Unit UNIT-2

Date - 24/10/19

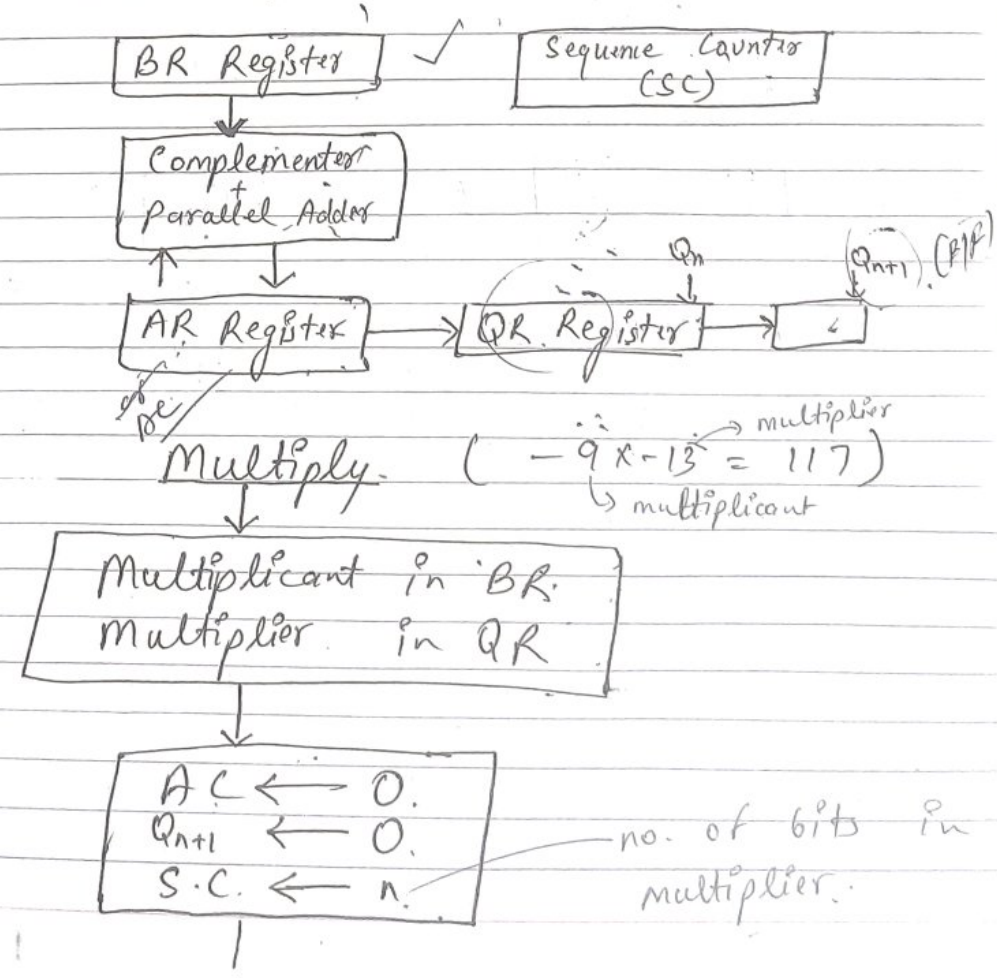
COMPUTER ARITHMETIC -

* MULTIPLICATION HARDWARE ALGO.

(i) BOOTH ALGORITHM -

Booth Algorithm gives a procedure for multiplying binary integers in signed 2's complement representation.

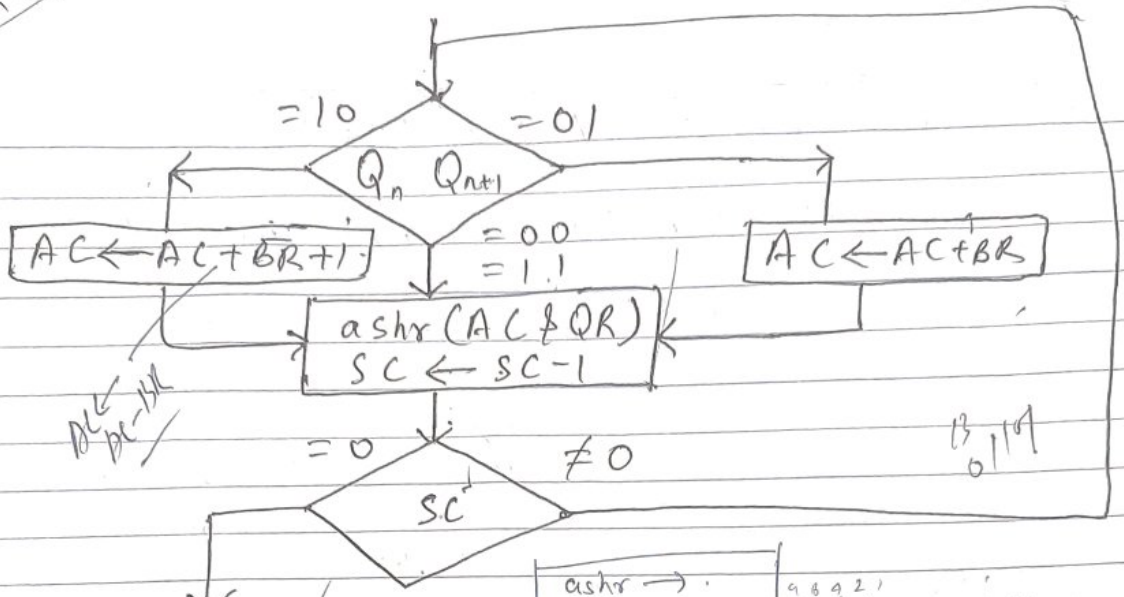
⇒ Hardware Used -



Q_n → right most bit

ashr → arithmetic shift right.

−x− = A
−x+ = S
+x− = 2's
+x+ = +



ashr →

Q_n Q_{n+1}	BR = 10111 ; BR+1 = 01001	AC	QR	Q_{n+1}	SC
		00000	1001	0	101
1 0	Add BR+1	01001			
		01001			
	ashr AC & QR	00100	1100	1	100
1 1	ashr AC & QR	00010	0110	0	011
		10111			
0 1	Add BR	11001			
	ashr AC & QR	11100	1011	0	010
0 0	ashr AC & QR	11110	0101	1	001
		01001			
1 0	Add BR+1	00111			
	ashr AC & QR	00011	1010	1	000

16 8 4 2 1

0101

Q -5 x 9 using booth Algo.

$Q_n Q_{n+1}$	BR+1 = 0101 BR+1 = 1011	AC	QR	Q_{n+1}	SC
		0000	0111	0	100
10	Add BR+1	0101			
		0101 0111			
	ashr AC&QR	0010 1011		1	011
11	ashr AC&QR	0001 0101		1	010
11	ashr AC&QR	0000 1010		1	001
		1011			
01	Add BR	1011 1010			
	ashr AC&QR	1101 1101		0	000

↓ 2's.

00100011 (-35) ✓

ARRAY MULTIPLIER -

(2- bit Multiplier) -

For j multiplier bits & K multiplicant bits we need j x K AND gates & (j-1) K-bits Adders to produce a product of (j+K) bits.

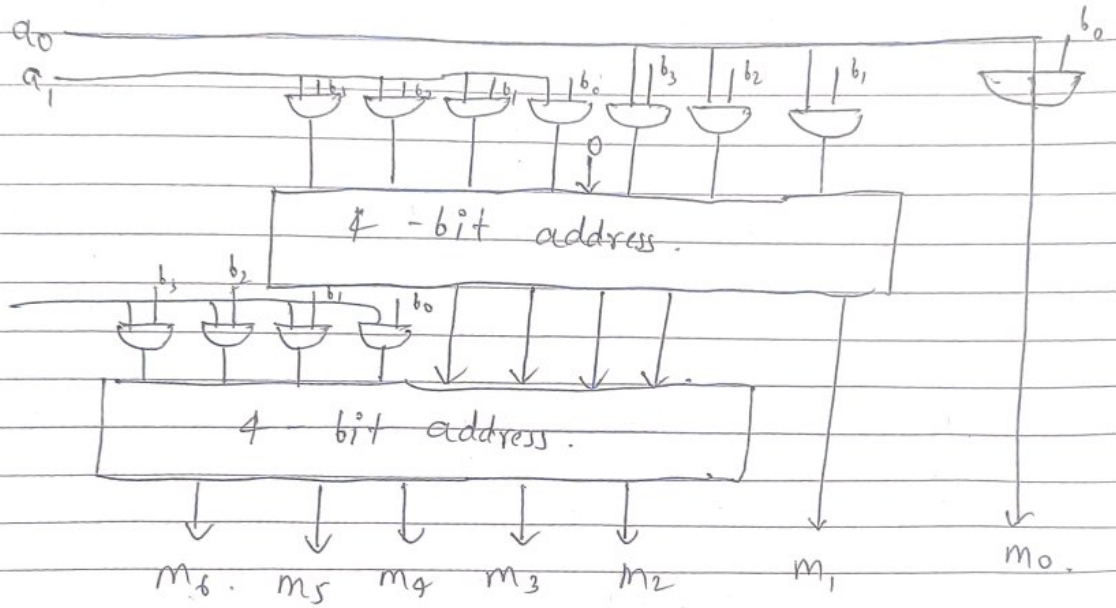
Ex:-

```
  1111   (4 bits)
x  11    (3 bits
------
  1111
 1111
1111
------
01000   (7 bits)
```

* 4 bit by 3-bit array multiplier.

$B = b_3\ b_2\ b_1\ b_0$.

$A = a_2\ a_1\ a_0$.



$b_3\ b_2\ b_1\ b_0$

$a_2\ a_1\ a_0$

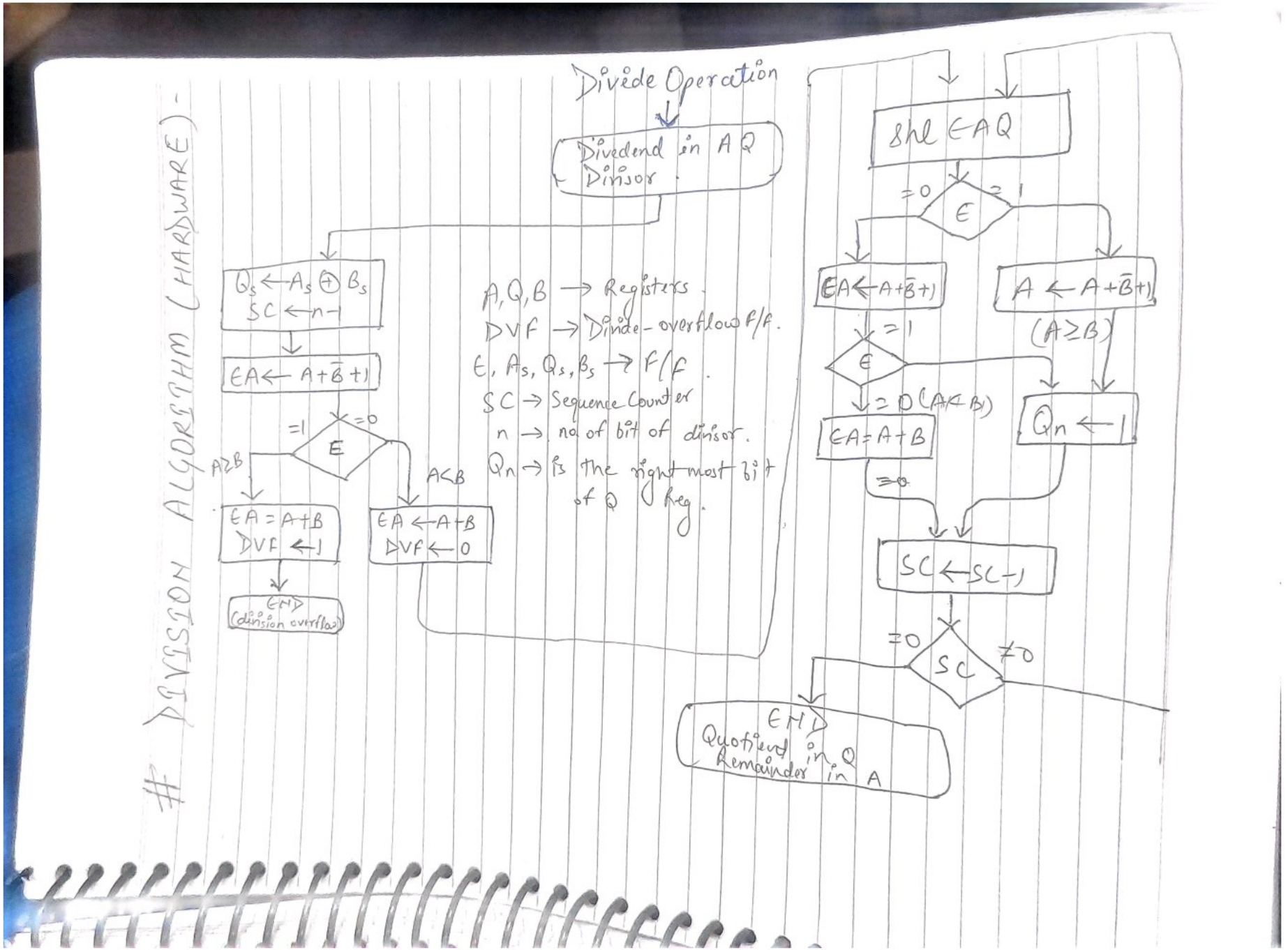
$a_0b_3\quad a_0b_2\quad a_0b_1\quad a_0b_0$

$a_1b_3\quad a_1b_2\quad a_1b_1\quad a_1b_0$

$a_2b_2\quad a_2b_1\quad a_2b_0\quad$ ~~a_2b_0~~

$m_5\quad m_4\quad m_3\quad m_2\quad m_1\quad m_0$

DIVISION ALGORITHM (HARDWARE) -



when A = 100, 101, 110 } then division overflow error come.

Q 15/4 using divisor algorithm

Q = 3, A = 3 | B = 100, $\bar{B}+1$ = 100 | AQ = 00 1111 (28) (15)
A Q

	E	A	Q	SC.
	0	001	111	011
Shl EAQ	0	011	110	
Add $\bar{B}+1$		100		
	0	111	110	
		100		
E=0, Add B	1 ←	1011	110	010.
Shl EAQ	0	111	100	
		100		
E=0, Add $\bar{B}+1$		1011	100	
E=1, Q_n ← 1	1	011	101	001
Shl EAQ	0	111	010	
		100		
=0, Add $\bar{B}+1$	0	1011	010.	
=1, Q_n ← 1	1	1011	011	000.

Remainder (3)

Quotient (3).

25/7 → H.W

A = 4 , Q = 3.

B = 111 , $\bar{B}+1$ = 001 & AQ = 011001

Q 25/7 using divisor algorithm.

=

	E	A	Q	SC
	0	011	001	011
Shl EAQ	0	110	010	
		001		
Add $\bar{B}+1$	0	111	010	
		111		
E=0, Add B	1	010	010	010
E=1, $Q_n \leftarrow 1$	1	010	011	
Shl EAQ	0	100	110	
		001		
Add $\bar{B}+1$	0	101	110	
		101		
	0	010	110	001
Shl EAQ	1	001	100	
		111		
Add $\bar{B}+1$	1	000	100	
$Q_n \leftarrow 1$	1	000	101	

Remainder (=4) | Quotient (=3)

Institute of Electrical & Electronic Engineers
(It is a society which recommend

FLOATING POINT REPRESENTATION

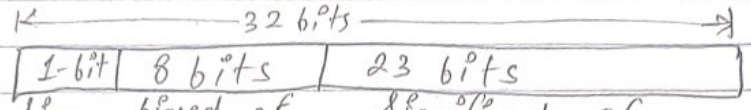
25.67×10^{5} → exponent

mantissa → Base/Radix.

25.67×10^{5}
2.567×10^{6}
256.7×10^{4}

1.1101×2^{5} → exponent

* IEEE 754 FORMAT (32-bits Format)



Bias value = $2^{K-1}-1$ where, K → no. of bits in exponent.

Bias value = $2^{8-1}-1 = 127$. (for 32-bits format)

Mantissa = normalized.
(Significant) = (1.____)

ex- (i) -1.5 & (ii) 384 represent these no.'s into 32-bits IEEE format.

Solution - (i) $-1.5 \Rightarrow (-1.1)_2$ → normalized. $.5 \times 2 = 1$
↓ mantissa. exp = 0+127 = 127.

IEEE format

| 1 | 01111111 | 10000000000000000000000 |

(ii) $384 \Rightarrow (110000000)_2 \Rightarrow$ normalize (1.1×2^{8})

exp = 127 + 8 = 135.

= 10000111

512 256 128 64 32 16 8 4 2 1

IEEE format | 0 | 10000111 | 00000000000000000000000 |

ex- 111.1010110×2^6

$\Rightarrow 1.1110101 10 \times 2^8$ | exp = 8 + 127 = 135, 10000111

IEEE format | 0 | 10000111 | 11010110000000000000000 |

* 64- Bits FORMAT

1-bit	11-bits	52-bits
Sign	biased of exponent	significant or mantisa.

Bias Value = $2^{K-1}-1$ where K → no. of bit of exponent

Bias Value = $2^{11-1}-1 = 2^{10}-1 = 1023$.

ex- 1110.0101×2^{-20}

$\Rightarrow 1.1100101 \times 2^{-17}$ | ~~exp = 127 - 17 = 110~~ exp = -17 + 1023 = 1006 = 01111001110.

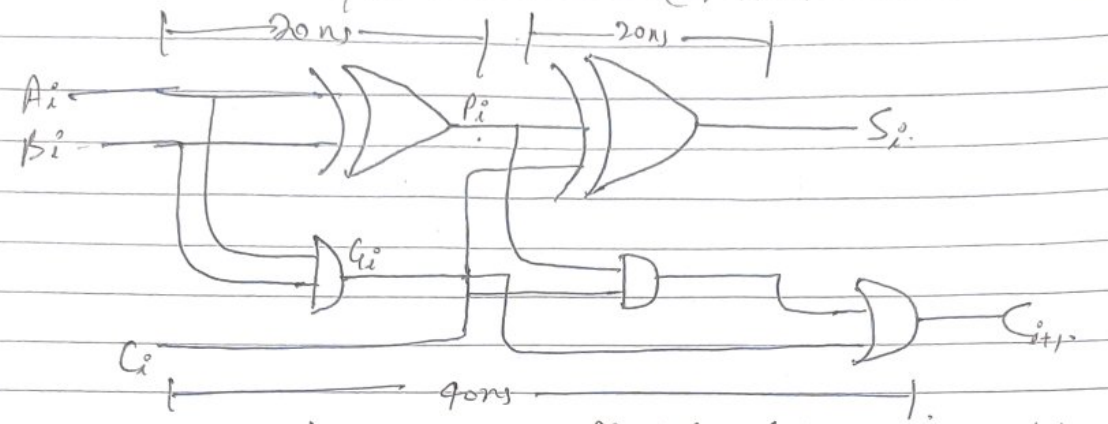
IEEE 64-bit format

0	01111001110	11001010000000000000000000000000000000000000000000

$S_i = P_i \oplus C_i$

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

PARALLEL ADDER WITH LOOK-AHEAD CARRY GENERATOR (FAST ADDER).

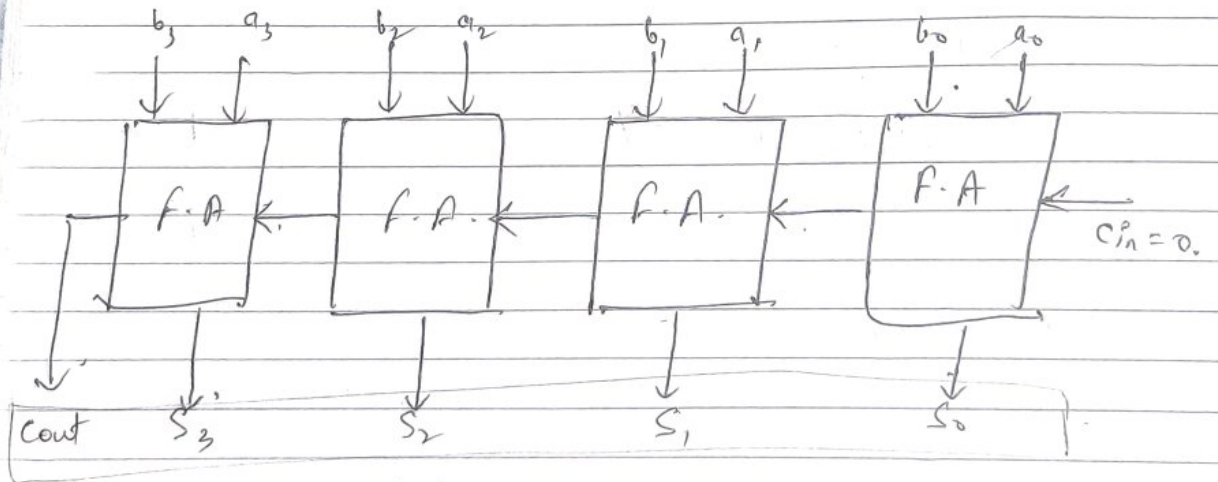


Propagation Delay :- $P_i = A_i \oplus B_i$ (Carry Propogate).
$G_i = A_i B_i$ (Carry Generator).

XOR - 20ns , OR & AND - 10ns.

FULL ADDER :-

* 4-bit Parallel Adder



Propagation delay = 4 x 40ns = 160ns.

In General - n (no. of bit of operands) x 40ns.

If $i=1$; $C_2 = G_1 + P_1 C_1$ — (1)

If $i=2$; $C_3 = G_2 + P_2 C_2 = G_2 + P_2 (G_1 + P_1 C_1)$

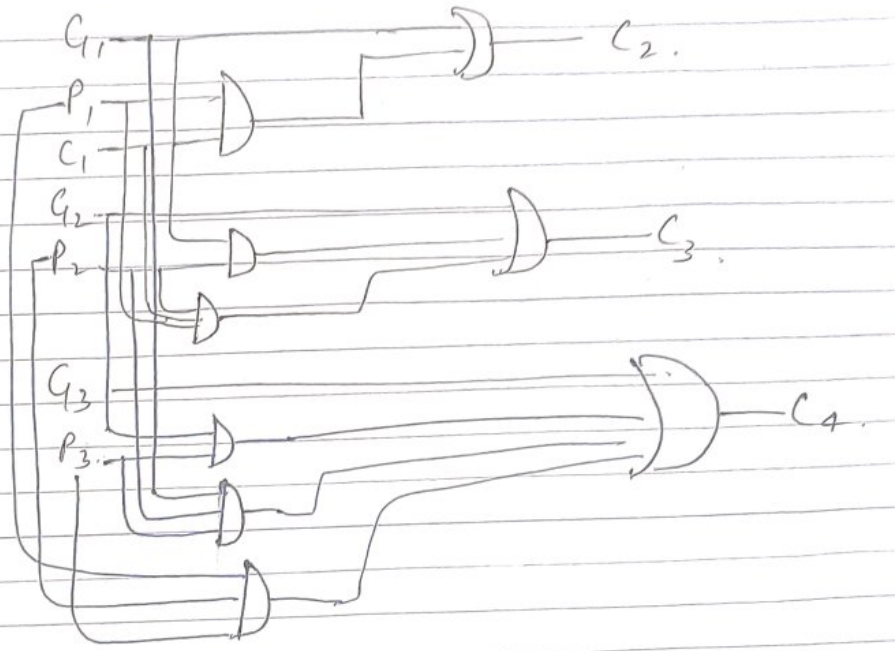
$C_3 = G_2 + G_1 P_2 + P_1 P_2 C_1$ — (2)

If $i=3$; $C_4 = G_3 + P_3 C_3$.

$C_4 = G_3 + P_3 (G_2 + G_1 P_2 + P_1 P_2 C_1)$

$C_4 = G_3 + G_2 P_3 + G_1 P_2 P_3 + P_1 P_2 P_3 C_1$ — (3)

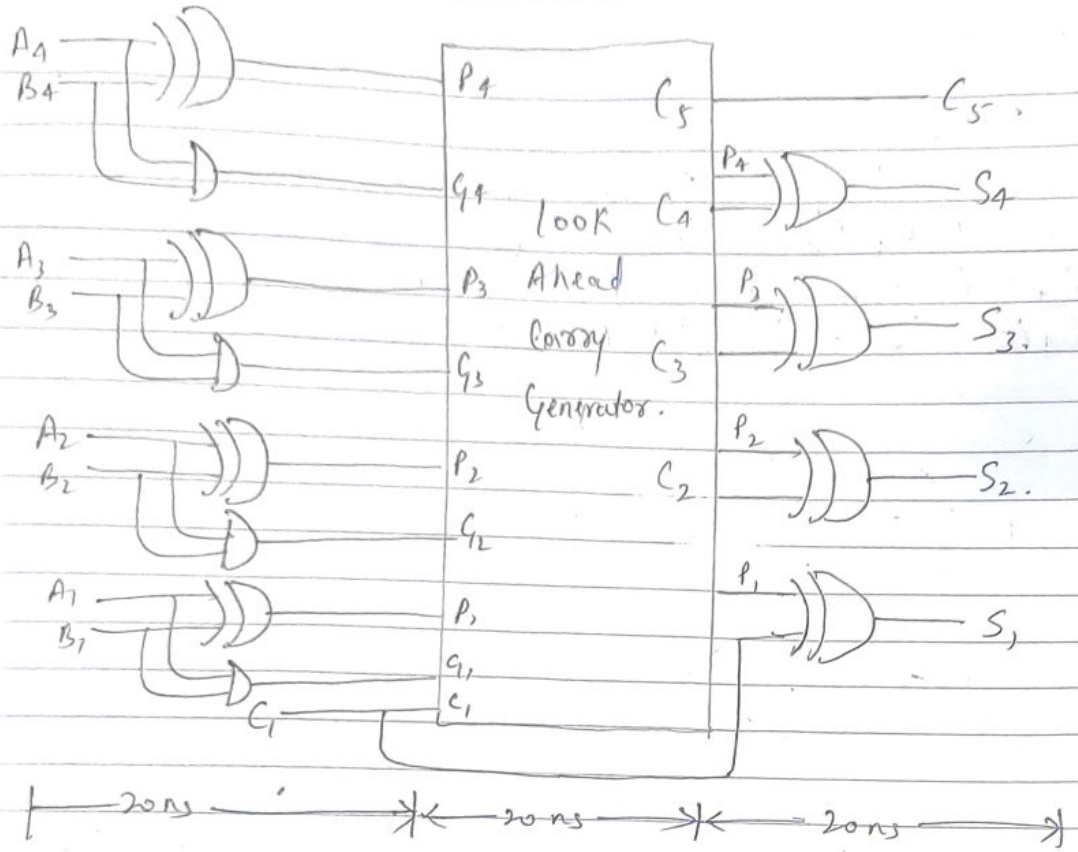
C_2 / C_3 / C_4 =>



Propagation delay of look ahead carry generator is constant & it is 20ns.

+++

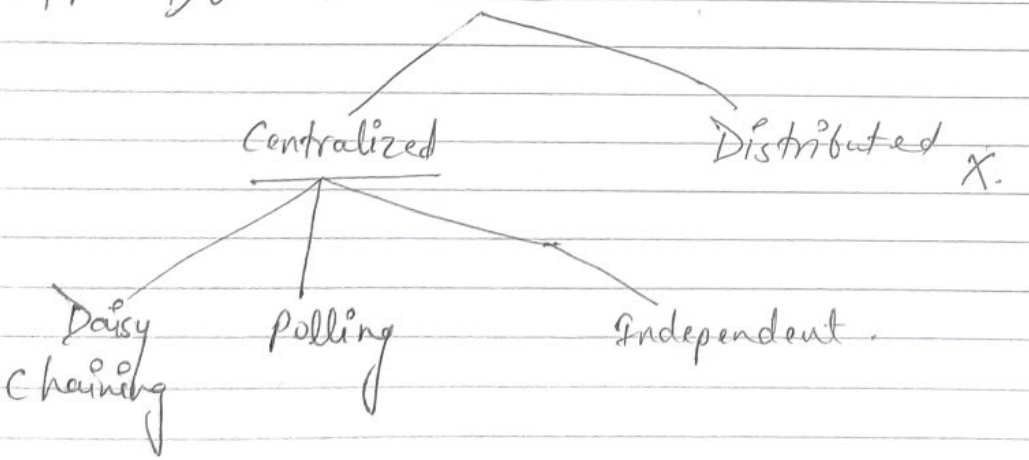
FAST ADDER



Propagation delay of Fast Adder is 60ns.
XOR = 20ns & AND or OR = 10ns.

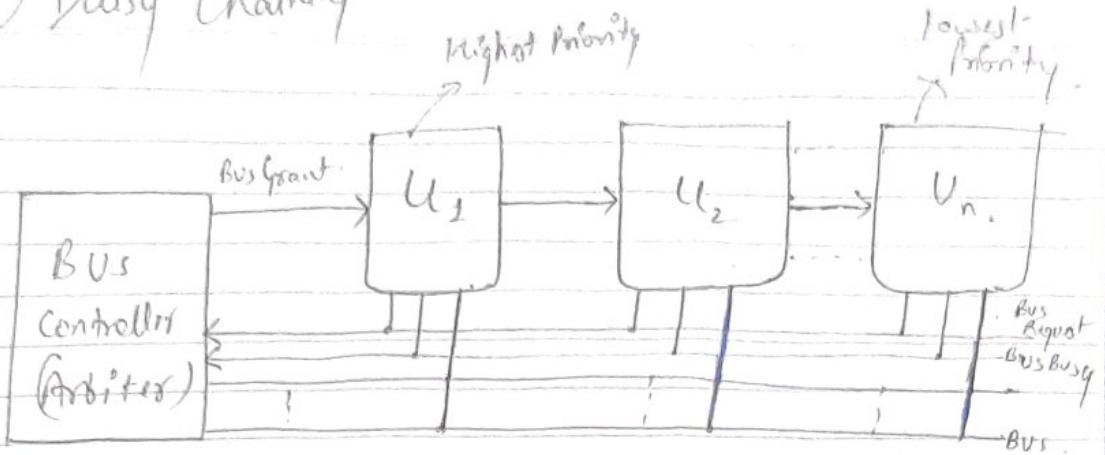
imp +++

BUS ARBITRATION:



Arbitration → mediator

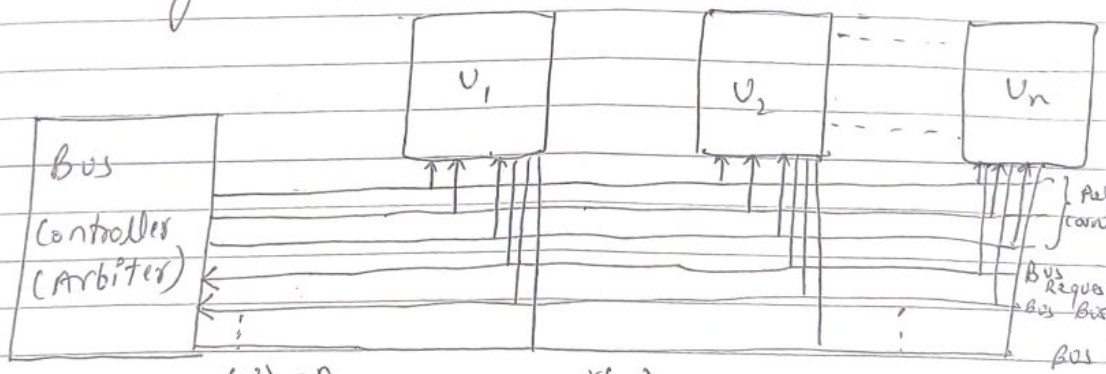
(i) Daisy Chaining



The drawback of daisy chaining is when any device is break the whole chain will break.

If n-devices then 2 control line < BG / BR

(ii) Polling



If 8 (2^3) 2^n devices then 3 or (n) poll count lines are there.

The drawback of Polling is that here the no. of wires increases so much.

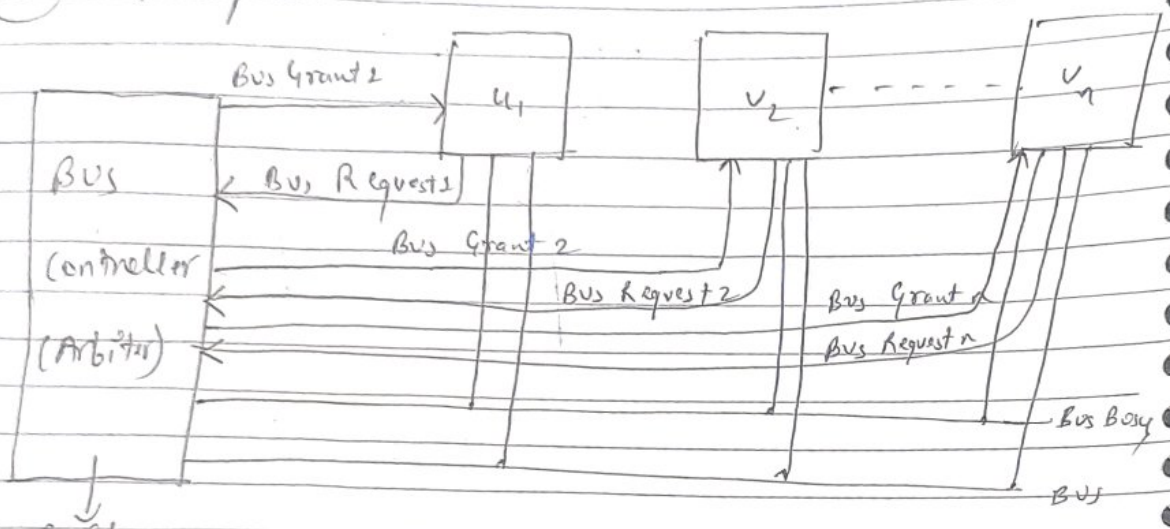
n- devices then lines = $\log_2 n$.

Both Daisy chaining & Polling are slow method.

Short Note → · CISC / RISC.

Generations of Computers. (2nd & 3rd Generation write arithmetic operation of floating pt Advantages

(iii) Independent -



priority encoder. (If more than one bus request issues by devices then Bus controller uses P.E.)

The drawback of Independent is the no. of wire increases.

· If n devices then 2n lines are required.

HARD