Name: KEY **Id#**

COE 202, Term 151
Digital Logic Design

Quiz# 4

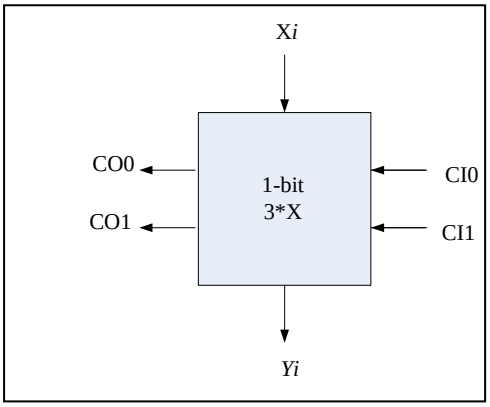
Date: Tuesday, Nov. 3

Q1. It is required to design a Tripler circuit. The circuit receives an n-bit number X and computes the result Y=3*X.

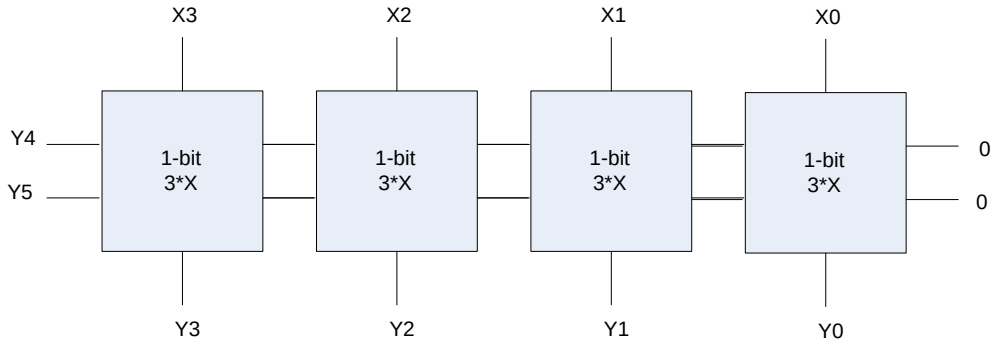
a. If the input is an *n*-bit unsigned number, what is the size of the output "*y*" in bits?

n+2

b. The circuit can be constructed using *n identical copies* of the basic 1-bit cell shown to the right. The cell processes one input bit (X*i*) and produces one output bit (Y*i*) and two output carry bits (CO0 and CO1). To allow for cascading *n* such cells to implement an *n*-bit Tripler, the basic cell also accepts two input carry bits (CI0 and CI1). When the output carry equals 1 then CO1 CO0 = 01, while when it equals 2 then CO1 CO0 = 10.



The Figure below shows how a 4-bit Tripler circuit is implemented using 4 copies of the basic 1-bit cell.

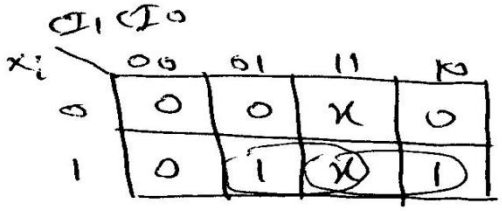


Derive the truth table for the basic one-bit cell.
(Hint: *As the initial input carries = 00, the maximum carry from one cell to the next is 2*)

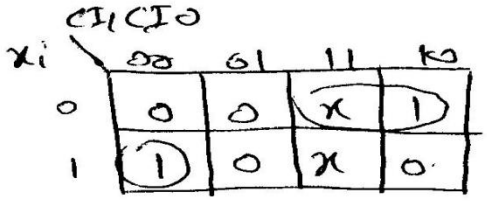
Truth Table:

x_i	CI_1	CI_0	CO_1	CO_0	y_i
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	x	x	x
1	0	0	0	1	1
1	0	1	1	0	0
1	1	0	1	0	1
1	1	1	x	x	x

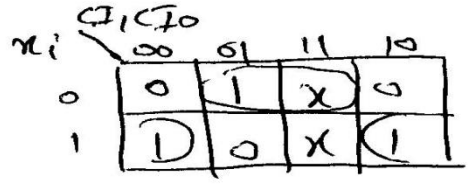
c. Derive a minimized sum-of-product expressions for the outputs of the basic one-bit cell.



$$CO_1 = x_i CI_1 + x_i CI_0$$



$$CO_0 = \overline{x_i} CI_1 + x_i \overline{CI_1}\,\overline{CI_0}$$



$$y_i = \overline{x_i} CI_0 + x_i \overline{CI_0}$$

Q2.

a. Fill in all blank cells in the two tables below. <u>All binary representations use 7 bits</u>

Binary	Equivalent decimal value with the binary interpreted as:			
	Unsigned number	Signed-magnitude number	Signed-1's complement number	Signed-2's complement number
1011010	90	-26	-37	-38

Decimal	Binary representation in:		
	Signed-magnitude notation	Signed-1's complement notation	Signed-2's complement notation
- 59	1111011	1000100	1000101

b. Using 2's-complement signed arithmetic in **5 bits**, perform the following operations in binary. Show all your work. Verify that you get the expected decimal results.

<u>Check for overflow and mark clearly any occurrences of it</u>.

(i)

11010 + 11001 → 10011

$-6 + -7 = -13$

$= -01101 = -13$ ✓

No overflow

(ii)

00101 - 10100

$+5 - -12 = +17 > +15$ → overflow expected

00101 + 01100 → 10001

$c_n \oplus c_{n-1} = 1$

∴ overflow

(iii)

(+5) + (-9) = -4

00101 + 10111 → 01100

$= -00100 = -4$ ✓

No overflow

(iv)

(-6) - (+8) = -14

11010 - 01000

11010 + 11000 → 10010

$= -01110 = -14$ ✓

No overflow

c. When doing signed 2's complement arithmetic in <u>**6 bits**</u>, the <u>*smallest*</u> binary number **that will cause overflow** when <u>*subtracted*</u> from $101000)_2$ is <u>$001001)_2$</u>

$= -011000$

$= -24$

$\therefore -24 - X = -33$

$X = -24 + 33 = +9$

$= 001001$