Name: SOLNS

CPE 133 Spring 2017 Final Exam

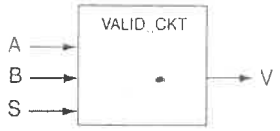
Closed book, notes, phone, calculator, tablet, pad, and laptop; feel free to use provided cheatsheets. Show your work and state any assumptions you make. Point value of problems is listed after problem number (100 pts).

Design Problem Directions: Unless the problem states otherwise:

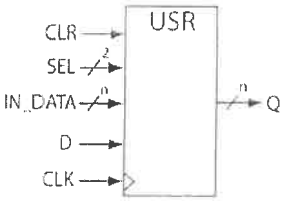
- Minimize the hardware in your designs, including not using a FSM if you don't need to
- Minimize the number of states in your FSMs when you use one
- For problems that require FSMs, you can use either a Moore or Mealy-type
- Use only standard digital modules in your designs; ***Don't use VHDL for anything***
- If the problem allows you to use a decoder, define it using tabular format

Provided Modules: You can use the following modules without providing lower-level descriptions

- **VALID_CKT** indicates validity based on the circuits three inputs
- **2sComp** outputs the 2's compliment of the input value



- **USR:** Use only the following model for any shift register you use in your problems

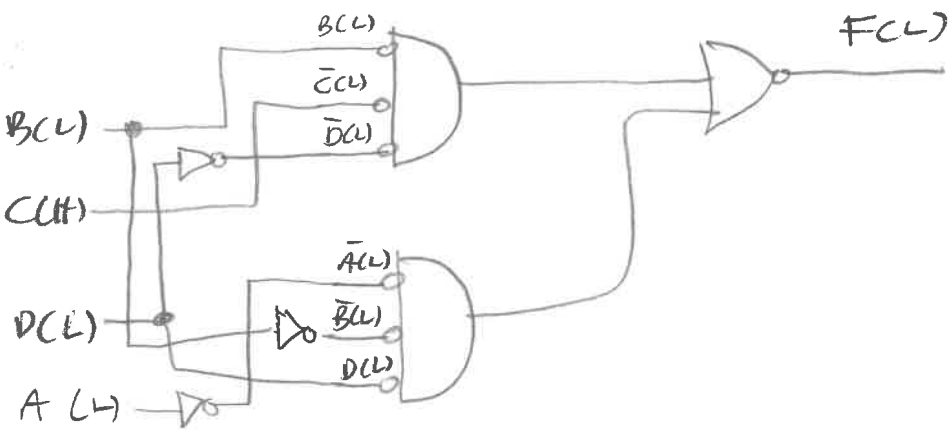


Shift Register Controls	
SEL	**Operation**
0 0	hold
0 1	parallel load
1 0	shift right
1 1	shift left

---

1. (5) Design a circuit that implements the following equation. Consider the inputs as A(L), B(L), C(H), and D(L). Use only NOR gates in your design.

$$F(A,B,C,D)(L) = \left[(B \cdot \overline{C} \cdot \overline{D}) + (\overline{A} \cdot \overline{B} \cdot D)\right](L)$$



2. (10) Design a circuit that indicates when two of the circuit's three inputs are asserted. Assume all inputs and outputs are positive logic. Provide the following models for your answer:

- Standard SOP form (equation)
- Standard POS form (equation)
- NAND/NAND form (equation)
- NOR/NOR form (equation)

A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

$F = \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$ SOP

$\bar{F} = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C}$

$F = (A+B+C)(A+B+\bar{C})(A+\bar{B}+C)(\bar{A}+B+C)$ POS

$F = \overline{\bar{A}BC + A\bar{B}C + AB\bar{C} + ABC}$

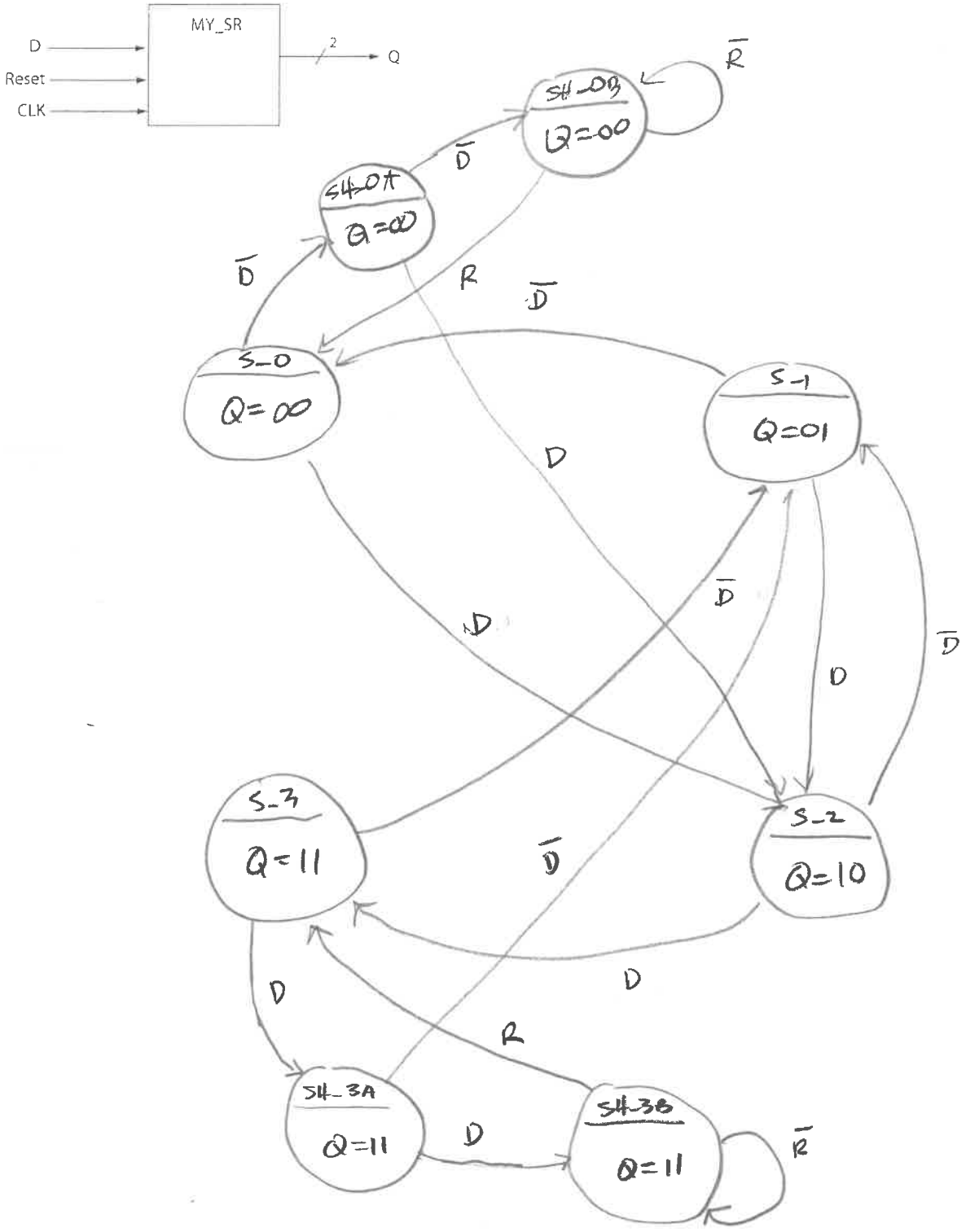
$F = \overline{(\bar{A}BC)}\,\overline{(A\bar{B}C)}\,\overline{(AB\bar{C})}\,\overline{(ABC)}$ NAND/NAND

$F = \overline{(A+B+C)(A+B+\bar{C})(A+\bar{B}+C)(\bar{A}+B+C)}$

$F = \overline{(A+B+C)} + \overline{(A+B+\bar{C})} + \overline{(A+\bar{B}+C)} + \overline{(\bar{A}+B+C)}$ NOR/NOR

3. (10) Provide a state diagram that models the operation of a 2-bit right-shifting shift register. For this design, if the shift register has the same output for two clock edges, the shift register goes into a stuck state and stay there until the RESET signal is asserted. The RESET signal is ***synchronous*** and is only active when the FSM is in a "stuck" state. When the FSM is in a stuck state, the shift register outputs are the same output as the associated non-stuck state. The RESET signal sends the FSM back to the unstuck state with the same output as the stuck state.

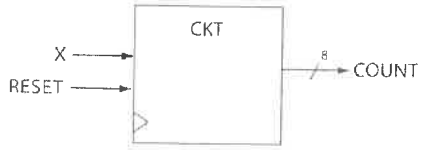
   - The D input is shifted into the shift register on the active clock edge
   - The Q output is the value of the shift register



4. (10) Design a circuit that counts how many times it detects the sequences "0101" and "1011" on a single serial input. When it detects the former sequence, it increments an 8-bit counter; when it detects the latter sequence, it decrements the same counter. If both sequences are simultaneously detected, the count holds. This circuit has an asynchronous reset that sends both FSMs back to their initial state in their associated sequence searches.

- The most straight-forward approach is to use two FSMs
- The serial input does not change more than once per clock cycle.

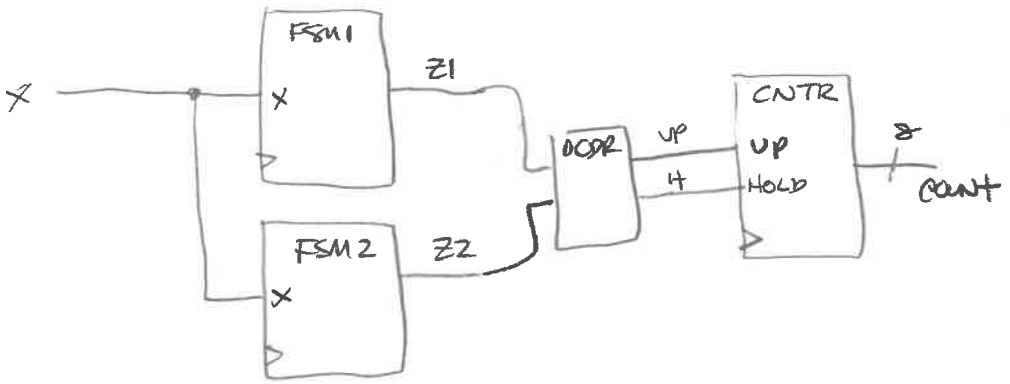
ASSUME FSMs Resets

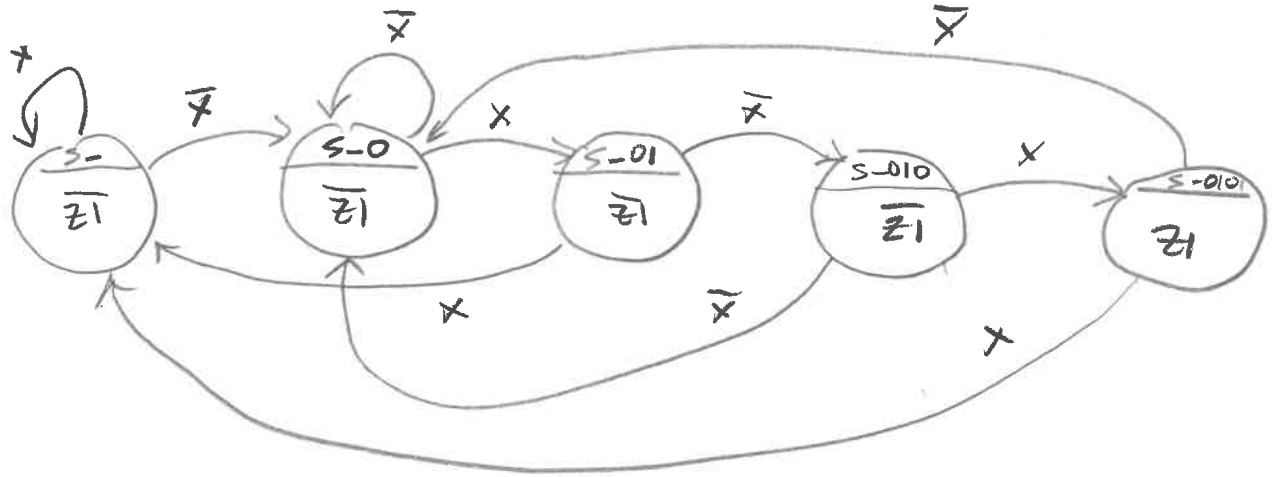


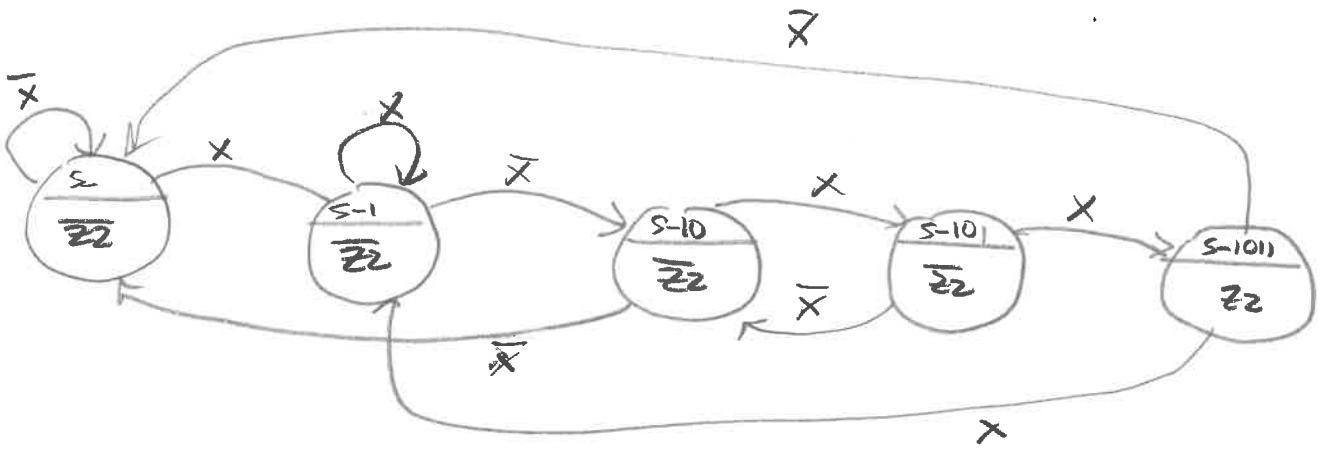
UP=0 & increment
UP=1 : decrement

Z1	Z2	H	UP
0	0	1	0
0	1	0	0
1	0	0	1
1	1	1	0

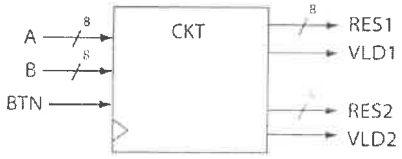
DECODER DEFINT

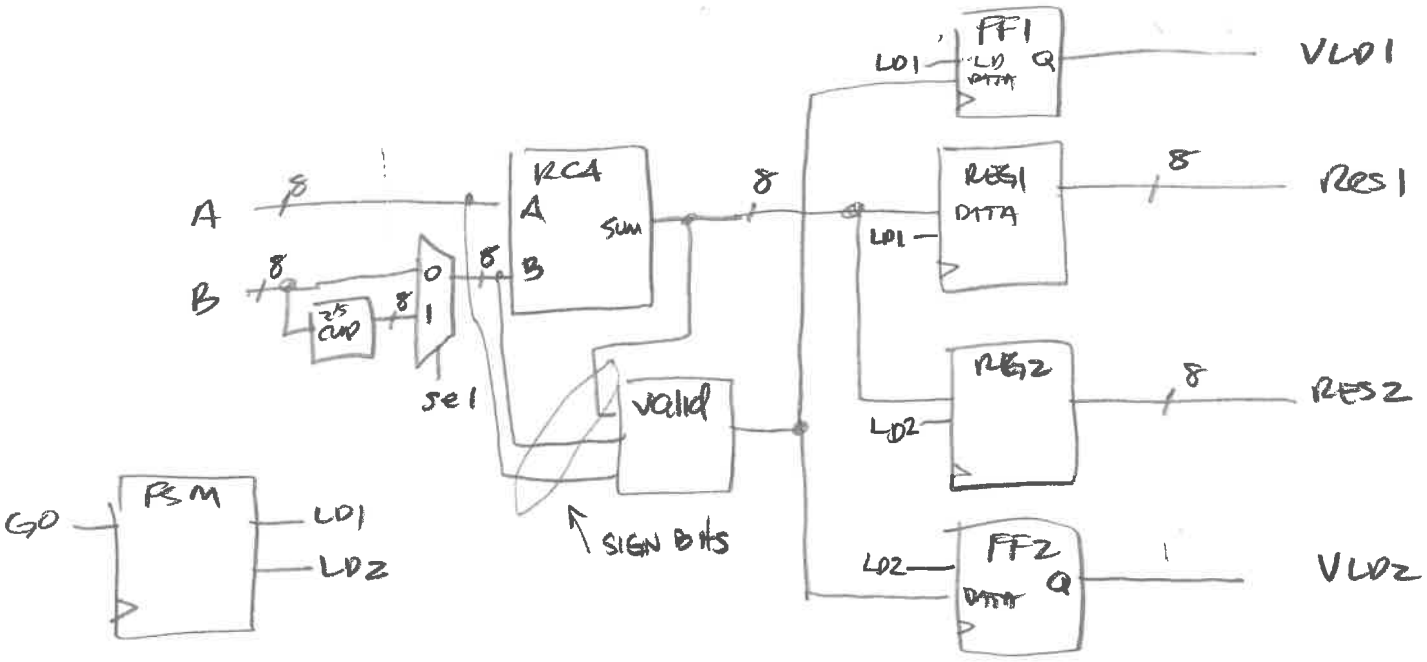


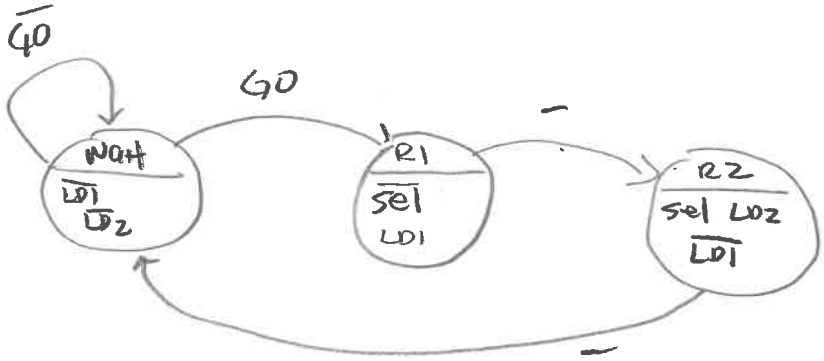




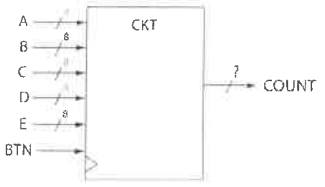
5. (15) Design a circuit that outputs the following two values when the circuit detects a button press on the circuits active clock edge: A+B & A-B. These two results are persistent until the button is pressed again. Consider A & B to be 8-bit signed binary numbers in RC format. Include two LED outputs that indicate when each of the two output values is valid. ***Use no more than one RCA in your design.*** Assume that the A & B inputs remain on the inputs long enough to complete the calculations.

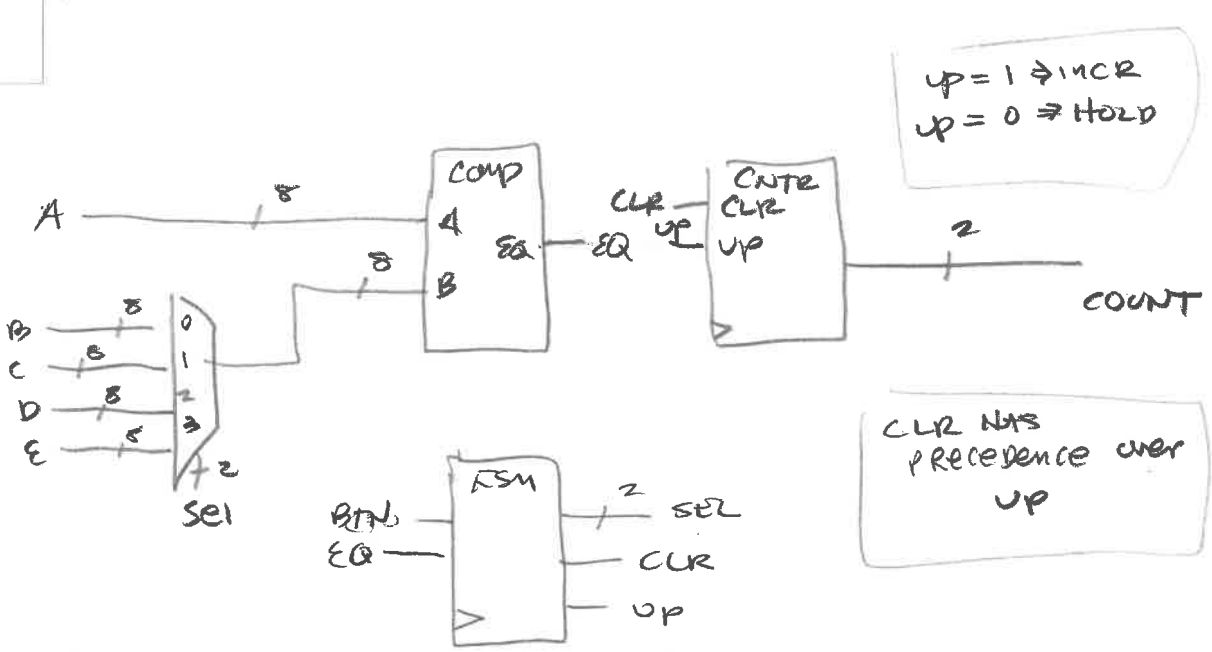


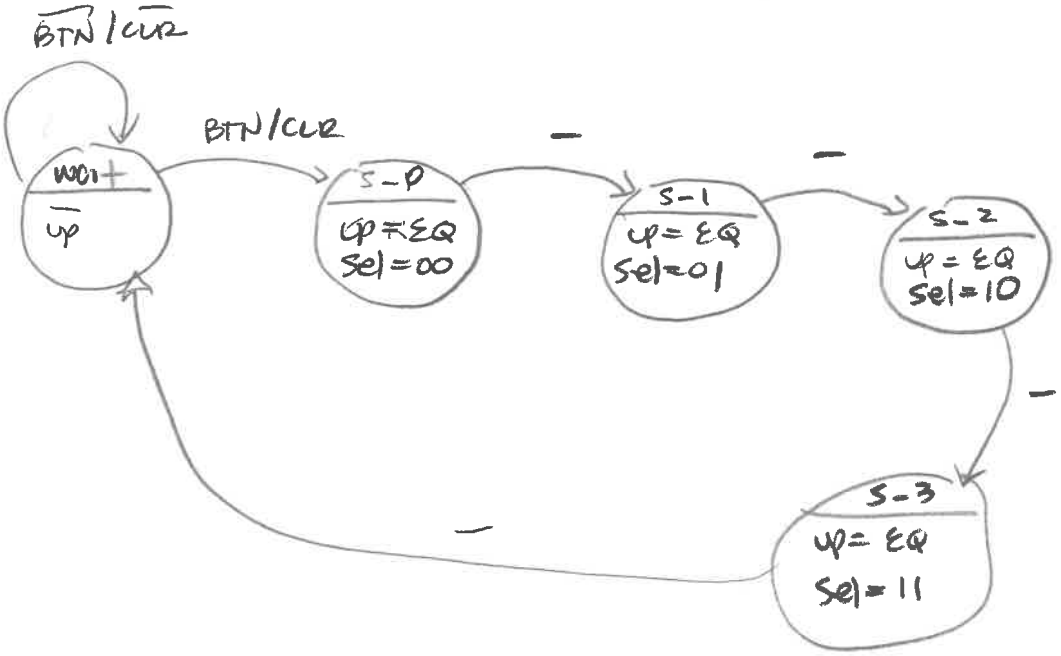




6. (10) Design a circuit that has five 8-bit signed binary inputs in RC format. When a button is pressed, the circuit displays the number of values that are equal to A (one of the five inputs) in unsigned binary format, then waits for another button press before the operation repeats itself. ***Use no more than one comparator in your design.***

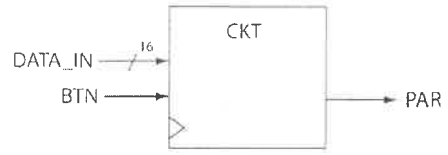


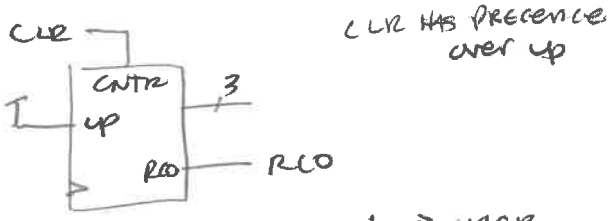


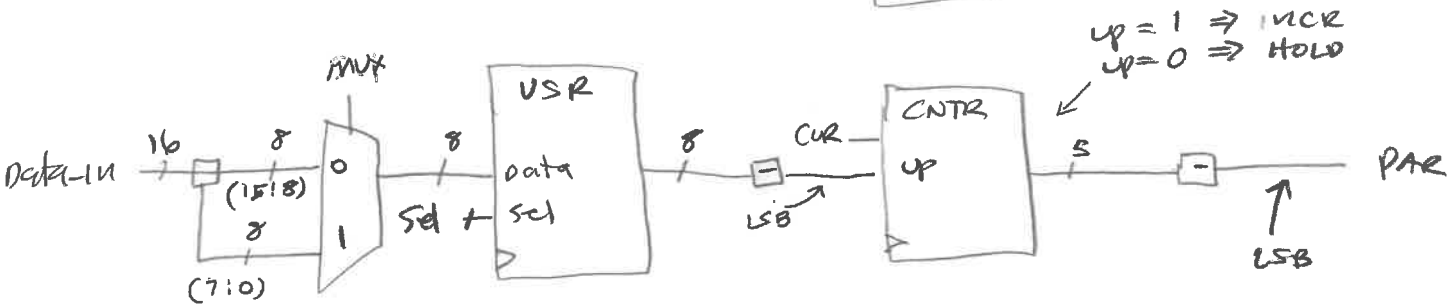


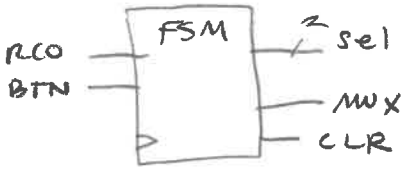
CLR IS NOT ASSERTED IN ALL STATES EXCEPT

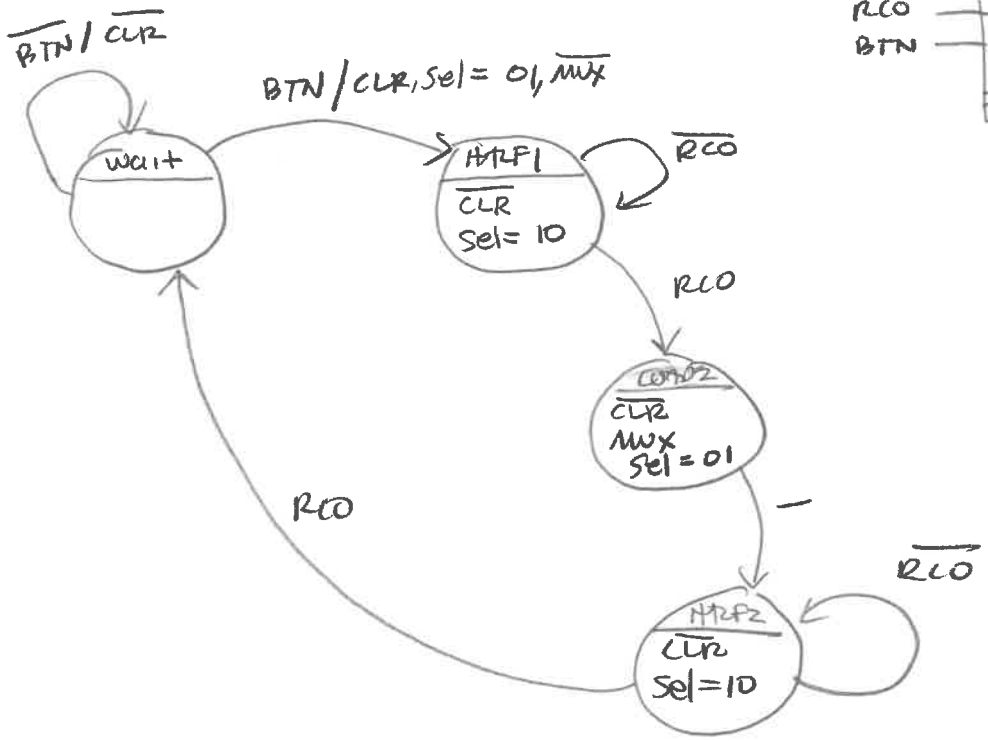
7. (15) Design a circuit that upon the press of a button, calculates the parity of a 16-bit value. The output of the circuit indicates parity of the 16-bit input and is persistent until another button press. ***The only shift register you can use in your design is an 8-bit shift register, and you can only use one of them.*** Don't use decoders or more than two XOR gates in your design.





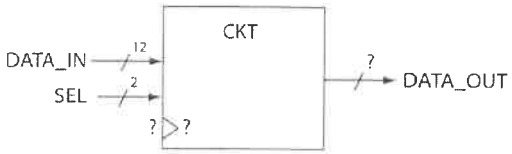






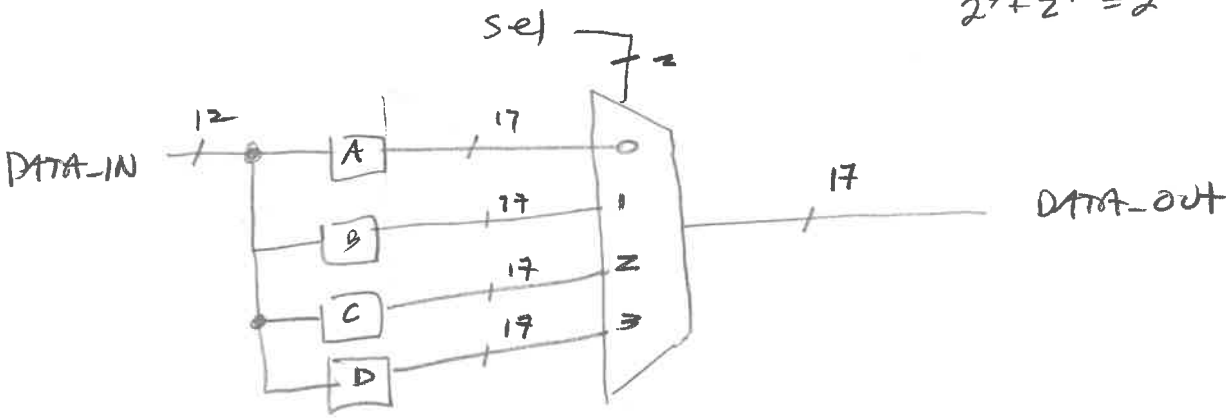
8. (10) Design a circuit with one 12-bit unsigned binary input and one output wide enough to support the following operations. The circuit's two bit input selects which value appears on the output according to the table below. For this problem, use truncation for the division operations, but the result of the multiplication operations should be exact. ***Don't use a shift register in your design.*** Strongly consider whether you can do this without a clock.

SEL	Operation
"00"	Divide by 4
"01"	Divide by 16
"10"	Multiply by 8
"11"	Multiply by 32



$32 = 2^5$

$2^5 + 2^{12} = 2^{17}$



A "0000000" & DATA_IN (11:2)

B "000000000" & DATA_IN (11:4)

C "00" & DATA_IN (11:0) & "000"

D DATA_IN (11:0) & "00000"

9. (15) Design a circuit that upon a button press, adds three non-zero input values present on the circuit's input on a rising clock edge. The circuit's data inputs are 10-bit unsigned binary values. The circuit's 10-bit output is a summation of the three values and an LED that indicates when the result of the summation is correct. The two outputs remain persistent after the circuit completes the calculation and awaits another button press. *Note: this circuit does not consider input values of zero to be one of the three summed values.*