EGEC 180: – Digital Logic and Computer Structures

Spring 2024

Lecture 14: Midterm Exam No 2 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*

Mid-Term Exam No 2

- Day: 1 May 2014
- Time: 11:30 8:00 AM
- Lecture Slides from 8 to 13
- Bring a cheat-sheet of one page (A4 Size), pen, pencil and calculator.

OR/AND to NOR/NOR (Practice Problem #3)

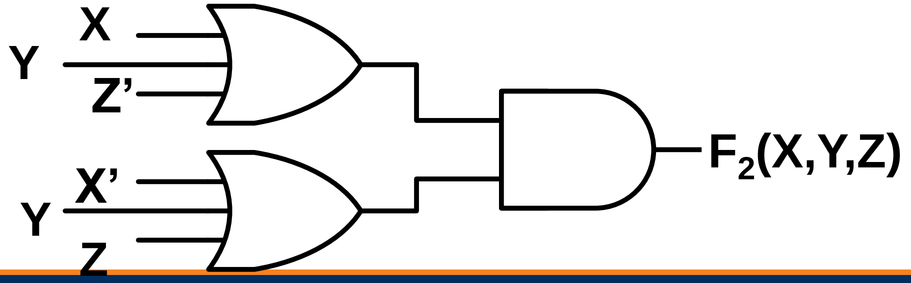
Given $F_2(X,Y,Z) = X'Z' + XZ + Y$

1. Find a minimum *product-of-sum* expression for F using a Karnaugh Map

X \ YZ	Y'Z' 00	Y'Z 01	YZ 11	YZ' 10
X' 0	1	0	1	1
X 1	0	1	1	1

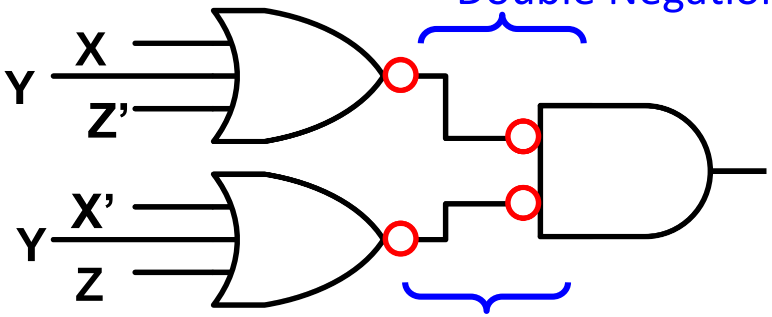
$F_2(X,Y,Z) = (X+Y+Z')(X'+Y+Z)$

2. Draw the corresponding two-level **OR-AND** circuit.



OR/AND to NOR/NOR (Practice Problem #3)

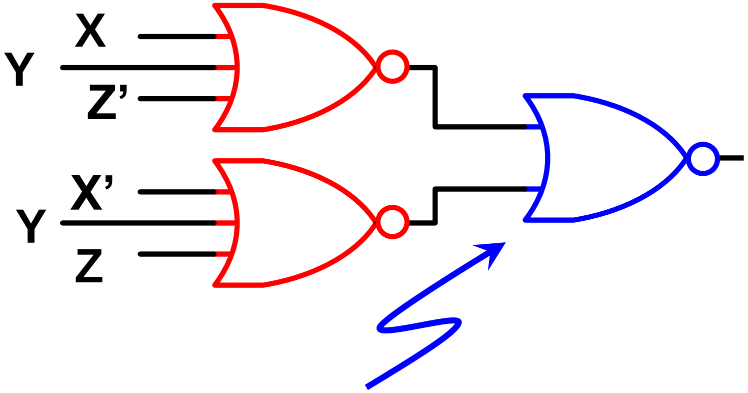
3. **Add bubbles to inputs and bubbles to outputs**



Double Negation

Double Negation

4. **Replace all gates with NOR gates leaving the gate interconnection unchanged.**



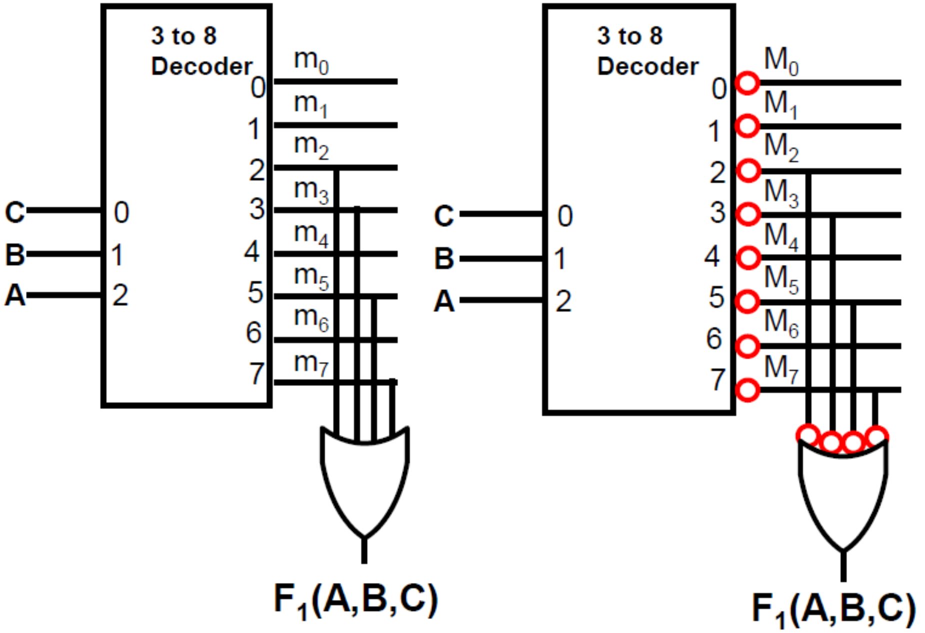
DeMorgan's Theorem

Decoders

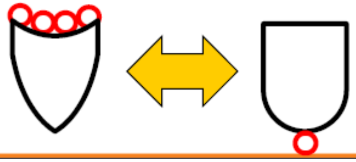
Design F1 with a Decoder

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



DeMorgan's Theorem

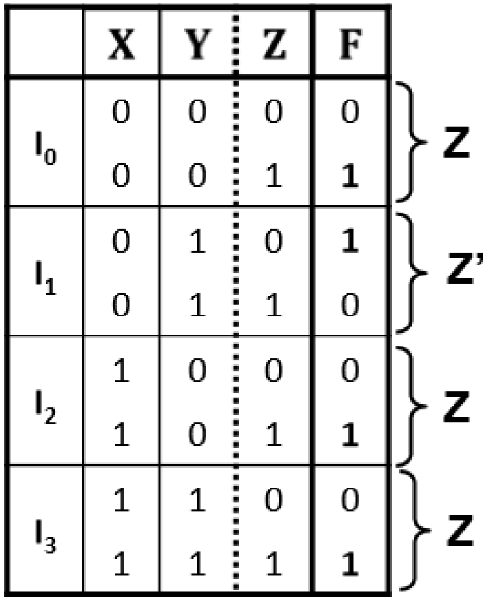


Designing Logic Circuits with MUXs

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.



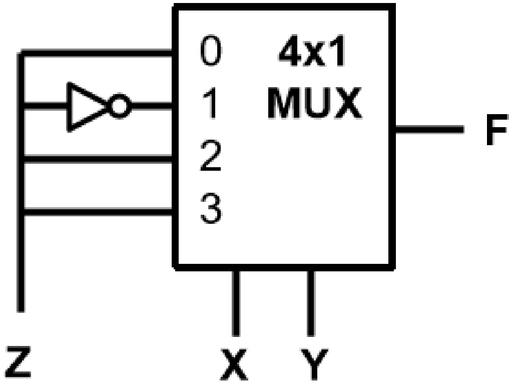
	X	Y	Z	F	
I_0	0	0	0	0	Z
	0	0	1	1	
I_1	0	1	0	1	Z'
	0	1	1	0	
I_2	1	0	0	0	Z
	1	0	1	1	
I_3	1	1	0	0	Z
	1	1	1	1	

Notice F and Z are the same

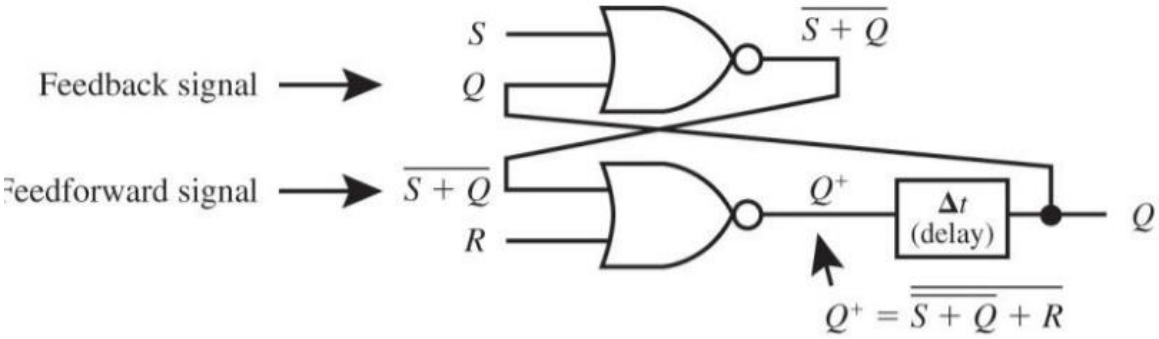
Notice F and Z are Complemented

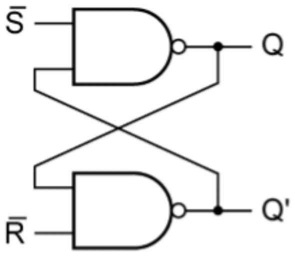
Notice F and Z are the same

Notice F and Z are the same

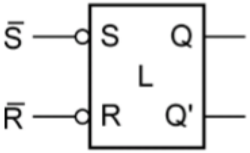


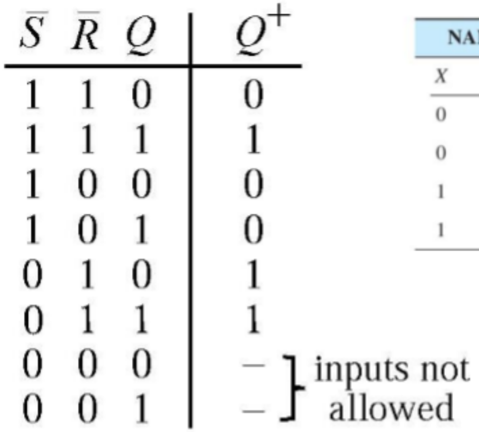
S-R Latch



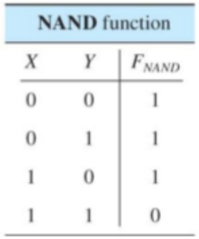


(a)





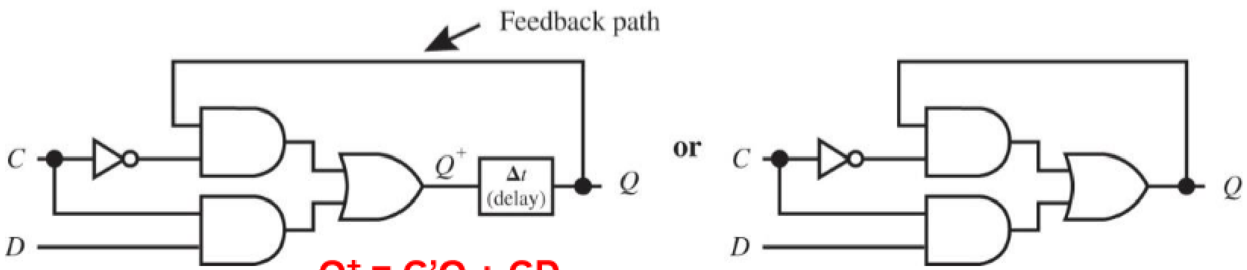
$\bar{S}$	$\bar{R}$	Q	Q^+	
1	1	0	0	
1	1	1	1	
1	0	0	0	
1	0	1	0	
0	1	0	1	
0	1	1	1	
0	0	0	–	inputs not allowed
0	0	1	–	



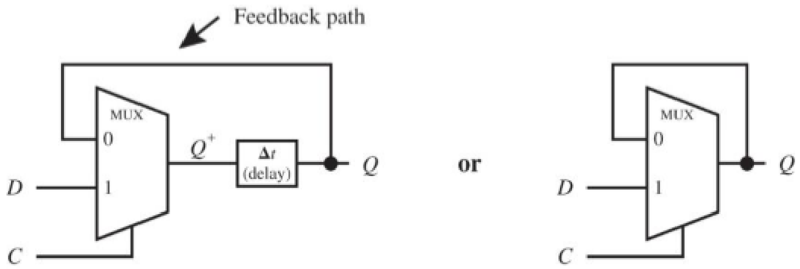
NAND function		
X	Y	F_{NAND}
0	0	1
0	1	1
1	0	1
1	1	0

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

D-Latch



$Q^+ = C'Q + CD$



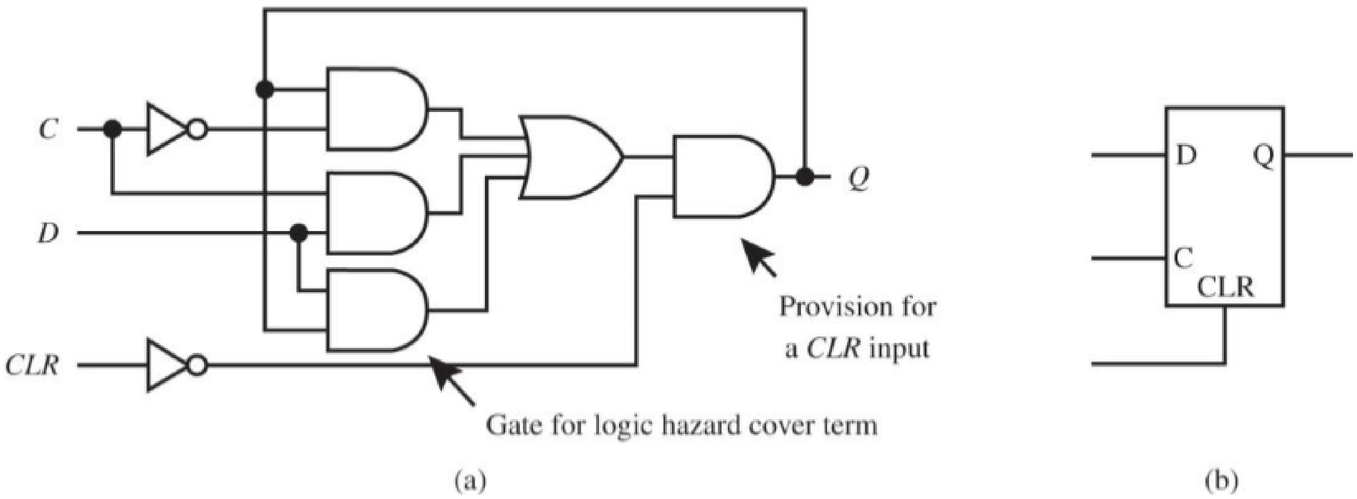
2-to-1 MUX Equation: Z (output) = $A'I_0 + AI_1$

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



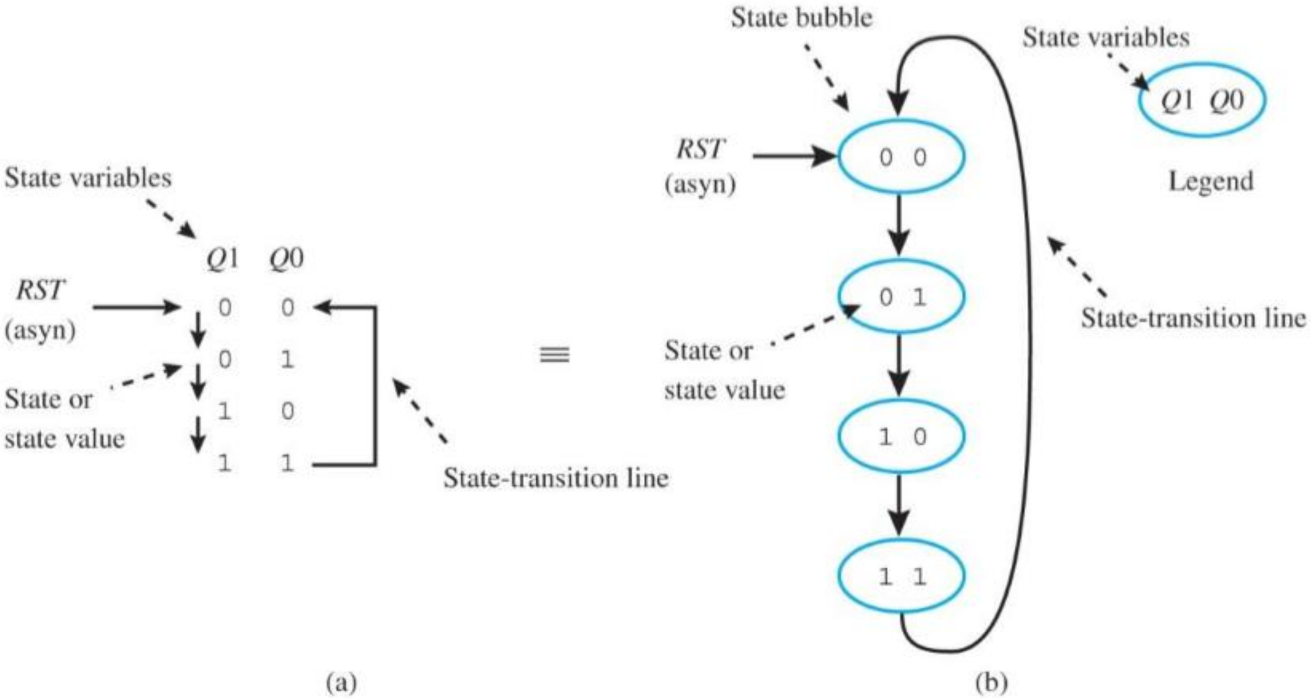
Extra term removes the logic glitch: C’Q + CD + DQ

Counter Design

Binary Up Counter (2 bits):

a) **Counting Sequence Diagram**

b) **Equivalent State Diagram**



Q&A