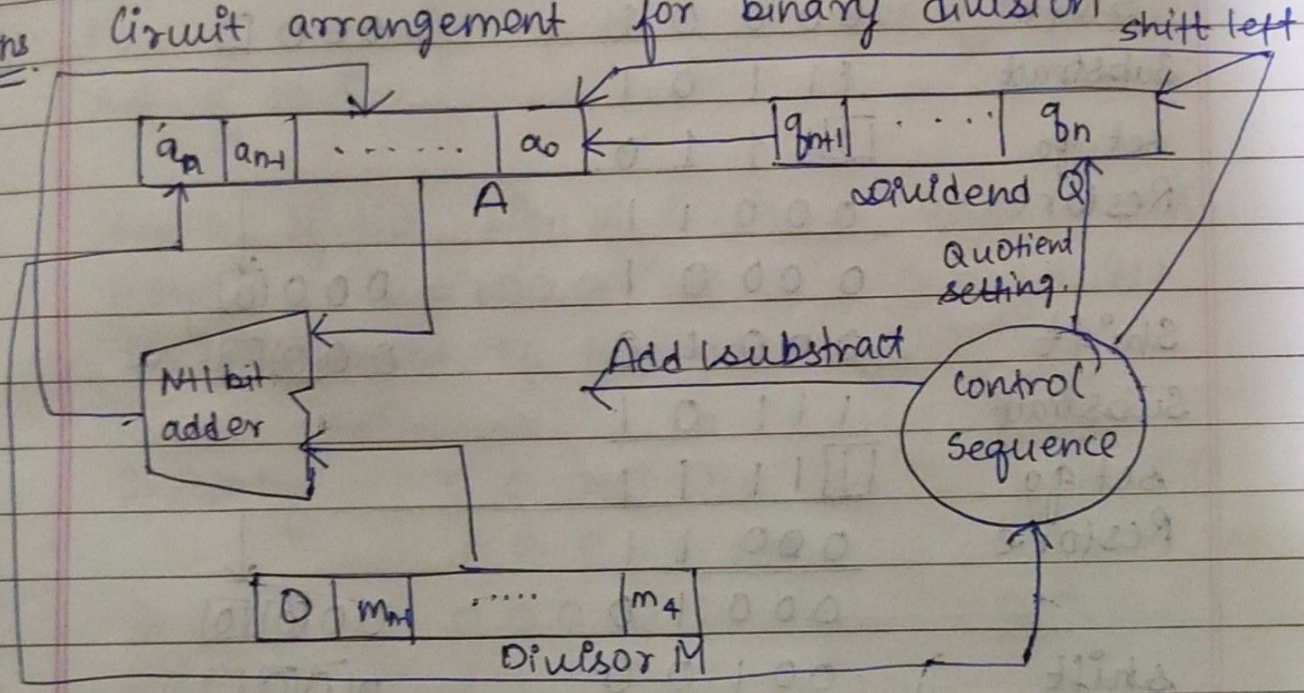
Unit4 (Important Question Answers) (for IA only

Q. Divide the numbers 8/3 or (1000/11) using restoring Division algorithm or non restoring Division algorithm.

Ans Circuit arrangement for binary division



Algorithm for Restoring Division

1. Shift A and Q left one binary position
2. Subtract M from A, and place the answer back in A
3. If the sign of A is 1, set q_0 to 0 and add M back to A (restore A); otherwise, set q_0 to 1.
4. Repeate these steps n times

Now: 8 = 1000 (Q)

3 = 11 (M)

2's complement of M = 11101

	(A)	Q
Initially	00000	1000
	00011 (M)	
Shift	00001	000□
Substract	11101	
set q0	11110	
Restore	00011	
	00001	0000
Shift	00010	000□
Substract	11101	
set q0	11111	
Restore	00011	
	00010	0000
shift	00100	000□
Substract	11101	
set q0	00001	
~~Restore~~	~~00011~~	
	~~0~~	0001
Shift	00010	001□
Substrat	11101	
setq0	11111	
Restor	00011	
	00010	0010
	Remainder	Quotient

Remainder = 00010 (2)

Quotient = 0010 (2)

Algorithm for non-restoring Division.

Step 1: (Repeat n times)

- If the sign of A is 0, shift A and Q left one bit position and subtract M from A; otherwise, shift A and Q left and Add M to A.
- Now, if the sign of A is 0, set q_0 to 1; otherwise, set q_0 to 0.

Step 2: If the sign of A is 1, add M to A.

Now, 8 = 1000 (Q) 3 = 11 (M)

2's complement of M = 11101

		A	Q
	Initially	0 0 0 0 0	1 0 0 0
		0 0 0 1 1 (M.	
1st cycle	Shift	0 0 0 0 1	0 0 0 [ ]
	Subtract	1 1 1 0 1	
	set q_0	(1) 1 1 1 0	0 0 0 [0]
2nd cycle	Shift	1 1 1 0 0	0 0 [0] [ ]
	Add	0 0 0 1 1	
	set q_0.	(1) 1 1 1 1	0 0 [0] [0]
3rd cycle	Shift	1 1 1 1 0	0 [0] [0] [ ]
	Add	0 0 0 1 1	
	set q_0.	(0) 0 0 0 1	0 [0] [0] [1]
4th cycle.	Shift	0 0 0 1 0	[0] [0] [1] [ ]
	Substract	1 1 1 0 1	
	set q_0.	(1) 1 1 1 1	[0] [0] [1] [0]

Quotient.

Add. 1 1 1 1 1 / 0 0 0 1 1 — Restore.

0 0 0 1 0 (Remainder)

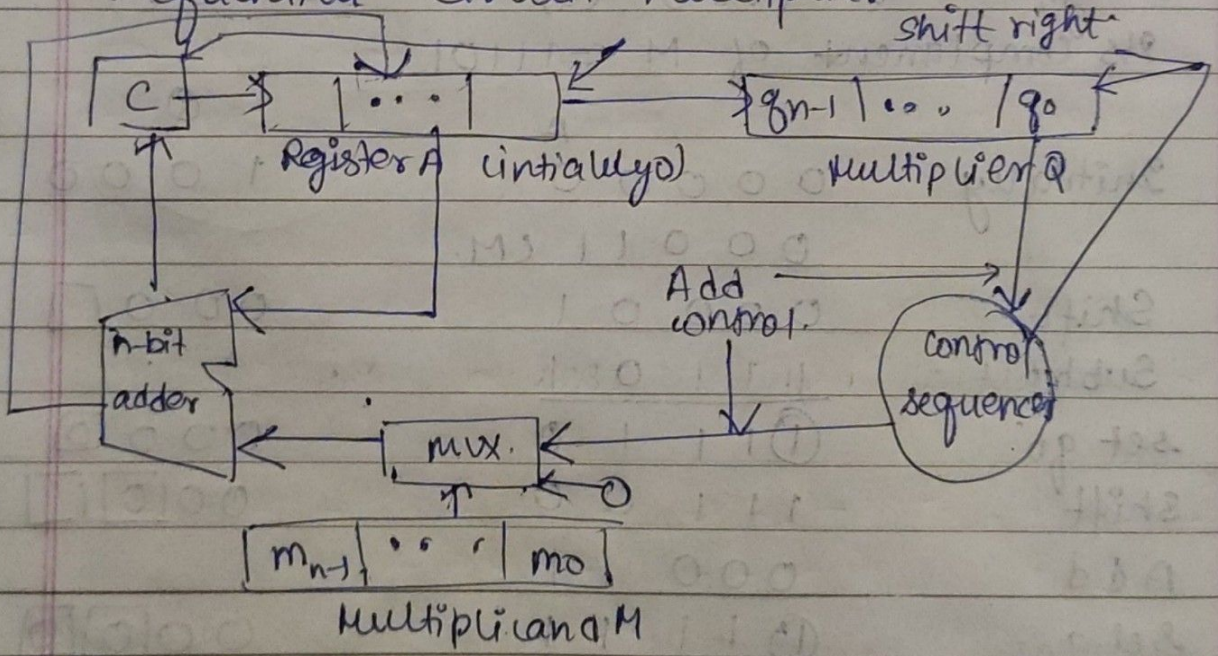
Q. Explain Sequential circuit with example.

or

Explain Binary ~~expla~~ multiplier with example.

Ans:- Sequential Multiplier is an old method to multiply two binary numbers. The multiplication between two operands a and b can be considered as add the operand a total b times

Sequential Circuit Multipliers



Sequential Multiplication Algorithm.

1. If (LSB of Q register == 1)
2. A = A+M (carry-out goes to 'C' register)
3. Treat the C, A and Q registers as one continous register and shift that register's contents right by one bit position.
4. Repeat the following steps "n" times.
   If (LSB of Q regisster == 1)
   Shift 2 and do 3.

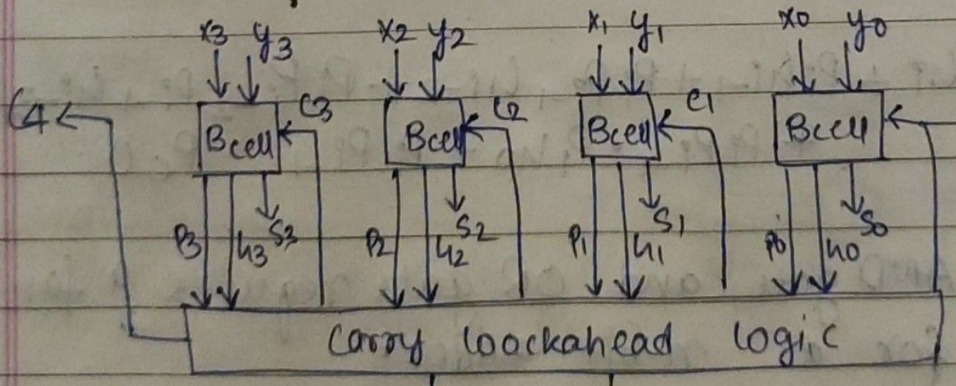
Eg:

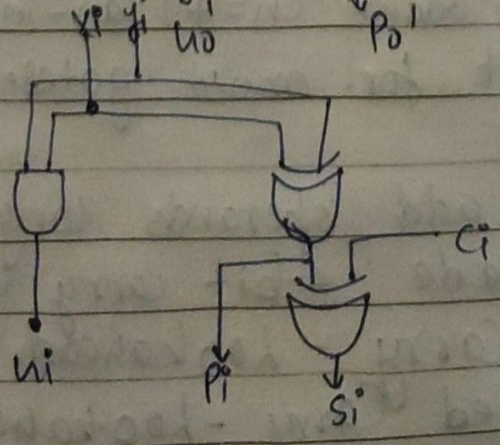
	M		Q	
	1 1 0 1		M = 13	
	0 0 0 0 A		Q = 11	
C			1 0 1 1 Q	
0	1 1 0 1		1 0 1 1	Add → First shift cycle
0	0 1 1 0		1 1 0 1	
	1 1 0 1			Add → Second shift cycle
1	0 0 1 1		1 1 0 1	
0	1 0 0 1		1 1 1 0	
				No Add → Third shift cycle
0	1 0 0 1		1 1 1 0	
0	0 1 0 0		1 1 1 1	
	1 1 0 1			Add → Fourth shift cycle
1	0 0 0 1		1 1 1 1	
0	1 0 0 0		1 1 1 1	

Product.

Q How do you design fast adders.

Ans





B cell for a single stage.

From the above

$S_i = x_i \oplus y_i \oplus c_i$

$C_{i+1} = x_i y_i + x_i c_i + y_i c_i$

The above equation can be written as

$C_{i+1} = x_i y_i + (x_i + y_i) c_i$

$C_{i+1} = G_i + P_i c_i$

where

$G_i = x_i y_i \qquad P_i = x_i + y_i$

$G_i \rightarrow$ Generate function $\qquad P_i \rightarrow$ Propagate function

→ G_i and P_i are computed only from x_i and y_i and not from c_i, thus they can be computed in one gate delay after x and y are applied to the inputs of ~~the~~ an n-bit adder.

→ ~~Perfm~~ Performing an n-bit addition in 4 gate delays independent of n is good only theoretically because of fan-in constraints

$$C_{i+1} = G_i + P_i G_{i-1} + P_i P_{i-1} G_{i-2} + P_i P_{i-2} P_{i-1} G_{i-3} + \cdots + P_i P_{i-1} \cdots P_1 G_0 + P_i P_{i-1} \cdots P_0 C_0$$

→ Last AND gate and OR gate require a fan-in of (n+1) for a n-bit adder.

- For a 4-bit adder (n=4) fan-in of 5 is required.
- Practical limit for most gates.

→ In order to add operands longer than 4 bits, we can cascade 4-bit carry lookahead adders. Cascade of Carry-Lookahead adders is called Blocked Carry-Lookahead adder.

2. Explain n-bit ripple carry adder.

⇒ → At the i^{th} stage:

Input: x_i, y_i and c_i (carry-in) are the inputs.

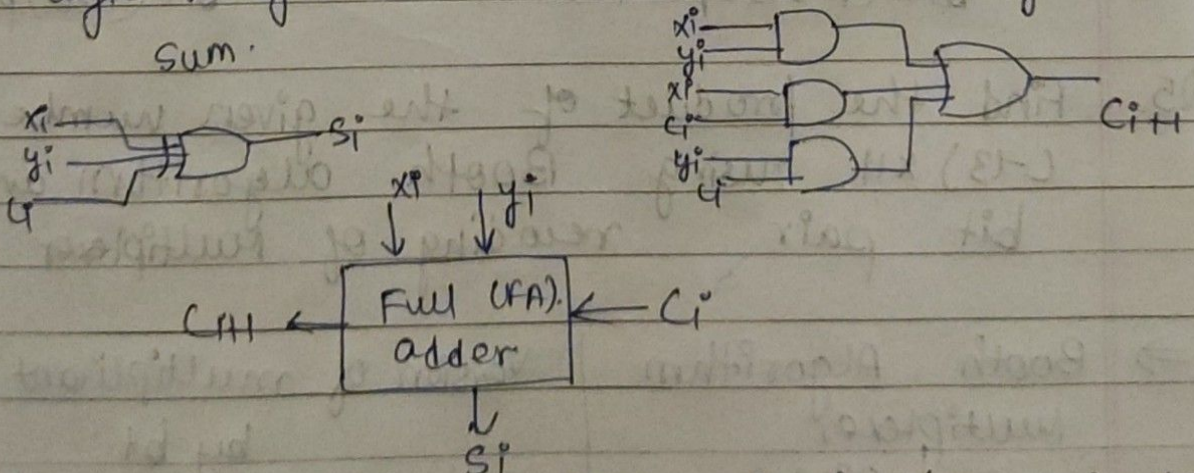
Output: s is the sum c_{i+1} (carry out to $(i+1)^{st}$ stage)

x_i	y_i	Carry-i(c_i)	Sum(s_i)	Carry-out (c_{i+1})
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

$$S_i = \bar{x_i} \bar{y_i} c_i + \bar{x_i} y_i \bar{c_i} + x_i \bar{y_i} \bar{c_i} + x_i y_i c_i = x_i \oplus y_i \oplus c_i$$

$$c_{i+1} = x_i y_i + x_i c_i + y_i c_i$$

Logic Diagram:-

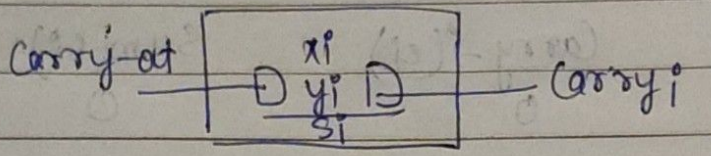


FA: Symbol for the complete circuit for a single stage of addition.

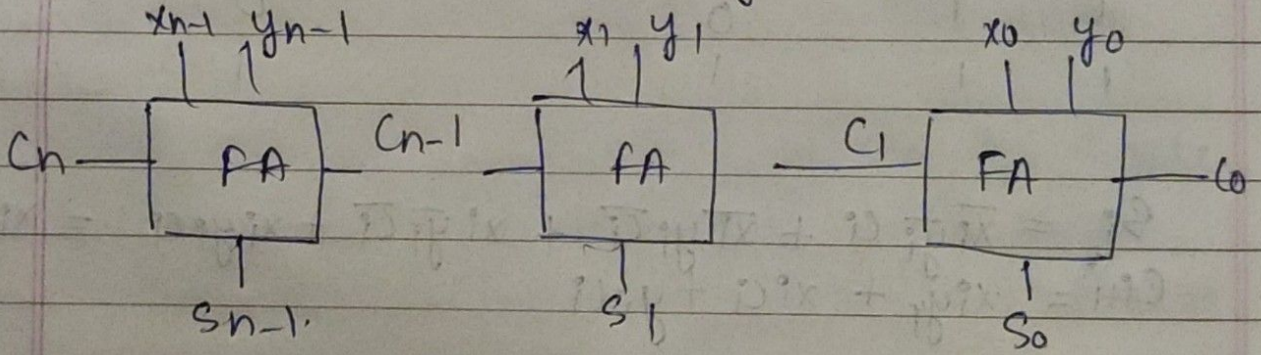
Teacher's Signature .....

Example:

$$\begin{array}{lllll} X & 7 & 0\ 1 & 1\ 1 & \\ +\ Y & = +\ 6 & = \ 0\ 0\ 1\ 1 & 1\ 0 & 0 \\ \hline Z & 13 & 1\ 1 & 0\ 1 & \end{array}$$



- Cascade n bit full adder (FA) blocks to form a n-bit adder.
- Carries propagate or ripple through this cascade, n-bit ripple carry added.



Most significant bit (MSB) position

least significant bit (LSB) position.

Q5 Find the product of the given numbers (-13) x 11 using Booth algorithm and bit pair recoding of multiplexer.

⇒ Booth Algorithm multiplexer

Biti	Biti-1	Version of multiplicant selected by bt
0	0	0 x M
0	1	+1 x M
1	0	-1 x M.
1	1	0 x M