CALIFORNIA STATE UNIVERSITY
FULLERTON™

EGEC 180 – Digital Logic and Computer Structures

Spring 2024

Final Exam Review

Rakesh Mahto, Ph.D.
***Office**: E 314, California State University, Fullerton*
***Office Hour**: Monday and Wednesday 2:00 - 3:30 pm*
Or by appointment
***Office Hour Zoom Meeting ID**: 891 2907 5346*
***Email**: ramahto@fullerton.edu*
***Phone No**: 657-278-7274*

Final Exam

- Day: Monday, 13 May 2024
- Time: 11:00 – 12:50 PM
- Lecture Slides from 1 to 14
- Bring a cheat-sheet of one page (A4 Size)- both side, pen, pencil and calculator.

Conversion

- 8-4-2-1 Code (BCD)
- 6-3-1-1 Code
- Excess-3 Code
- 2-out of 5 Code
- Gray Code
- Binary
- Octal
- Hexadecimal

Arithmetic Operations

- Adding two Binary, Octal, Hexadecimal number
- Subtraction of two Binary, Octal, Hexadecimal number
- Subtraction using 1’s and 2’s Complement
- Multiplication
- Division

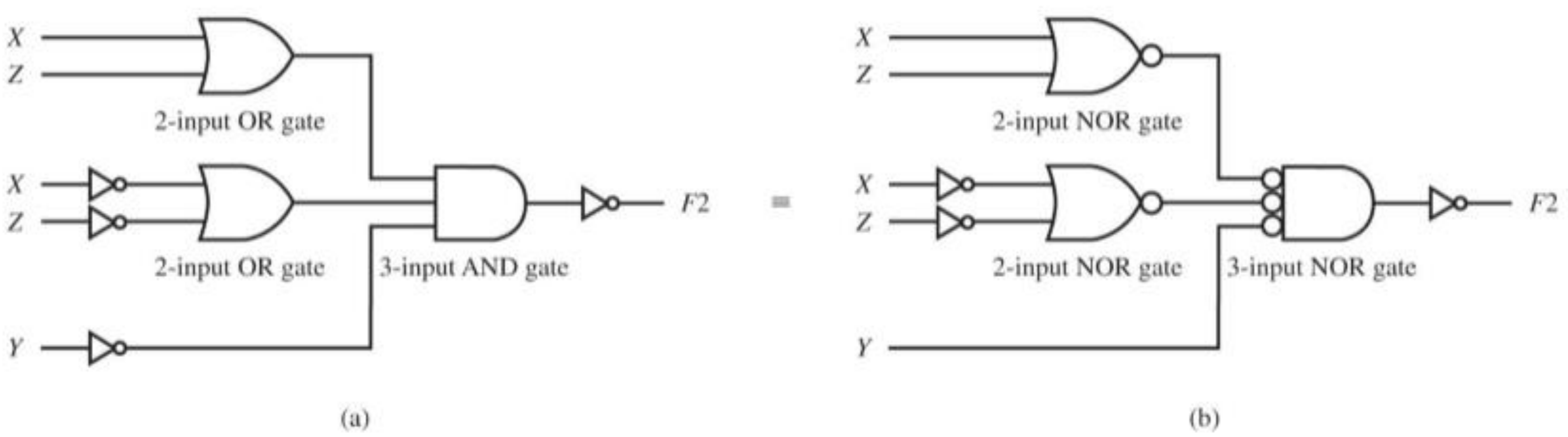
Boolean Algebra

- Boolean Functions
- Truth Table
- Logical Circuits
- Minterm
- Maxterm
- K-Map
- Circuit design using NAND Gate and NOR Gate.
- Decoder

Designing Circuit using NAND or NOR Gate

OR/AND to NOR/NOR

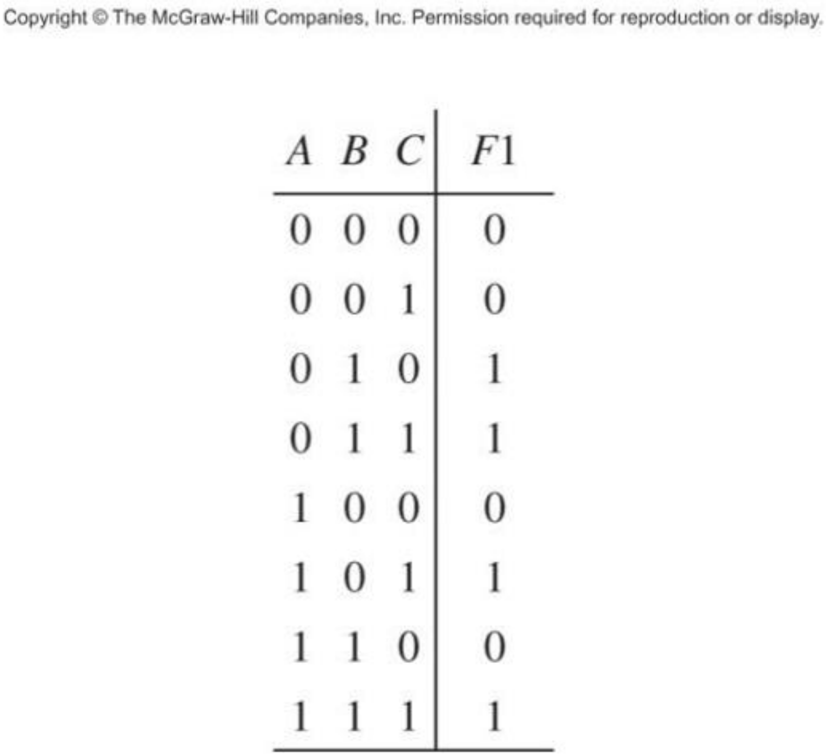
F2 = X'Z' + XZ + Y
F2' = (X + Y)(X' + Z')Y'



Decoders

Design F1 with a Decoder

F1 = $\Sigma m(2,3,5,7)$ - SOM



A	B	C	$F1$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Designing Logic Circuits with MUXs

Technique 1:

- For n variables, use n-1 variables as inputs to select lines. Hence, we need a 2^{n-1}-to-1 MUX.

Example: Given $F_1(X,Y,Z) = \Sigma\, m(1,2,5,7)$, 3 variables so we need a 2^2-to-1 MUX. Lets allocate X and Y to select lines, leaving Z for the Data Lines.

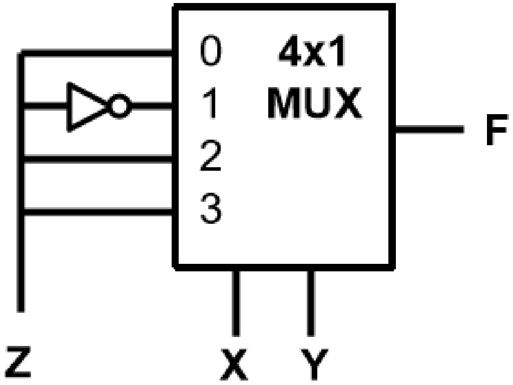
	X	Y	Z	F
I_0	0	0	0	0
	0	0	1	1
I_1	0	1	0	1
	0	1	1	0
I_2	1	0	0	0
	1	0	1	1
I_3	1	1	0	0
	1	1	1	1

Z Notice F and Z are the same

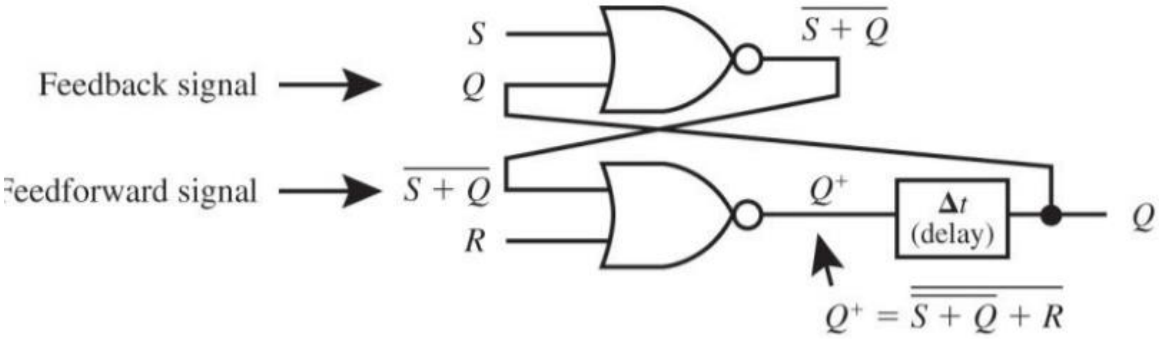
Z' Notice F and Z are Complemented

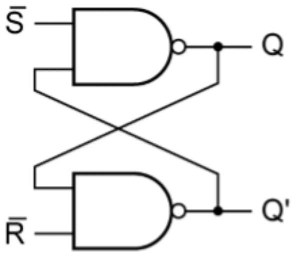
Z Notice F and Z are the same

Z Notice F and Z are the same

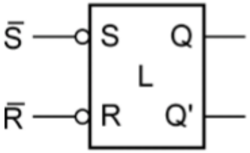


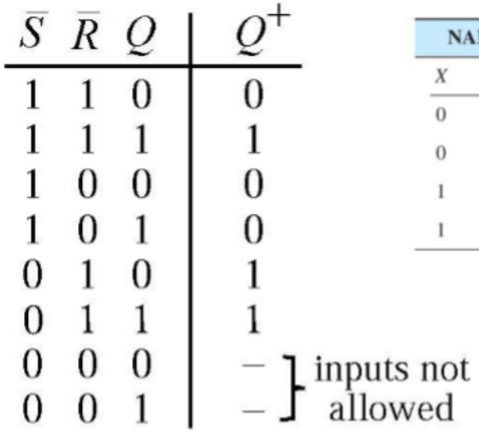
S-R Latch





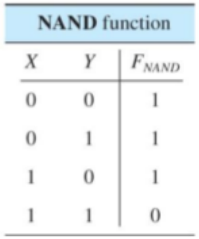
(a)





$\bar{S}$	$\bar{R}$	Q	Q^+
1	1	0	0
1	1	1	1
1	0	0	0
1	0	1	0
0	1	0	1
0	1	1	1
0	0	0	–
0	0	1	–

} inputs not allowed

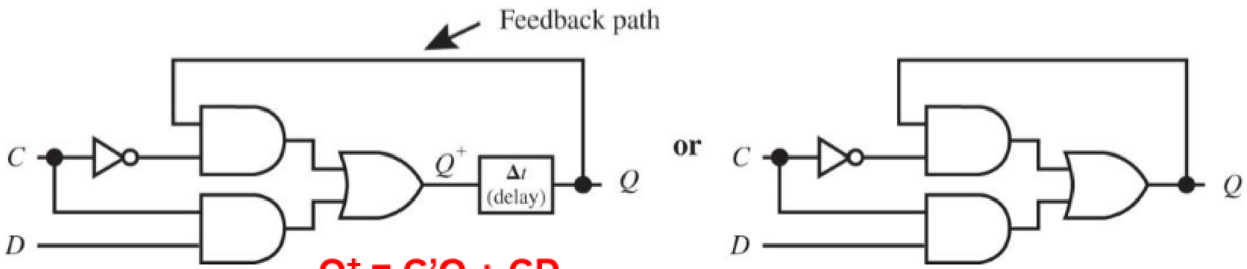


NAND function

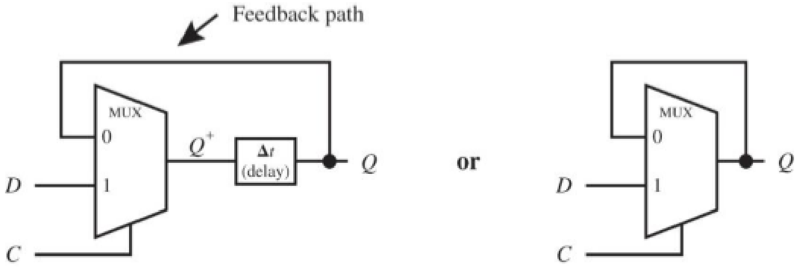
X	Y	F_{NAND}
0	0	1
0	1	1
1	0	1
1	1	0

NOTE: any time one of the inputs X or Y is a 0 the output F is a 1.

D-Latch



$Q^+ = C'Q + CD$



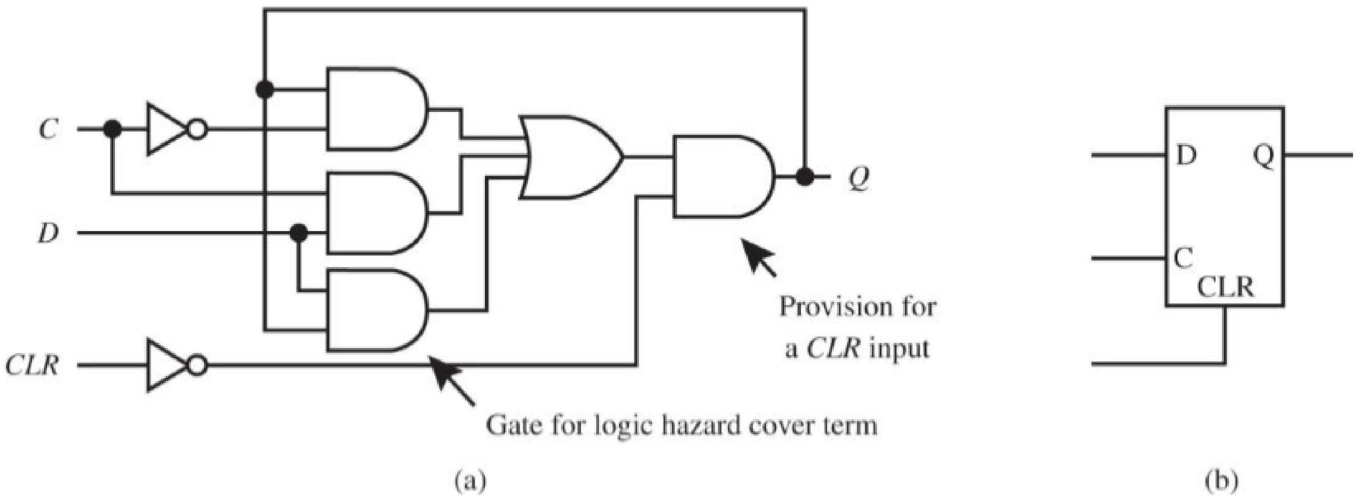
2-to-1 MUX Equation: Z (output) = $A'I_0 + AI_1$

A is select line, I_0 and I_1 are input lines.

$Q^+ = C'Q + CD$; $C = A$

D-Flip Flop

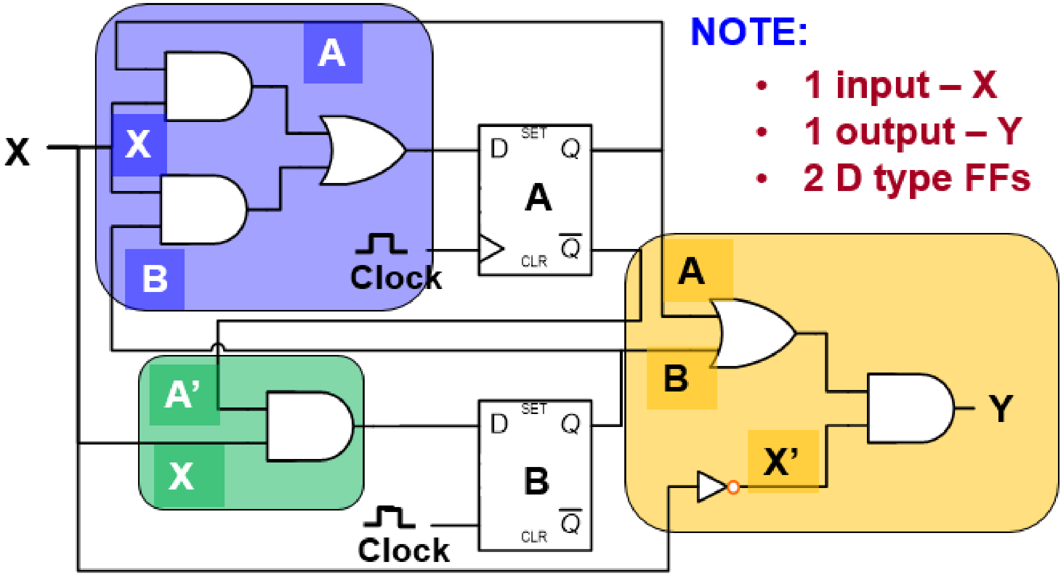
D Latch with a CLR Input: a) implemented with a logic hazard-free function in AND-OR circuit form, and b) logic symbol



Extra term removes the logic glitch: C'Q + CD + DQ

Mealy State Machine Analysis

Starting with the Logic Diagram (circuit)



NOTE:

- 1 input – X
- 1 output – Y
- 2 D type FFs

NOTE: This is a Mealy Model

Step 1) Write the state, input and output equations

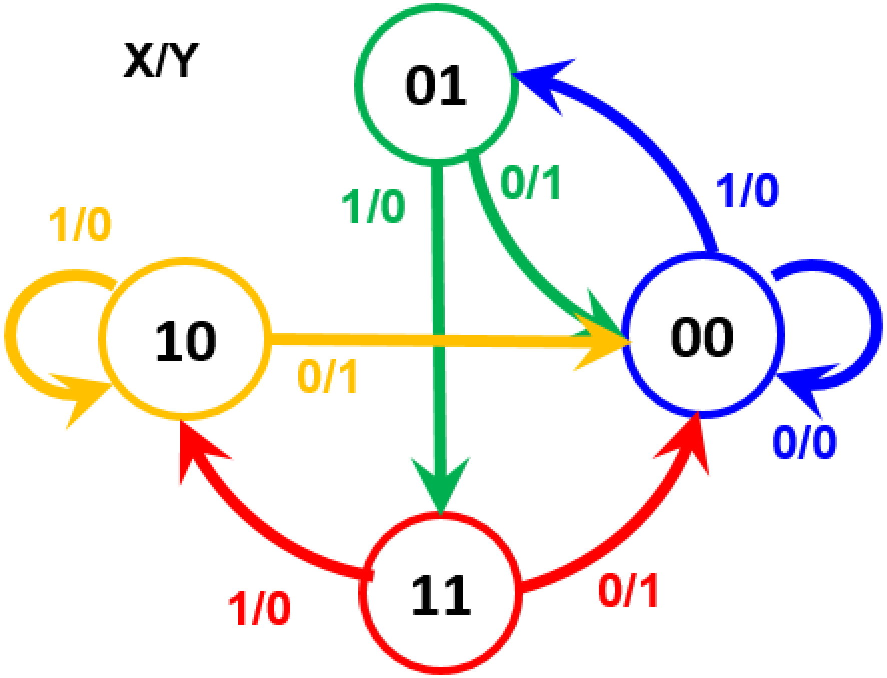
Output equations $Y = (A + B)X'$

Input equations $A^{+} = AX + BX$

$B^{+} = A'X$

Mealy State Machine Analysis (Continue)

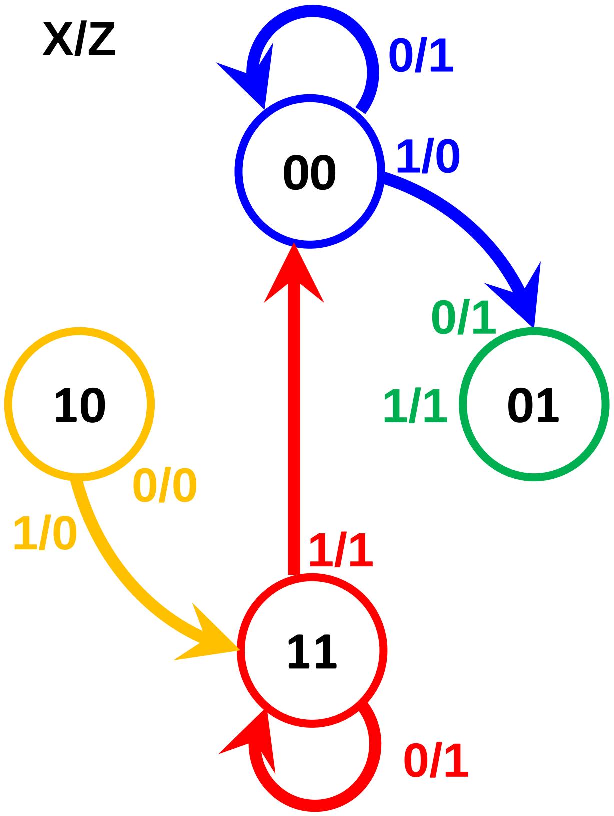
Step 3) Draw the State Diagram



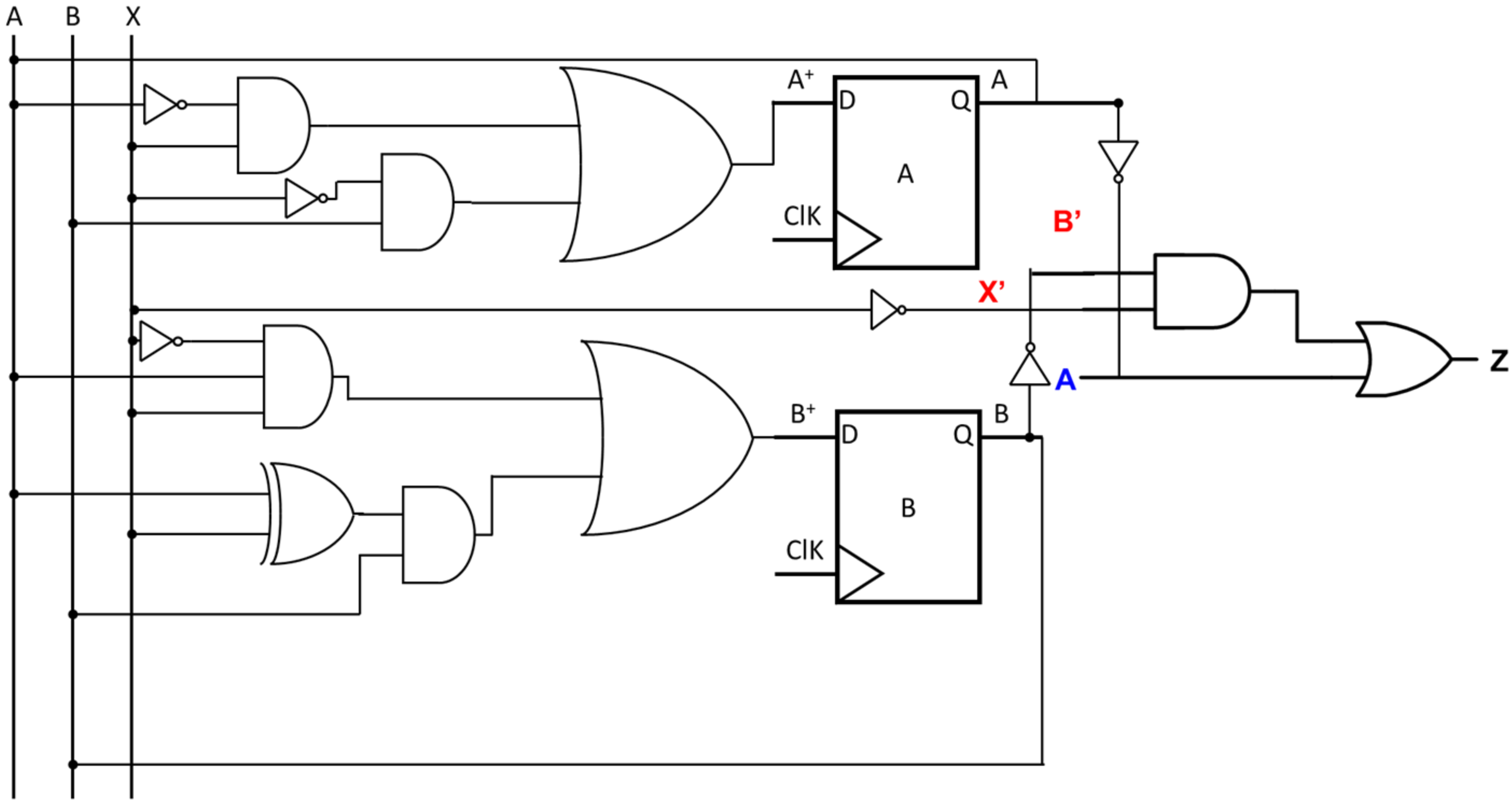
Present State		X	Next State		Y
A	B		A	B	
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Mealy State Machine Design

State Diagram to Design



Mealy State Machine Design (Continue)



Q&A