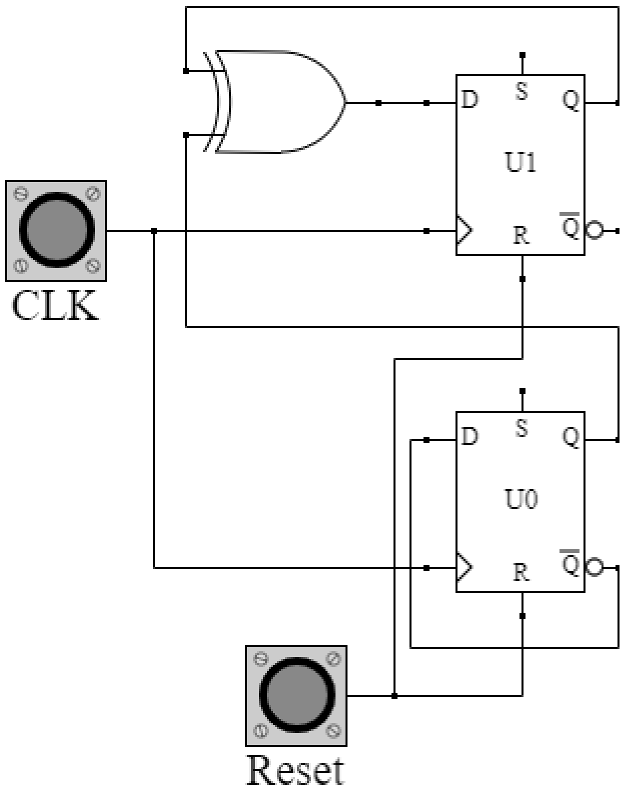
c) Draw a circuit diagram for a finite state machine that generates this sequence. Clearly indicate the circuit outputs:

This writeup assumes a 2-bit state. See below for the 4-bit state solution.

First you needed to determine the expressions for the next state:

$$S_0{}^* = S_0'$$
$$S_1{}^* = S_1 \text{ xor } S_0$$

Then the actual circuit for the state machine is pretty easy:



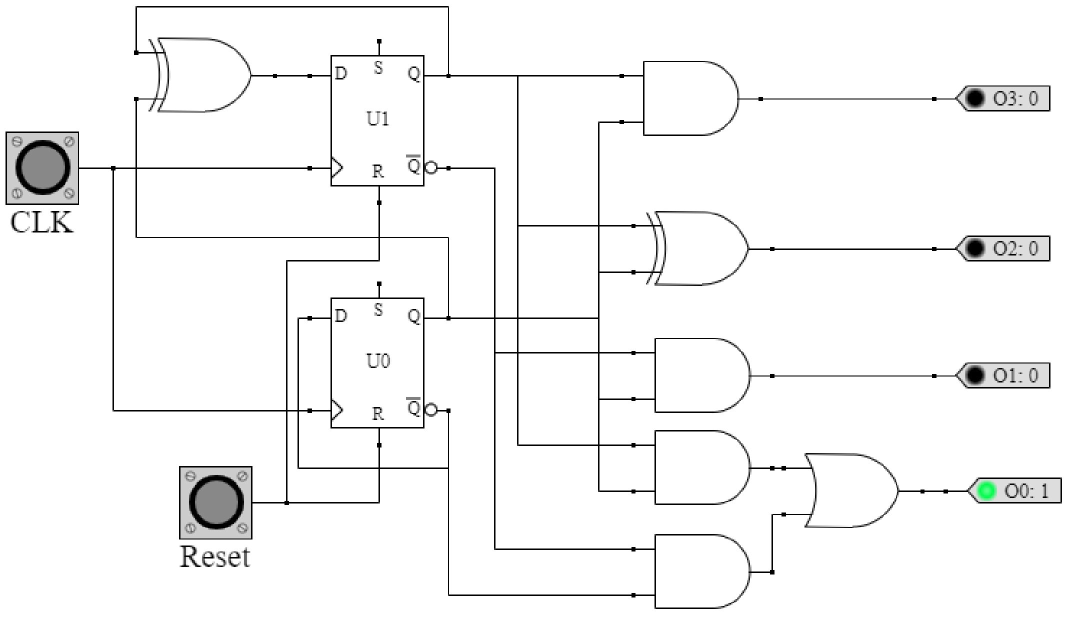
Of course, this does not generate the output we need. We need some logic to convert the states to the correct output values. First, here's a truth table for the output:

S_1	S_0	O_3	O_2	O_1	O_0
0	0	0	0	0	1
0	1	0	1	1	0

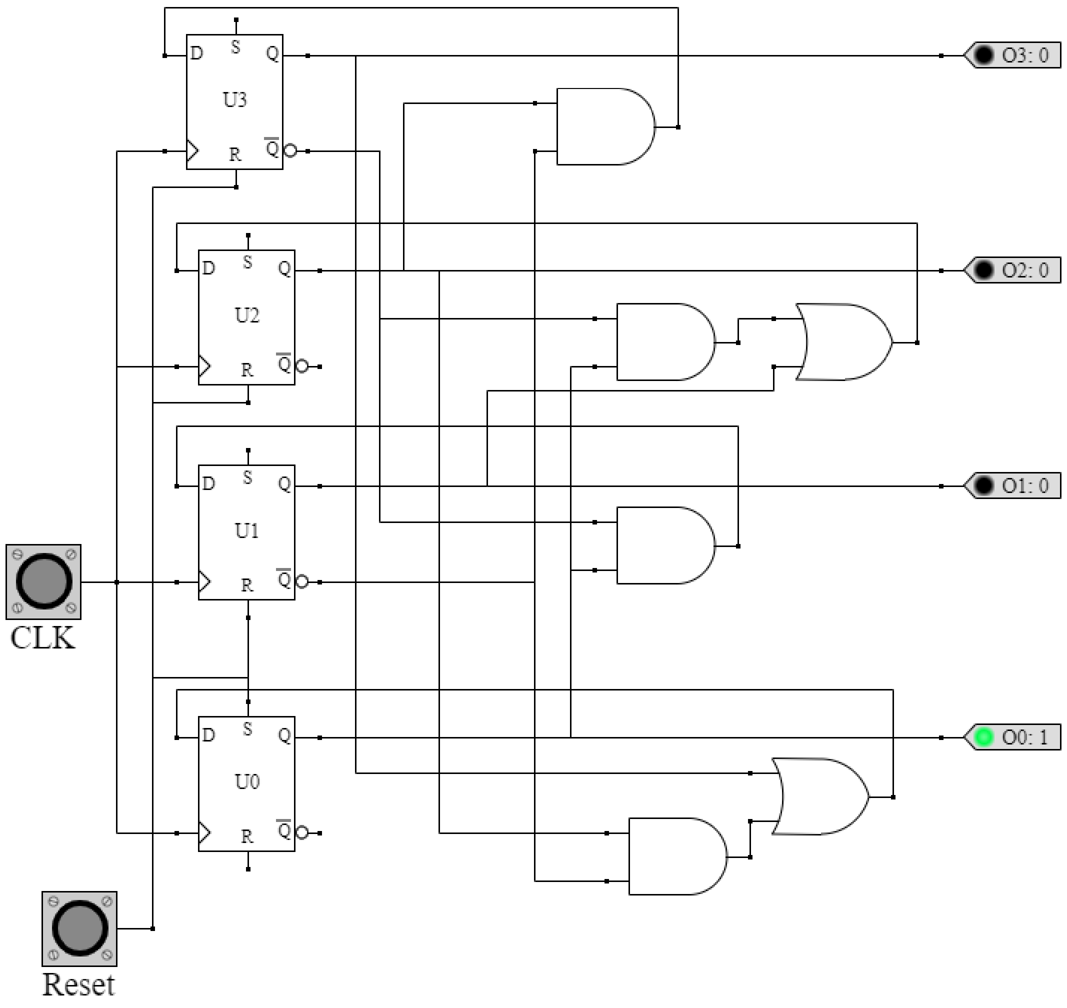
1	0	0	1	0	0
1	1	1	0	0	1

Of course, this is easy to implement.

$$O_3=S_1S_0'$$
$$O_2=S_1 \text{ xor } S_0$$
$$O_1=S_1'S_0$$
$$O_0=S_1S_0+S_1'S_0'$$



If you were really committed to the idea of states as outputs, here's the 4-bit solution:



Up to -10 for (c).

7.(10 pts) This well-commented assembly language function prints the largest number in a passed array using printf. Fill in the missing code in the supplied blanks.

```
.data

largestMsg: .asciz "Largest value is %d\n"

.text
.global largest

@
@ Print largest number in an array
@
@ void largest(int *data, int size)
@
largest:
    stmfd sp!,{r4-r8,lr}

    ldr r4, _[r0]_    @ Use first location in the
                            @ array as the current largest r4

largest_loop:

     movs r1,r1 (or) cmp r1,#0  @ size=0?
    beq largest_result      @ If so, we are done

    ldr r5, [r0]            @ Load value at r0
    cmp r5, r4              @ Is it larger than current?
    ble largest_skip        @ If not larger than current

    mov r4, r5              @ Make the new largest

largest_skip:

     add r0, r0, #4   @ Next array location
    sub r1, r1, #1          @ size -= 1
    b largest_loop          @ Loop back

largest_result:
    ldr r0,=largestMsg      @ Message to print

     mov r1, r4   @ Largest value

     bl printf   @ Print it using printf

largest_done:

     ldmfd sp!,{r4-r8,lr}

     bx lr   @ return
```