King Fahd University of Petroleum and Minerals
College of Computer Science and Engineering
Computer Engineering Department

COE 202: Digital Logic Design (3-0-3)

Term 151 (Fall 2015)

Final Exam

Sunday Dec. 20, 2015

7:00 p.m. – 9:30 p.m.

Time: 150 minutes, Total Pages: 12

Name: _KEY________________________ ID: _____________ Section: ______

Notes:

- Do not open the exam book until instructed
- Calculators are not allowed (basic, advanced, cell phones, etc.)
- Answer all questions
- All steps must be shown
- Any assumptions made must be clearly stated

Question	Maximum Points	Your Points
1	12	
2	14	
3	10	
4	8	
5	9	
6	7	
Total	60	

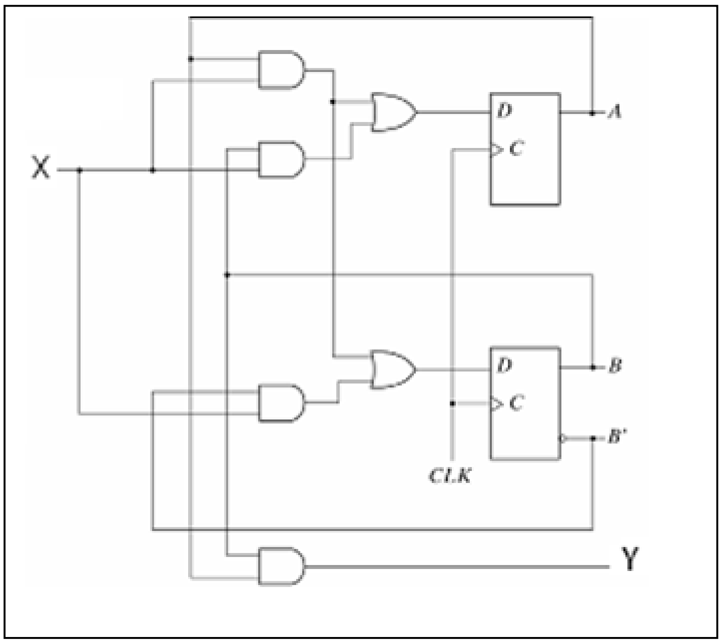
Question 1. **(12 Points)**

The shown synchronous sequential circuit has a single input X and a single output Y. Answer the following questions:

(i) The circuit is __MOORE___ (Moore / Mealy) **(1 point)**

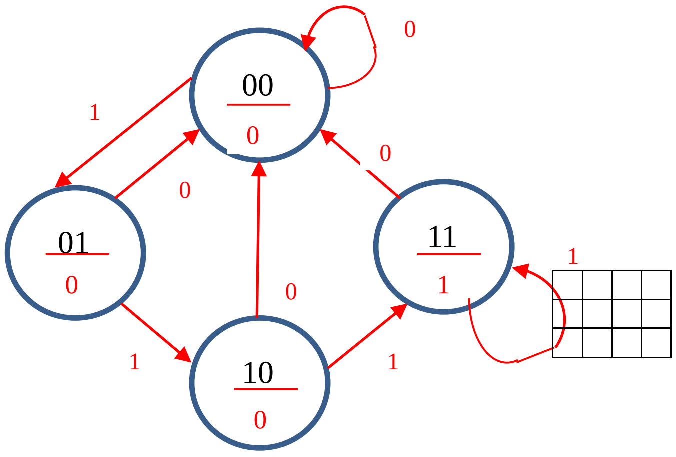
(ii) Derive Boolean expressions for the flip-flop D-inputs, i.e. D_A and D_B and the output Y. **(3 points)**

D_A= XA + XB=X.(A+B)
D_B= XA + XB'=X.(A+B')
Y = AB



(iii) Fill in the state table of this circuit and draw the corresponding state diagram **(6 Points)**

PS	**NS (A⁺ B⁺)**		**Y**	
AB	**X=0**	**X=1**	**X=0**	**X=1**
0 0	0 0	0 1	0	0
0 1	0 0	1 0	0	0
1 1	0 0	1 1	1	1
1 0	0 0	1 1	0	0



(iv) For a starting state of **(10)**, what is the resulting output sequence (**Y**) for an applied input sequence (**X**) of **{1→ 1→0}? (2 points)**

PS	(10)	(11)	(11)	(00)
X	1	1	0	Don't care
Y	0	1	1	0

Question 2. **(14 Points)**

Consider the following state transition table for a synchronous sequential circuit that increments a binary number by two, i.e. Z=X+2. The circuit has a single input **X**, a single output **Z**, and two state variables **Y_1**, and **Y_0**. The states are encoded using binary codes **00**, **01**, **10.** The **reset state** is **10.**

PS (Y_1 Y_0)^t	**NS ($Y1$ Y_0)$^{t+1}$**		**Z**	
	X = 0	**X = 1**	**X = 0**	**X = 1**
0 0	0 0	0 0	0	1
0 1	0 0	0 1	1	0
1 0	0 1	0 1	0	1

(i)

a. Using D-FFs and <u>**minimal**</u> combinational logic, determine the equations for the D-FF inputs and the output Z for this circuit.

b. Draw the resulting circuit and add the required logic to achieve **Synchronous Reset**, to reset the machine to state Y_1Y_0=**10** when a Reset input is asserted.

(7 points)

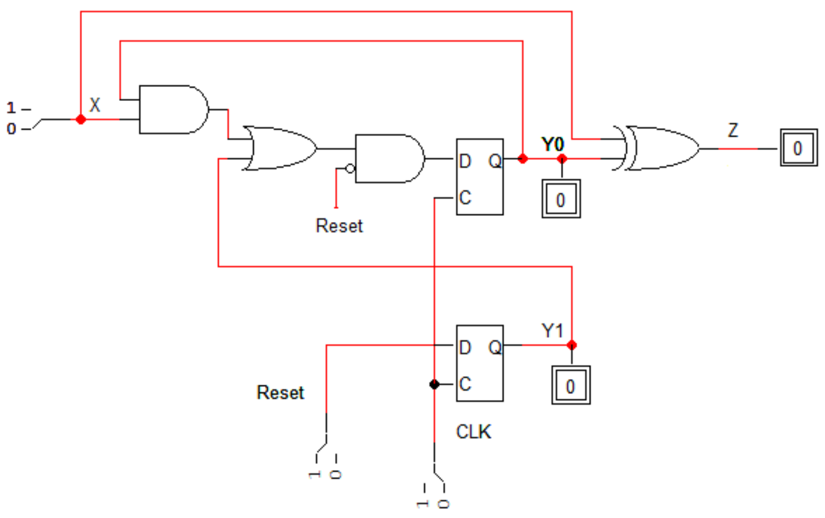
	00	**01**	**11**	**10**
0	0 0	1 1	? 3	0 2
1	1 4	0 5	? 7	1 6

Z = X' Y0 + X Y0' = X ⊕ Y0

	00	**01**	**11**	**10**
0	0 0	0 1	? 3	1 2
1	0 4	1 5	? 7	1 6

D0 = Y1 + X Y0

D1 = 0

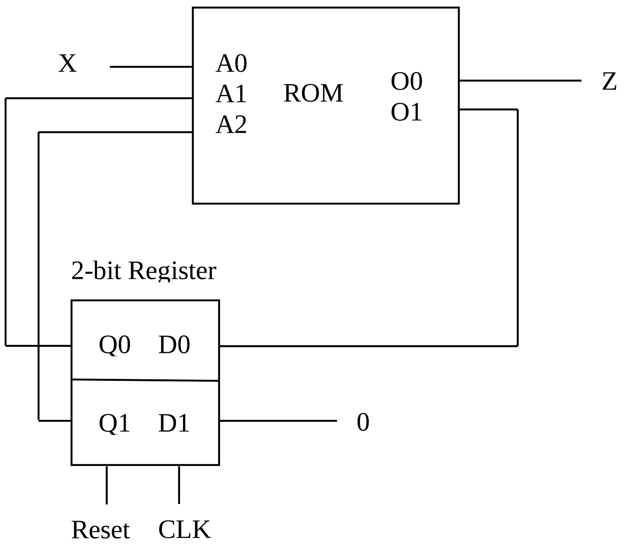


(ii) You are required to implement the above circuit using a **ROM** and a **register** with minimum sizes.

a. What is the minimum size of the ROM (number of memory locations × number of memory bits per location)? **(2 points)**

Since D1=0, we do not need to store it. Thus, the ROM size is 2^3 x 2 bits

b. Draw the block diagram for such implementation. Add **Asynchronous Reset** to the register to reset the machine to state Y_1Y_0=**10**. (**Label all components inputs and outputs together with various signals**) **(3 points)**



Note that the Reset will be connect to Set input for Q1 and to CLR inputs for Q0.

c. Starting from address **0,** fill in the following table to show the data stored in the first four memory locations in the ROM device. **(2 points)**

Binary Address	**Binary Stored Data**
000	00
001	01
010	01
011	10

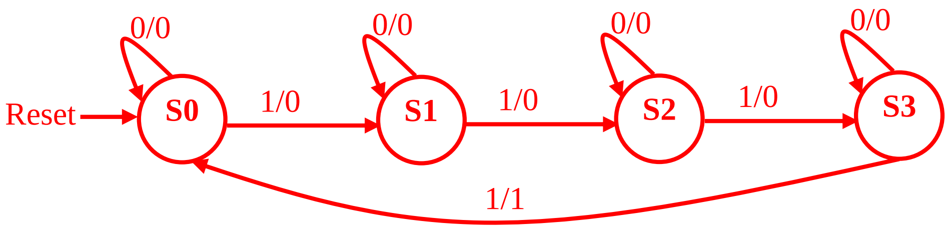
Question 3. **(10 Points)**

(i) Using minimum number of states, show the state diagram of a circuit that counts the number of 1's in an input stream (not necessarily consecutive). The circuit has one input X and one output Y. Y remains 0 until 4 ones are received on X, then Y becomes 1 and the circuit starts counting the number of 1s in the input stream again and Y return to 0. The asynchronous Reset input resets the circuit to an initial state where no 1 has been received yet. An example of an input sequence and the corresponding output sequence is shown below: **(5 points)**

t = 0 ↓ time ⟹

X	0 0 1 0 1 1 0 0 1 0 0 1 1 0 1 0 1
Y	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1

(**NOTE**: You are *only* required to draw the state diagram **Nothing MORE**)

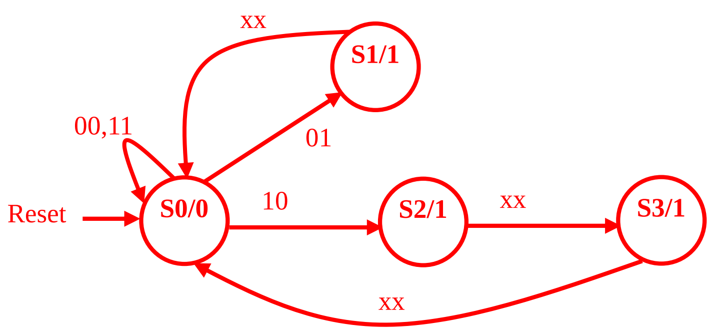


(ii) Show the **Moore-Model** state diagram of a sequential circuit that has two inputs X_1, X_2 and one output Z. The asynchronous Reset input resets the circuit to an initial state with Z=0. If $X_1=X_2$ (i.e. X_2X_1 = 00 or 11), the circuit remains at the initial state. While in the initial state, if X_2X_1 become 01, the circuit produces an output of 1 then goes back to the initial state. If X_2X_1 become 10, the circuit produces the output sequence {1,1} then goes back to the initial state. **If the inputs change while the circuit is producing an output sequence, these changes are ignored until the circuit goes back to the initial state.** An example of an input sequence and the corresponding output sequence is shown below: **(5 points)**

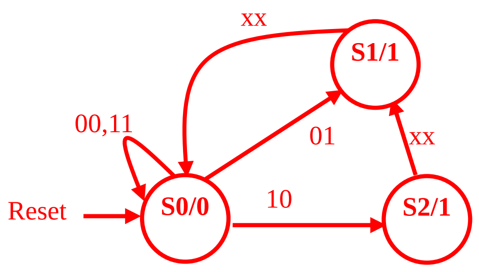
$\boxed{t = 0}$ time $\Rightarrow$

X_2	0 1 1 1 1 0 0 0 1 …
X_1	0 1 0 0 0 0 1 1 0 …
Z	0 0 0 1 1 0 0 1 0 1 1 0 …

(**NOTE**: You are *only* required to draw the state diagram **Nothing MORE**)



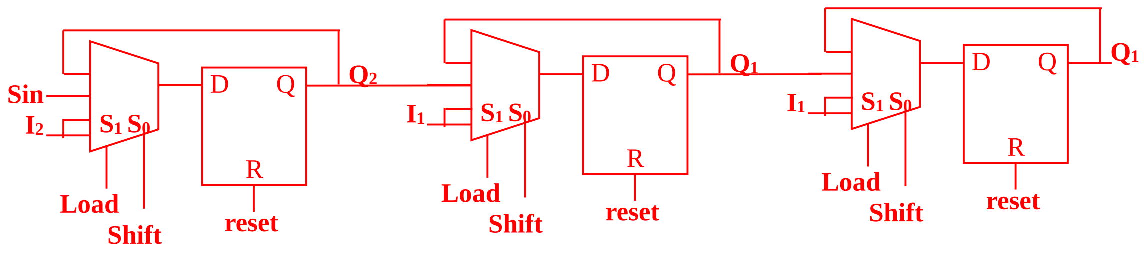
Another solution—S1 and S3 are the same



Question 4. **(8 Points)**

(i) Using minimum number of D-FFs and other needed components, show the design of a 3-bit register Q that has two control inputs: Load and Shift. The register has a 3-bit external input $I_2I_1I_0$ and a serial input S_{in}. The table below shows the functionality of the register. **(4 points)**

Load	Shift	Action
0	0	No change in Q
1	X	Load parallel input (i.e. $Q_2Q_1Q_0 \leftarrow I_2I_1I_0$)
0	1	Shift Q to the right with S_{in} inserted from the left (i.e., most significant bit)



(ii) Complete the following table by showing the content of register Q after each clock cycle: **(4 points)**

Inputs in a cycle affect the register in the next cycle

Clock #	Load	Shift	S_{in}	D_2 D_1 D_0	Q_2 Q_1 Q_0
1	0	0	0	0 0 0	0 0 0
2	1	0	0	1 0 1	0 0 0
3	0	1	1	1 1 1	1 0 1
4	0	0	1	0 0 1	**1 1 0**
5	0	1	0	0 0 0	**1 1 0**
6	1	1	0	1 0 0	**0 1 1**
7	0	0	1	1 1 0	**1 0 0**

Question 5 **(9 Points)**

In this question, you are to use only a 4-bit register and MSI parts, and minimum number of gates.

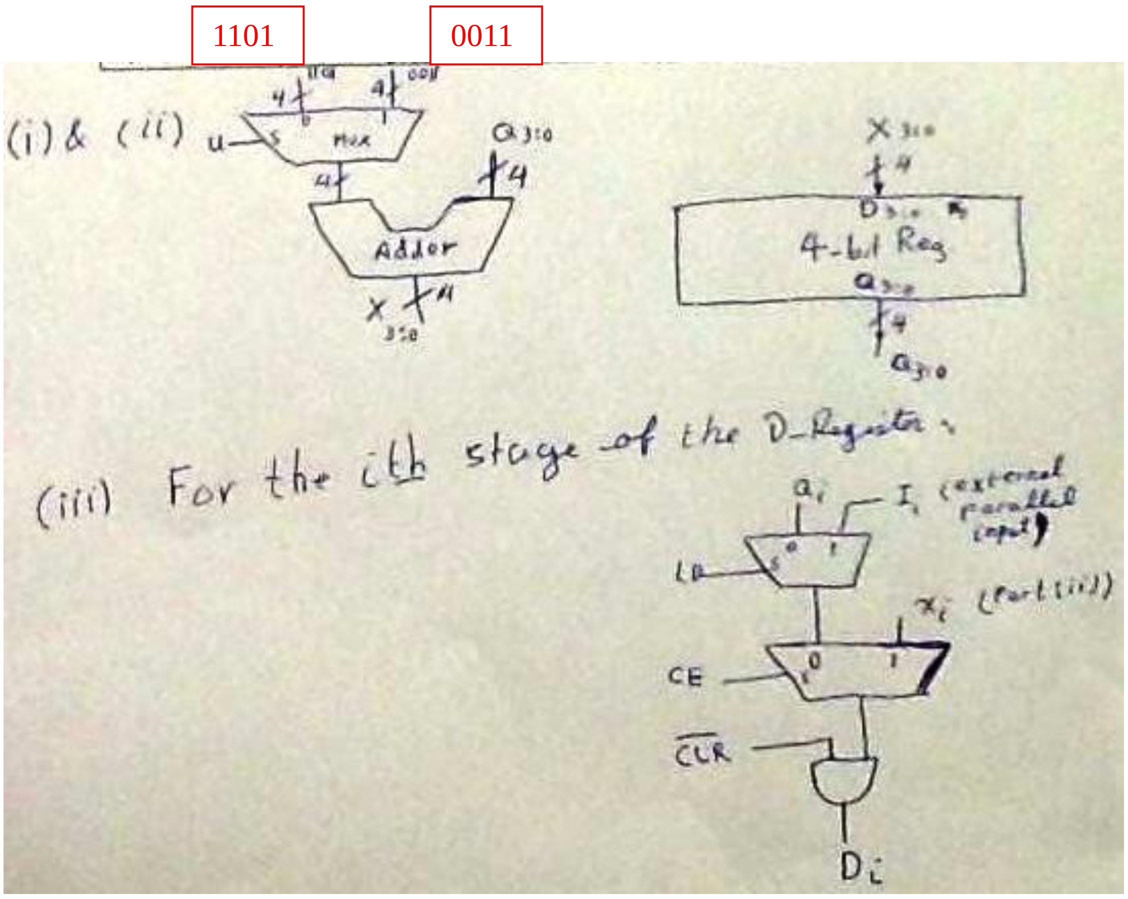
(i) Design a modulo-16 counter that can increment by 3, i.e. (0 → 3 → 6 …13 → 0). **(3 points)**

(ii) Show how to modify the above counter such that it can count either up (by 3) or down (by 3) based on the value of an additional control input **U (**if U=1 counting is up otherwise it is down**)** **(2 points)**

(iii) Show how to modify the above counter to have synchronous clear, parallel load and count enable capabilities as follows: **(4 points)**

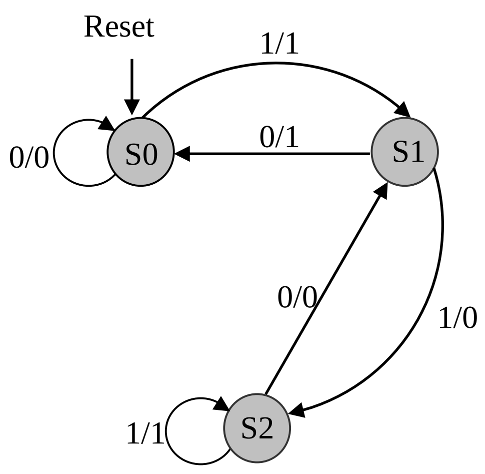
CLR CE LD	**Operation**
1 X X	Counter is Cleared
0 1 X	Counting (Up or down depending on the value of U)
0 0 1	Parallel Load of external input Data
0 0 0	No Change

Note that you have the option to show the final design to implement all the requirements without showing a step-by-step design.



Question 6 **(7 Points)**

Write a Verilog module for modeling the behavior of the sequential circuit represented by the state diagram given below which has a single input X and a single output Z. Assume the state assignment: S0=00, S1=01, and S2=10. Assume the availability of **Synchronous Reset** input that resets the machine to state S0.



```
module FSM(output reg Z, input X, Reset, CLK);

parameter S0 = 2'b00, S1=2'b01, S2=2'b10;
reg [1:0] state, next_state;

always @(posedge CLK)
  if (Reset) state <= S0; else state <= next_state;

always @(state, X) begin
Z=0;
case (state)
      S0:  if (X)  begin Z=1; next_state=S1; end else next_state=S0;
      S1:  if (X)  next_state=S2; else begin Z=1; next_state=S0; end
      S2:  if (X) begin Z=1;  next_state=S2; end else next_state=S1;
      default: begin  Z='bx; next_state='bx; end
   endcase
end
endmodule
```

Verilog Primitives

- Basic logic gates only
  - and
  - or
  - not
  - buf
  - xor
  - nand
  - nor
  - xnor

These gates are expandable: 1st node is O/P node, followed by 1, 2, 3 … number of input nodes

Verilog Operators

{ }	concatenation
+ - * / **	arithmetic
%	modulus
> >= < <=	relational
!	logical NOT
&&	logical AND
||	logical OR
==	logical equality
!=	logical inequality
===	case equality
!==	case inequality
? :	conditional

~	bit-wise NOT
&	bit-wise AND
|	bit-wise OR
^	bit-wise XOR
^~ ~^	bit-wise XNOR
&	reduction AND
|	reduction OR
~&	reduction NAND
~|	reduction NOR
^	reduction XOR
~^ ^~	reduction XNOR
<<	shift left
>>	shift right