Computer Organization Laboratory (CS39001)

LAB TEST - I SOLUTIONS.

1) (a). $Z = S_3 XY + S_2 X\bar{Y} + S_1 \bar{X} Y + S_0 \bar{X}\bar{Y}$

(b). (a). For $S = 0110$, we get $Z = \bar{X}Y + X\bar{Y} = X \oplus Y$

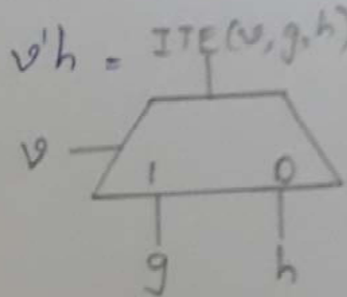
(b). For $S = 0011$, we get $Z = \bar{X}Y + \bar{X}\bar{Y} = \bar{X}$

(c). (a). For XOR function, $S = 0110$

(b). For OR function, $S = 1110$, where

$(Z = XY + X\bar{Y} + \bar{X}Y = Y + X\bar{Y} = X + Y)$.

2). Given If-Then-Else, $ITE(v, g, h) = vg + v'h$ =



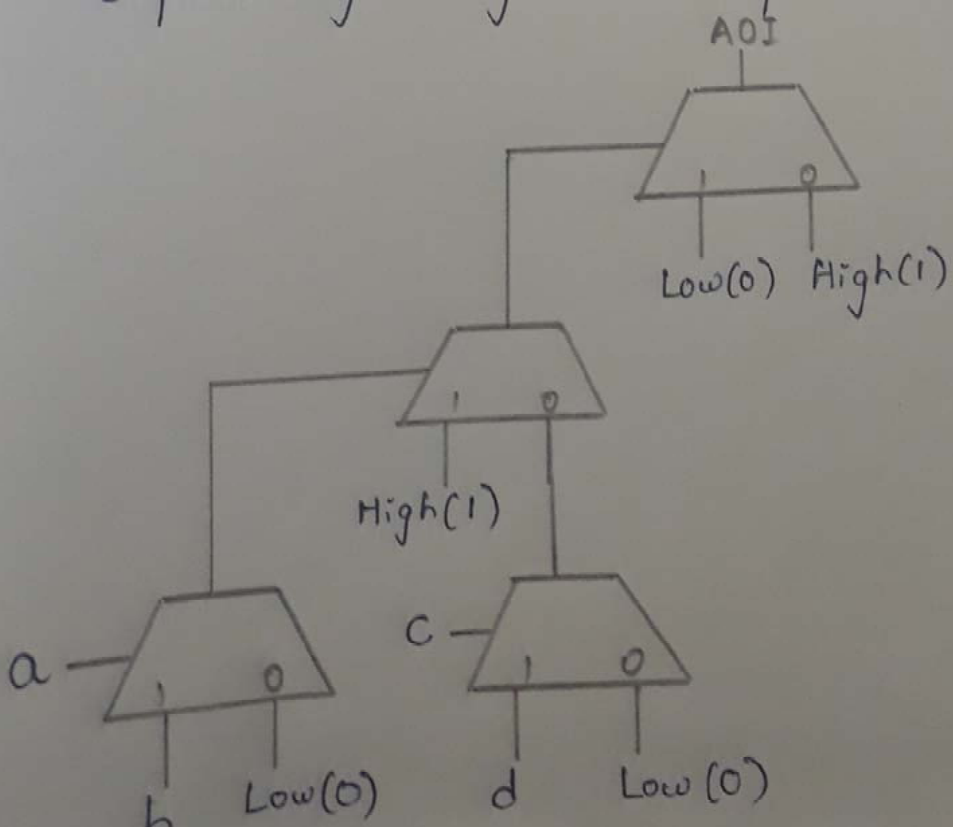
If v is TRUE then g, else h.

And-Or-Invert, $AOI(a,b,c,d) = \overline{(a \wedge b) \vee (c \wedge d)}$

which is realizable as:

$$AOI(a,b,c,d) = ITE\big(ITE(ITE(a,b,0), 1, ITE(c,d,0)), 0, 1\big)$$

Implementing using 2:1 multiplexers:



3). Given an n-bit unsigned number with a leading 0,

i.e., $X = 0\ x_{n-2} \ldots x_1\ x_0$. For 2's complement the weights are: $-2^{n-1}\ 2^{n-2} \ldots 2^1\ 2^0$

$\therefore$ The numerical value is $\sum_{i=0}^{n-2} x_i 2^i$.

Since, $\sum_{i=0}^{n-1} x_i 2^i = x_{n-1} 2^{n-1} + \sum_{i=0}^{n-2} x_i 2^i$

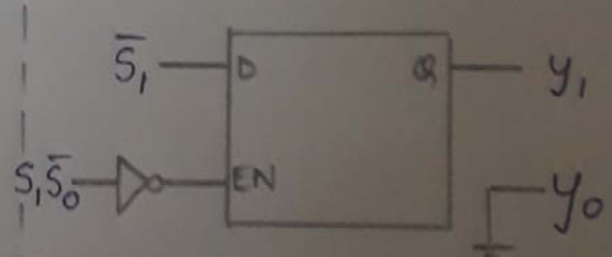
In this case, The 2's complement of binary number X is the number itself.

4). YES, Booth's multiplication algorithm is faster in cases where the binary number has successive 1's and 0's (since after Booth's encoding these strings require no operations.

No, Booth's multiplication algorithms does not always offer a spped advantage with respect to shift-and-add multiplication schemes.
Booth's Algorithm only offers a speed advantage in situtations where the available hardware does not allow add/subtract combind with shifting to take place together in the same clock cycle. In situatons where such hardware is availble, both shift-and-add schemes and Booth's algorithm take "n" clock cycles to perform n-bit * n-bit multiplication.

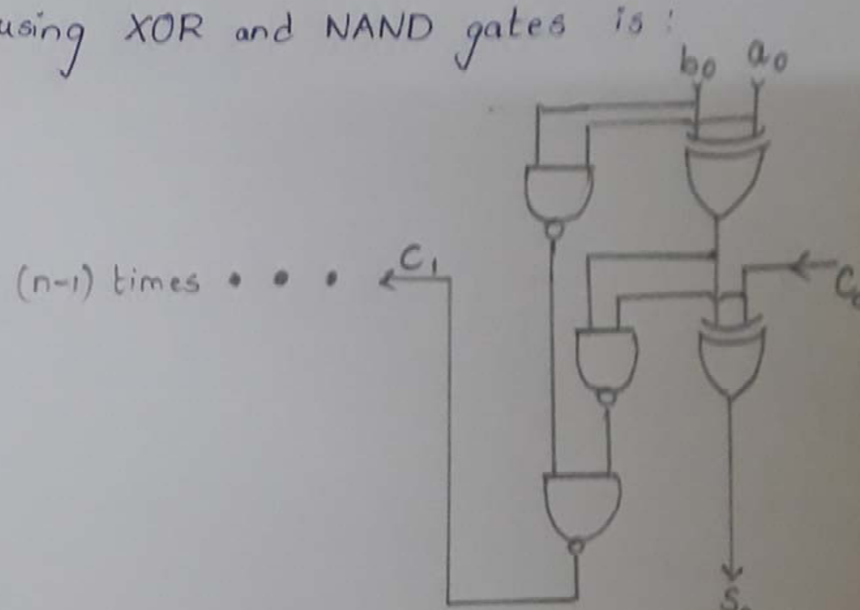
5). Note that the CASE statement does not have a default condition defined. Hence, a level-sensitive TRANSPARENT LATCH will be inferred by the synthesis tool for undefined cases. When these undefined cases are given as input, then the output will remain unchanged (i.e., previous output). The Logic diagram is:

INPUTS curr_state S_1	S_0	OUTPUTS flag y_1	y_0
0	0	1	0
0	1	1	0
undefined 1	0	NO CHANGE (Previous)	
1	1	0	0



Level-sensitive Transparent Latch:
When EN is active, Q = D
otherwise, Q = previous Q

6). The implementation using XOR and NAND gates is :



The critical path (worst-case) delay of an n-bit RCA using above implementation is : $= n \times 2t_{NAND} + t_{XOR}$

---

7). a). $c_{i+1} = g_i + c_i p_i$

since, $c_{i+1} = x_i y_i + c_i (x_i \oplus y_i)$.

b). $\overline{c_{i+1}} = k_i + \bar{c}_i p_i$

$\bar{c}_i$ \ $p_i k_i$	00	01	11	10
0		1	X	
1		1	X	1

for $\bar{c}_{i+1}$

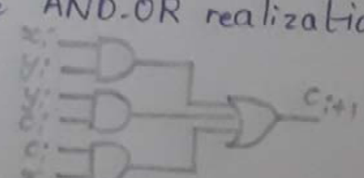
c). $c_1 = g_0 + c_0 p_0$

$c_2 = g_1 + c_1 p_1 = g_1 + (g_0 + c_0 p_0) p_1 = g_1 + p_1 g_0 + p_1 p_0 c_0$

$c_3 = g_2 + c_2 p_2 = g_2 + p_2 g_1 + p_2 p_1 g_0 + p_2 p_1 p_0 c_0$

$$\therefore \quad c_{i+1} = g_i + \sum_{k=0}^{i-1} g_i \left(\prod_{j=i}^{k+1} p_j \right) + \left(\prod_{l=0}^{i} p_l \right) c_0 .$$

d). The AND-OR realization of $c_{i+1} = x_i y_i + y_i c_i + c_i x_i$



a) One OR gate with 3 inputs

b). 3 AND gates in total.

e). The worst-case delay for 4-bit CLA is,

$$= \underbrace{t_{AND,n \text{ or } OR,n}}_{1^{st} \text{ level for } p_i, g_i} + \underbrace{t_{AND,n}}_{2^{nd} \text{ level}} + \underbrace{t_{OR,n}}_{3^{rd} \text{ level}} + \underbrace{t_{AND,n} + t_{OR,n}}_{\text{XOR for sum calculation}} \Rightarrow \begin{cases} 2t_{AND,n} + 3t_{OR,n} \\ 3t_{AND,n} + 2t_{OR,n} \end{cases}$$