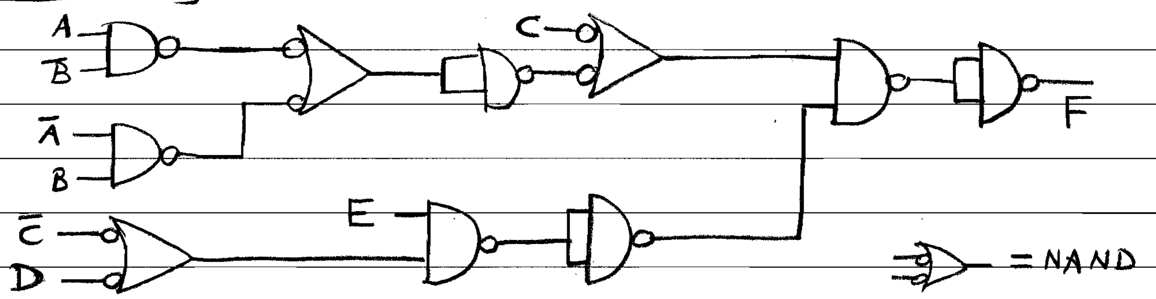
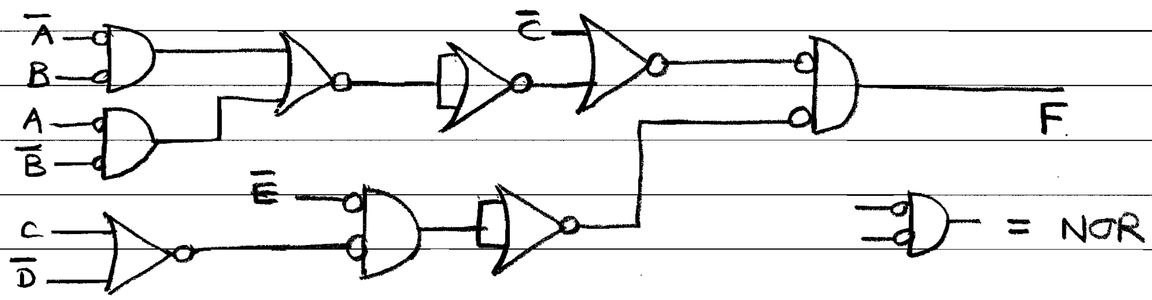
1. a: Using NANDs:



b: Using NORs



2.

a.

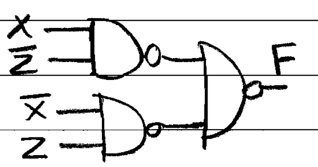
wx \ yz	00	01	11	10
00	1 (0)	0 (1)	0 (3)	1 (2)
01	0 (4)	1 (5)	1 (7)	0 (6)
11	0 (12)	1 (13)	1 (15)	0 (14)
10	1 (8)	0 (9)	0 (11)	1 (10)

$F = xz + \bar{x}\bar{z}$ — (1)

$\bar{F} = x\bar{z} + \bar{x}z$ — (2)

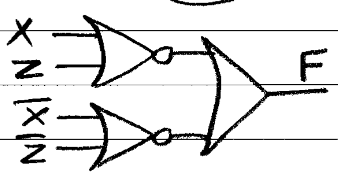
b. i. NAND-AND = $\underbrace{\text{AND-NOT}}_{\text{OR-NOT}}$-AND = $\overline{\underbrace{\text{AND-OR}}_{\text{SOP}}\text{-NOT}} = \overline{\text{SOP}}$ (Hardware invert)

$F = \overline{\text{SOP}} \leftarrow \bar{F}$ as SOP From K-map (Eqn (2))



ii. NOR-OR = OR-$\underbrace{\text{NOT-OR}}_{\text{AND-NOT}}$ = $\underbrace{\text{OR-AND}}_{\text{POS}}$-NOT = $\overline{\text{POS}}$ (Hardware invert)

$F = \overline{\text{POS}} \leftarrow \bar{F}$ as POS: F as SOP From K-map then invert with DeMorgans to get $\bar{F}$ as POS.



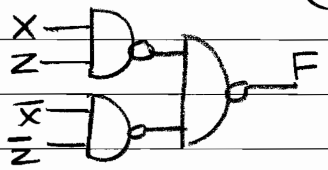
From (1) $\bar{F} = \overline{xz + \bar{x}\bar{z}} = (\bar{x}+\bar{z})\cdot(x+z)$

2 b, contd.

iii. NAND-NAND = AND-$\overbrace{\text{NOT-AND-NOT}}^{=\text{OR}}$ = $\overbrace{\text{AND-OR}}^{\text{SOP}}$

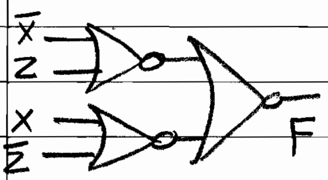
F = SOP i.e get F as a SOP from K-map.

(Eqn 1): $F = xz + \bar{x}\bar{z}$



iv. NOR-NOR = OR-$\overbrace{\text{NOT-OR-NOT}}^{\text{AND}}$ = $\overbrace{\text{OR-AND}}^{\text{POS}}$

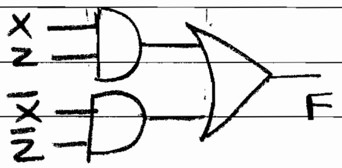
$F = \text{POS} = \overline{\text{SOP}}_{\bar{F}}$ ← DeMorgan's invert



$\bar{F}_{SOP}$ From eqn (2) $= x\bar{z} + \bar{x}z$

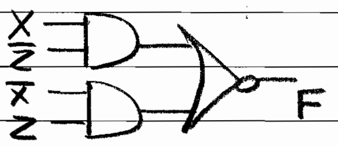
$F_{POS} = \bar{\bar{F}} = \overline{x\bar{z} + \bar{x}z} = (\bar{x} + z)\cdot(x + \bar{z})$

v. AND-OR = SOP (Same as iii)



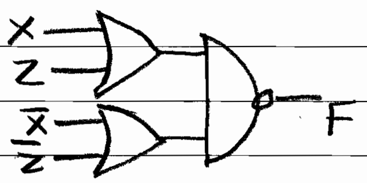
vi. AND-NOR = $\overbrace{\text{AND-OR}}^{\text{SOP}}$-NOT = $\overline{\text{SOP}}$ ← Hardware invert

(Same as i)

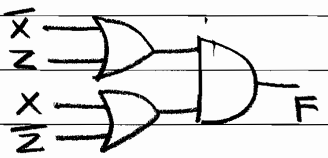


vii. OR-NAND = $\overbrace{\text{OR-AND}}^{\text{POS}}$-NOT = $\overline{\text{POS}}$ ← Hardware invert

(same as ii)



viii. OR-AND = POS (same as iv)



# of 0's in I/p.

3.

	X_3	X_2	X_1	X_0		Y_2	Y_1	Y_0
0	0	0	0	0	4	1	0	0
1	0	0	0	1	3	0	1	1
2	0	0	1	0	3	0	1	1
3	0	0	1	1	2	0	1	0
4	0	1	0	0	3	0	1	1
5	0	1	0	1	2	0	1	0
6	0	1	1	0	2	0	1	0
7	0	1	1	1	1	0	0	1
8	1	0	0	0	3	0	1	1
9	1	0	0	1	2	0	1	0
10	1	0	1	0	2	0	1	0
11	1	0	1	1	1	0	0	1
12	1	1	0	0	2	0	1	0
13	1	1	0	1	1	0	0	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	0	0	0	0

Largest # of 0's = 4 → need 3 bits for o/p

By inspection:

$Y_2 = \bar{X}_3 \bar{X}_2 \bar{X}_1 \bar{X}_0$

Y_1 map

$X_3X_2 \backslash X_1X_0$	00	01	11	10
00	0_0	1_1	1_3	1_2
01	1_4	1_5	0_7	1_6
11	1_{12}	0_{13}	0_{15}	0_{14}
10	1_8	1_9	0_{11}	1_{10}

$$Y_1 = \bar{X}_3\bar{X}_2X_1 + \bar{X}_3\bar{X}_2\bar{X}_1 + \bar{X}_3X_2\bar{X}_0 + X_3\bar{X}_1\bar{X}_0 + X_3\bar{X}_2\bar{X}_1 + X_3\bar{X}_2\bar{X}_0$$

Note: other optimal solutions exist

Y_0 can not be simplified

$$Y_0 = \Sigma(1,2,4,7,8,11,13,14)$$

can also represent this canonical form in the corresponding algebraic sum of minterms.

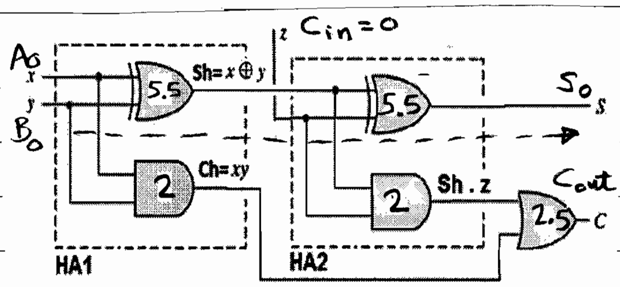
Y_0 map

$X_3X_2 \backslash X_1X_0$	00	01	11	10
00	0_0	1_1	0_3	1_2
01	1_4	0_5	1_7	0_6
11	0_{12}	1_{13}	0_{15}	1_{14}
10	1_8	0_9	1_{11}	0_{10}

4.

a. 1-bit Adder:

- Critical path (- - -)
- Delay = 5.5 + 5.5 = 11 ns.

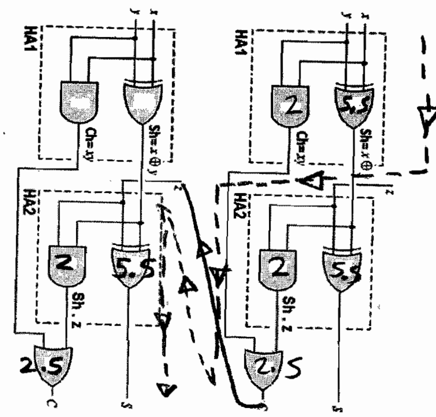


b. 2-bit ripple-carry Adder:

→ Critical path (---)

→ Delay = 5.5 + (2 + 2.5) + 5.5

= 15.5 ns



4.2 # of additions/sec

$= \frac{1\ \text{sec}}{15.5\ \text{ns}} = \frac{10^9}{15.5}$

$\simeq 64.5 \times 10^6$ additions/sec.

5.

	Inputs			Output		Overflow Occurred? (Yes/No)	Is the result correct? (Yes/No)
	A	B	Subtract/$\overline{\text{Add}}$ [O/P = (A-B) or (A+B)]	C4	S (binary)		
a	0010	0101	0	0	0111	No	Yes
b	1100	1011	1	1	0001	No	Yes
c	0111	1101	0	1	0100	No	Yes
d	1100	0110	1	1	0110	Yes	No

a: addition: 0010 + 0101 = 0111; +2 + +5 = +7 ✓

b: subtraction: 1100 − 1011 = −0100 − −0101 = (−4) − (−5) = +1

1100 + 0101 = 1 ← 0001 = +1 ✓

c: Addition: 0111 + 1101; (+7) + (−3) = +4

0111 + 1101 = 1 ← 0100 = +4 ✓

d: Subtraction: 1100 − 0110 = − 0100 − + 0110 = −4 − +6 = −10 (exceeds −8)

⇒ 1100 + 1010 = 1 ← 0110 = +6 ✗

overflow.