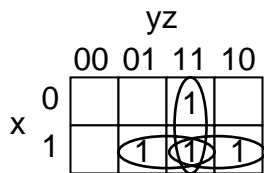


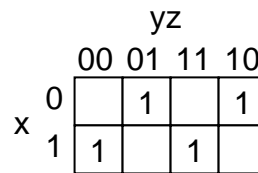
CS309 Spring 06 Homework 4

4-5

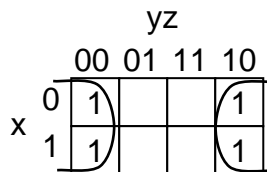
xyz	ABC
000	001
001	010
010	011
011	100
100	011
101	100
110	101
111	110



$A = xz + xy + yz$



$B = x \text{ XOR } y \text{ XOR } z$



$C = z'$

4-28

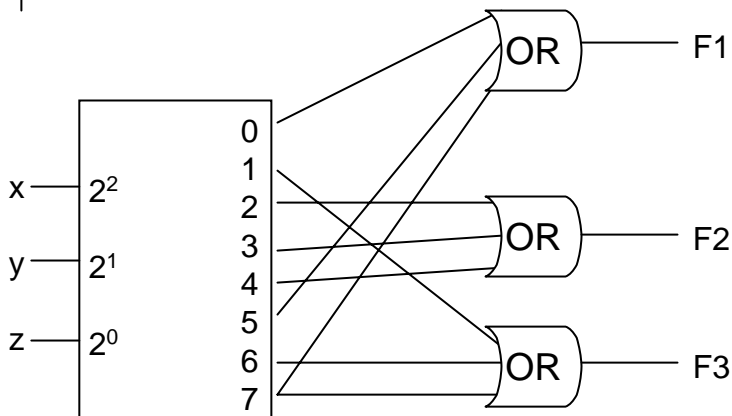
xyz	F1	F2	F3
000	1	0	0
001	0	0	1
010	0	1	0
011	0	1	0
100	0	1	0
101	1	0	0
110	0	0	1
111	1	0	1

From Truth Table:

F1 = Sum of Minterms (0,5,7)

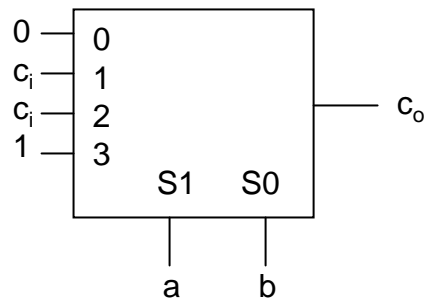
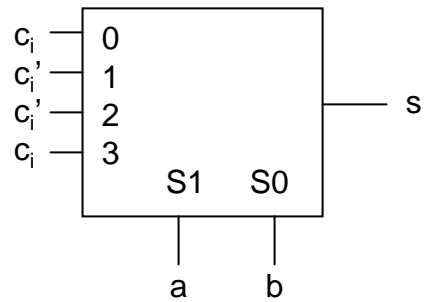
F2 = Sum of Minterms (2-4)

F3 = Sum of Minterms (1,6,7)



Call inputs a , b , and c_i . Outputs are s (sum) and c_o (carry)

a b c_i	s	c_o	
0 0 0	0	0	$s=c_i, c_o=0$
0 0 1	1	0	
0 1 0	1	0	$s=c_i', c_o=c_i$
0 1 1	0	1	
1 0 0	1	0	$s=c_i', c_o=c_i$
1 0 1	0	1	
1 1 0	0	1	$s=c_i, c_o=1$
1 1 1	1	1	



What gates are functionally complete?

NAND and NOR are both functionally complete – that is, you can implement any Boolean expression using only NAND gates or NOR gates.

XOR is not functionally complete. You can show this for a 2-input XOR gate by exhaustively testing the (reasonable) combinations of inputs:

$$A \text{ XOR } B = AB' + A'B$$

$$A \text{ XOR } B' = AB + A'B'$$

$$A \text{ XOR } A = 0$$

$$A \text{ XOR } A' = 1$$

$$A \text{ XOR } 1 = A'$$

$$A \text{ XOR } 0 = A$$

$$(A \text{ XOR } B) \text{ XOR } A = A$$

$$(A \text{ XOR } B) \text{ XOR } A' = A'$$

Now, for XOR to be functionally complete, you would have to be able to implement the complement and either AND or OR. But none of the possibilities above give AND or OR, so XOR can't be functionally complete.