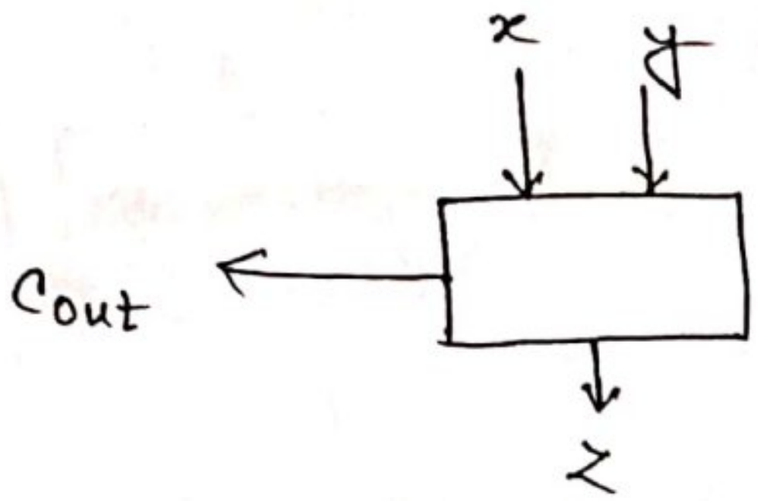
Adder Circuit : দুই রকম : Half Adder
Full Adder

```
   3        0 0 1 1        4 bit addition
  +2        0 0 1 0          4 bit Adder
  ---------------------
   5        0 1 0 1
```

> n bit adder for n bit addition

Half Adder : (1 bit Half Adder)

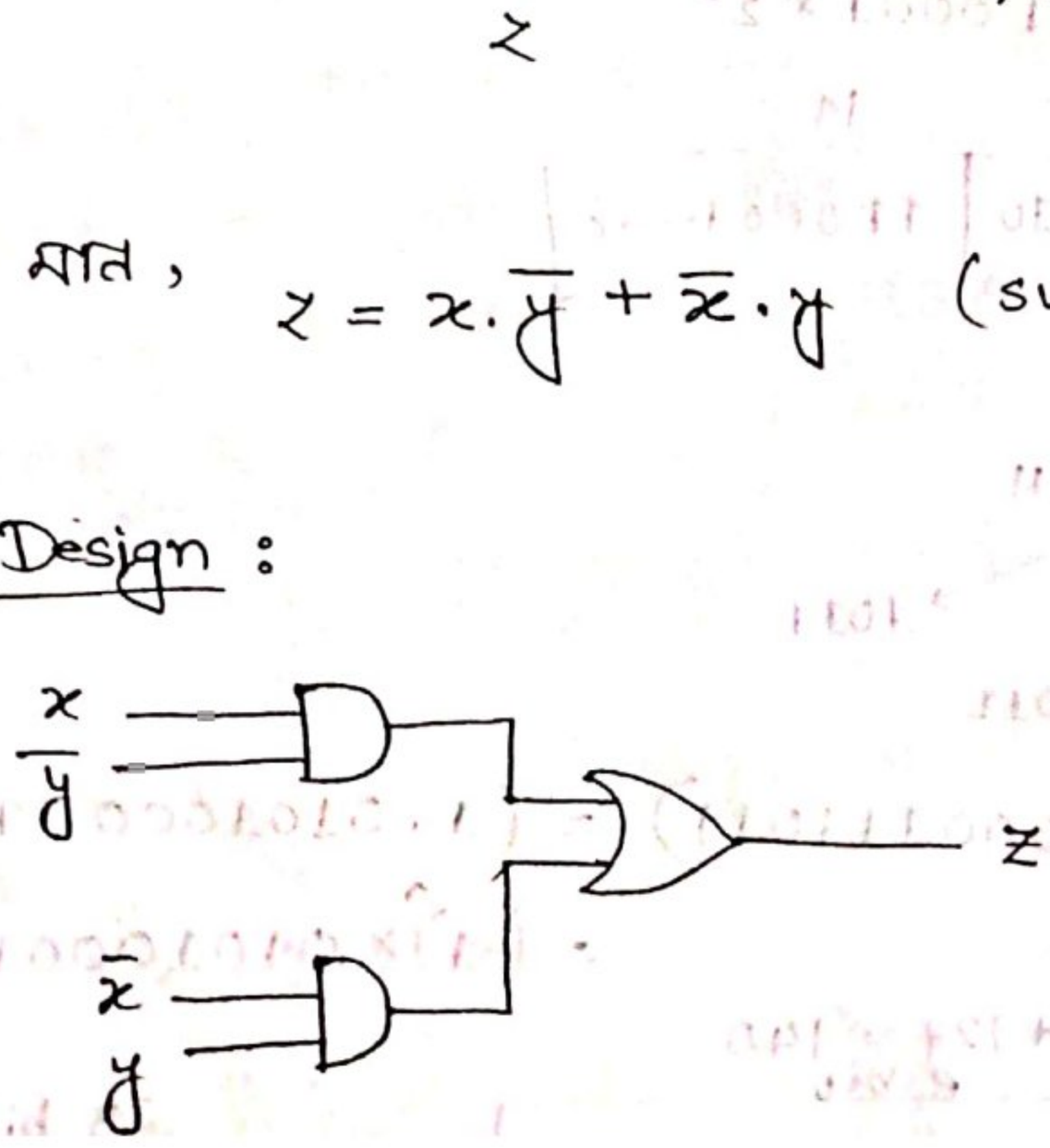


i/p → x, y
o/p → sum(z),
carry (cout)

z এর মান, $z = x.\overline{y} + \overline{x}.y$ (sum 1 পাব)

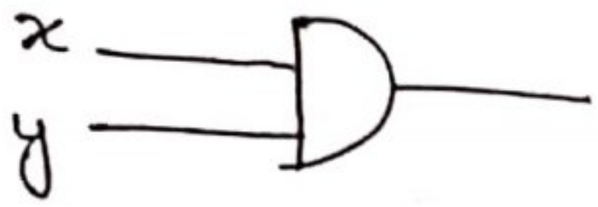
> কখন আমরা sum এ 1 পাব?

Gate Design :



carry = 1 শুধুমাত্র যখন $x = 1$, $y = 1$

$$C_{out} = x \cdot y$$

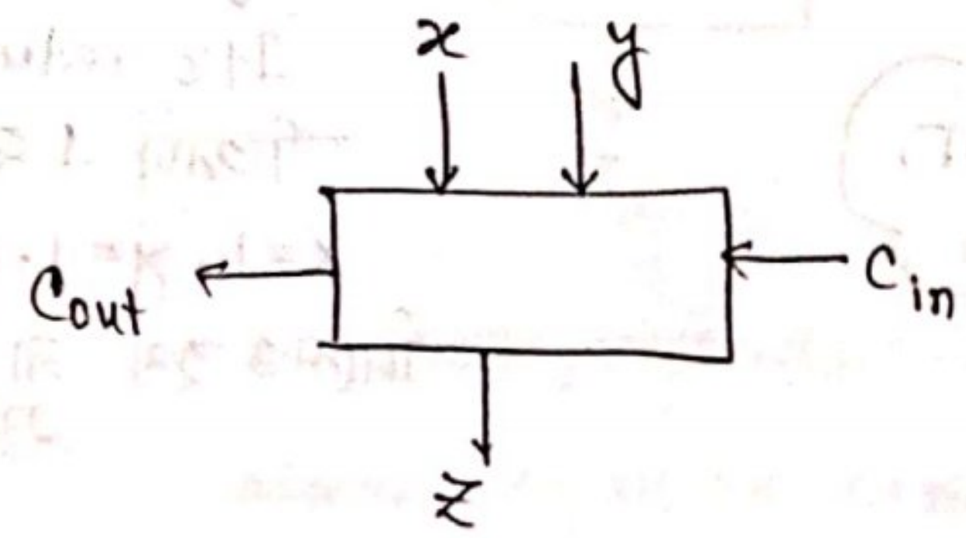


* কখন Half Adder কাজ করে?

It can add two 1 bit number independently, but it can't be used in a circuit with a series of number.

এজন্য Half adder থেকে full adder এ switch করি।

Full Adder : (1 bit Full Adder)



I/p : x, y, Cin

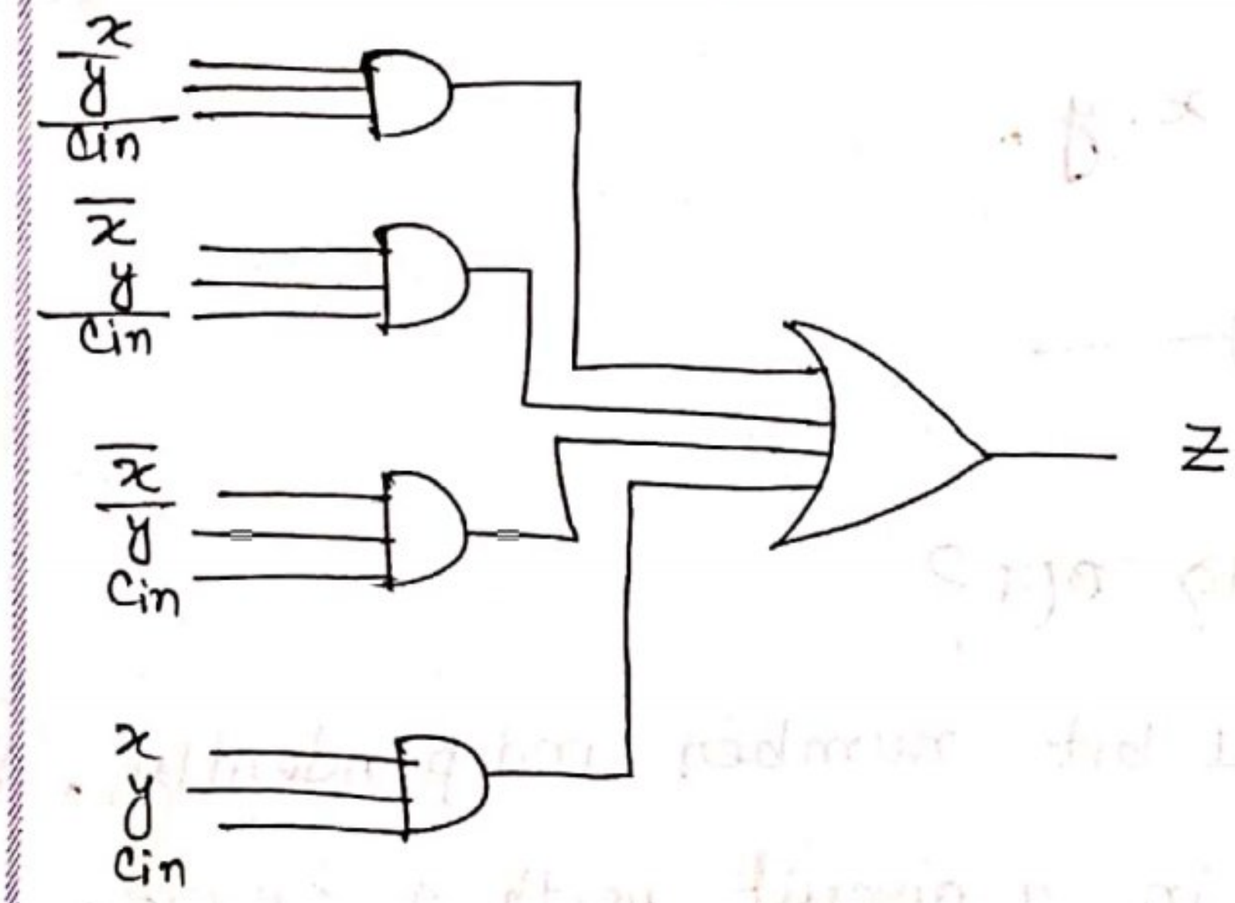
O/p : z, Cout

Sum 1 পাব যখন , যেকোনো একটা 1 হবে provided by the others zero , তিনটা input এর মধ্যে , আবার তিনটা 1 হলেও আমরা Sum এ 1 পাব।

$$\therefore \boxed{z = x \cdot \bar{y}\,\bar{C_{in}} + \bar{x} \cdot y \cdot \bar{C_{in}} + \bar{z} \cdot \bar{y} \cdot C_{in} + x \cdot y \cdot C_{in}}$$

Subject.....

Date:..... Time:.....



* $C_{out} = 1$ পাব তখনই যেকোনো দুইটা 1 হলে। এক্ষেত্রে third টা দেখা লাগে না।

$$C_{out} = x.y + y.c_{in} + x.c_{in} + x.y.c_{in}$$

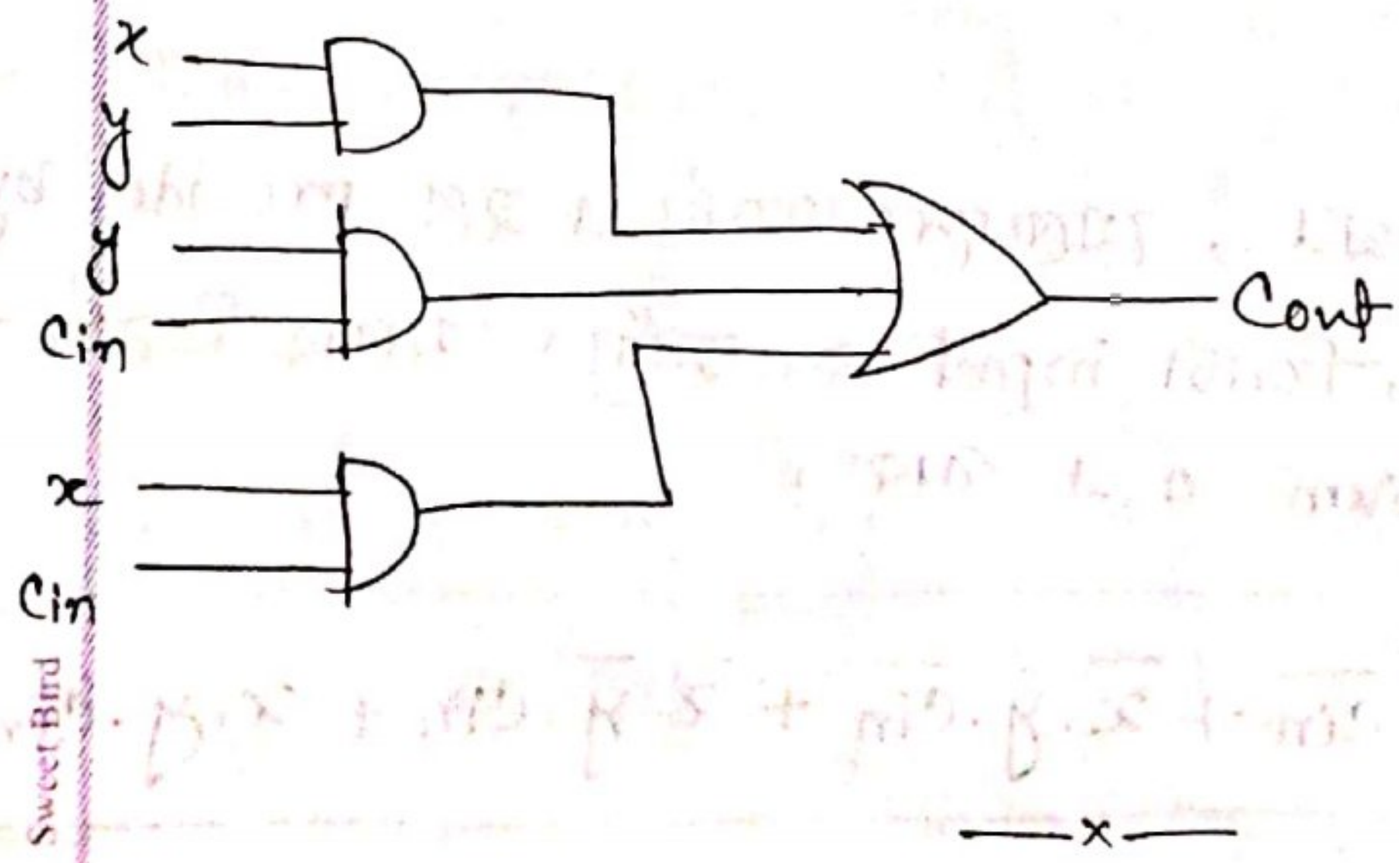
It's redundant

তিনটা 1 হলে:

$x=1, y=1, c_{in}=1$

দিলেও হয়, না দিলেও হয়।

generally কেউ না,
$x.y.c_{in}$



Sweet Bird

Subject .....

Date: ..... Time .....

4 bit Full Adder :

$$i/p \begin{cases} x = x_0 \quad x_1 \quad x_2 \quad x_3 \\ y = y_0 \quad y_1 \quad y_2 \quad y_3 \end{cases} \text{ \& } c_{in}$$

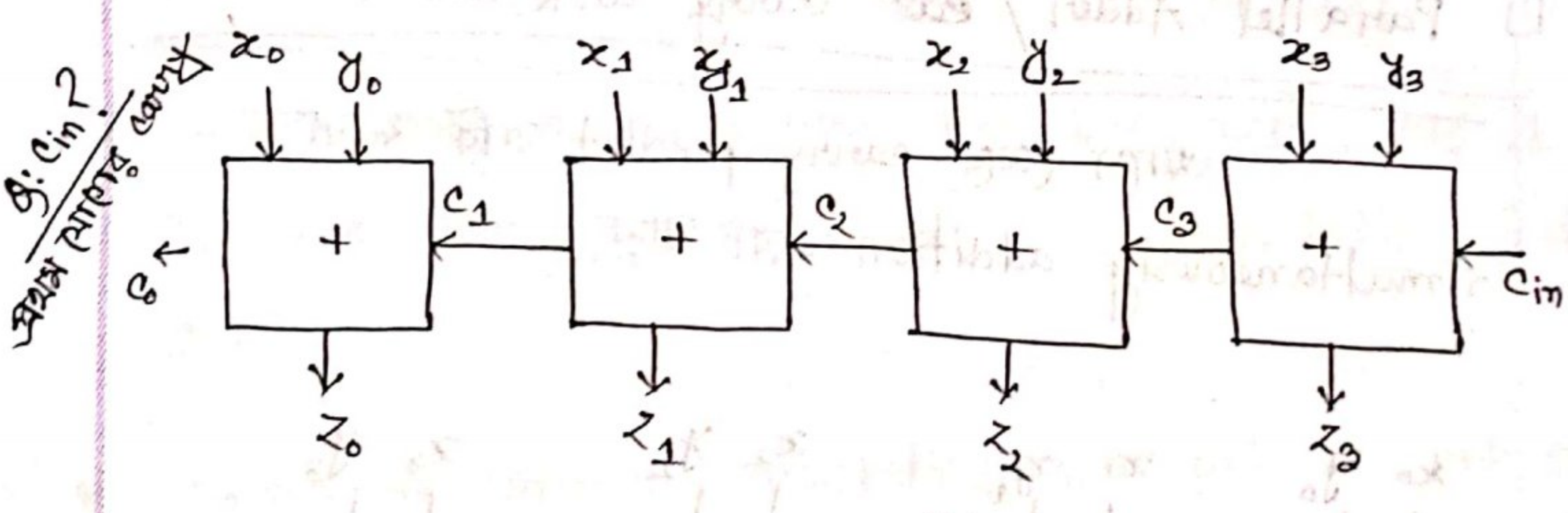
$$o/p \leftarrow z = z_0 \quad z_1 \quad z_2 \quad z_3 \text{ \& } C_0$$

$x = 0010$

$y = 10011$

Addition :

Serial / Slow/ripple Carry Adder :



প্রথম ক্ষেত্রে $c_{in} = 0$ ধরি, c_2 তখন কোনো carry থাকবে।

এক্ষেত্রে ~~adding~~ addition serially হয়। serially হয় দেখে It's slow & এক্ষেত্রে carry টা ripple হয়।

অসুবিধা : একটা addition আরেকটার উপর dependent so একটা না হলে অন্যটা হবে না।

এক্ষেত্রে, LSB টা জানা থাকব। এটা change করলে figure change হবে।

4 bit এর জন্য 4 cycle, n bit এর জন্য n cycle হবে ; time consuming.

Ripple হয় অর্থাৎ ঢেউ এর মত করে pass হয়।
এজন্য এই adder use করা হয় না। Computer fast
calculation করে।

এজন্য parallal adder এ switch করি।

→

62 bit 62 cycle

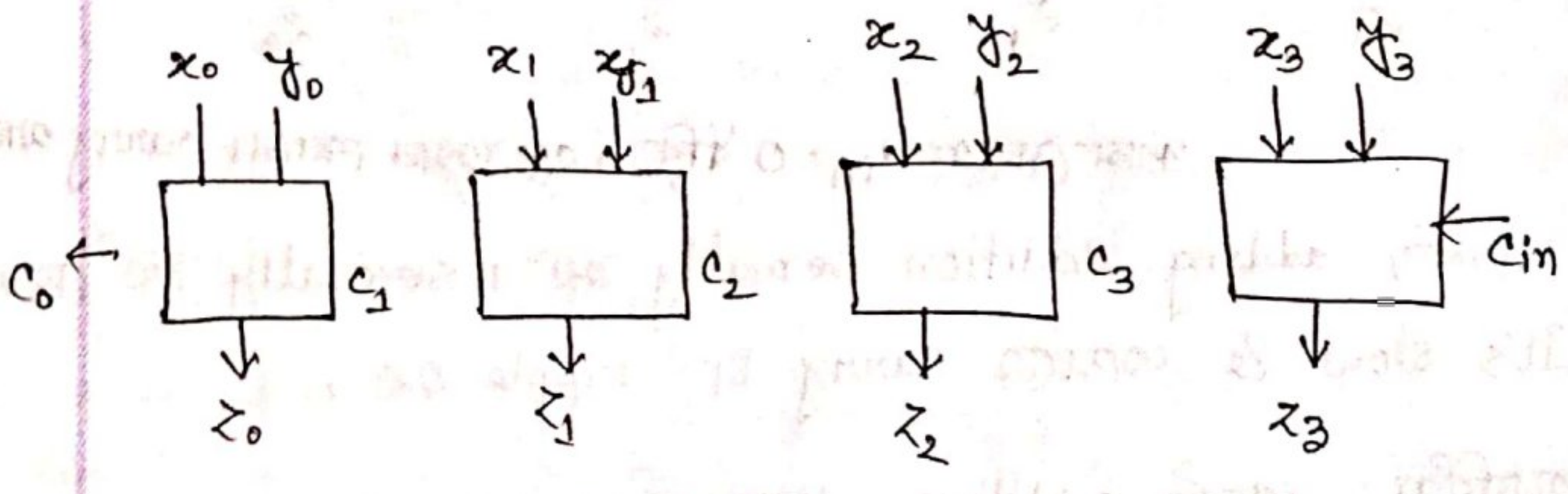
1 cycle এ 2 sec লাগলে

62 × 2 sec

for serial adder

Parallel Adder / ~~Add~~ Carry look ahead adder:

আগে থেকে carry predict করি ফলে
simultaneously addition করা যায়।



generate, $g = x \cdot y$

propagate, $p = x + y$

$$C_3 = x_3 \cdot y_3 + y_3 \cdot C_{in} + x_3 \cdot C_{in}$$
$$= x_3 y_3 + C_{in}(y_3 + x_3)$$
$$= g_3 + C_{in} P_3$$

input পেলেই C_3 generate করা যাবে।

$$C_2 = g_2 + P_2 C_3 = g_2 + P_2 (g_3 + c_{in} P_3)$$

$$C_1 = g_1 + P_1 C_2 = g_1 + P_1 (g_2 + P_2 (g_3 + c_{in} P_3))$$

C_3 বা C_2 এর জন্য অন্যটার পের dependent থাকা লাগে না। c_{in} দিয়ে আর input দিয়েই করা যায়।

$$\therefore C_0 = g_0 + P_0 C_1$$

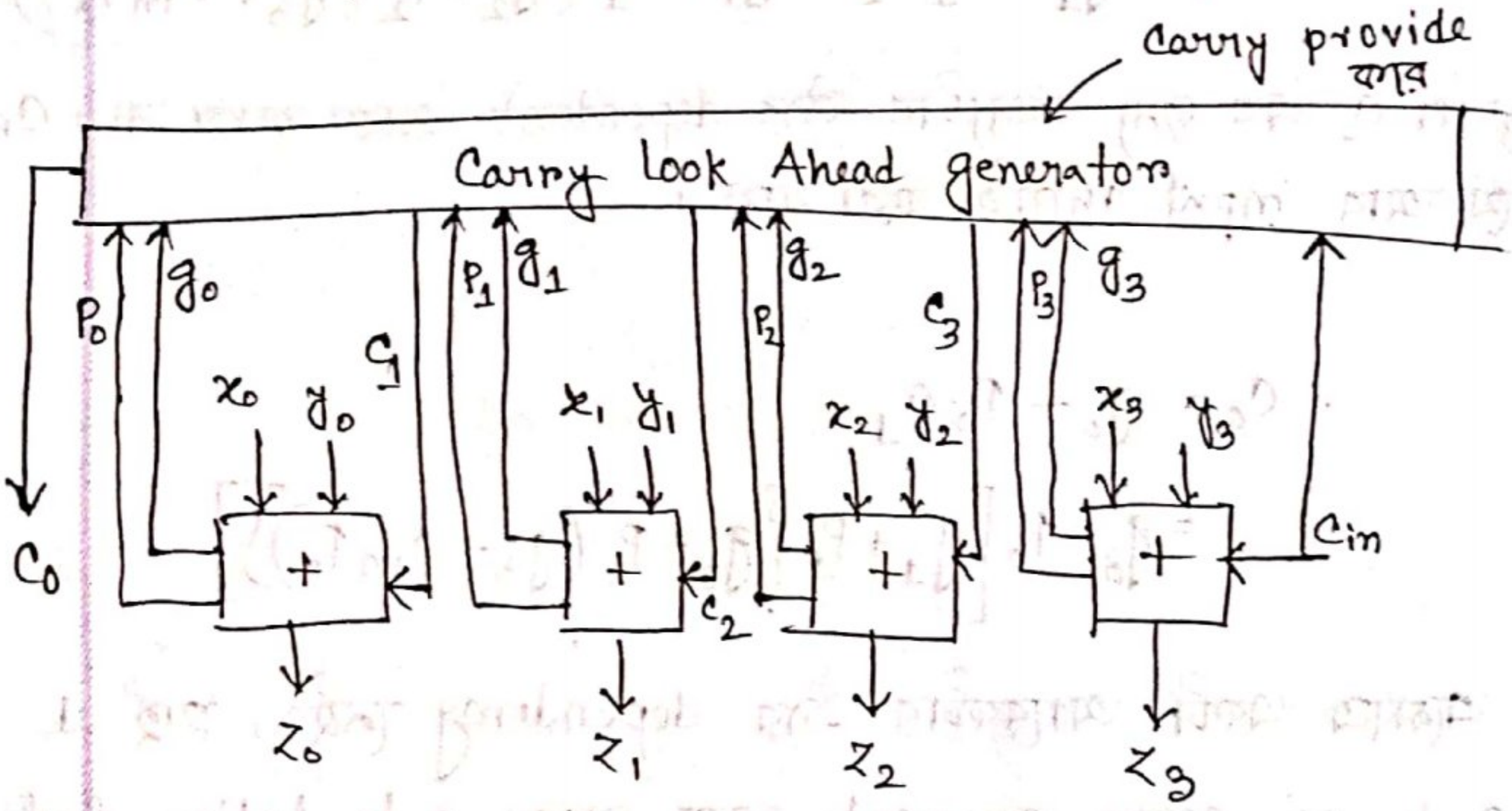
$$= g_0 + P_0 \left[g_1 + P_1 \{ g_2 + P_2 (g_3 + c_{in} P_3) \} \right]$$

এক্ষেত্রে একটা আরেকটার পের dependency নেই, তাই 1 cycle এই carry generate হয়ে যাবে, calculation fast হবে।

* অনেক bit একসাথে calculation করা prb না, prb হল wait করে একটা আরেকটার পের depend করে calculate করা। So, parallal Adder effective.

* 1st cycle এ generate & propagate produce করবে।
2nd cycle এ carry produce করবে।
3rd cycle এ addition হবে।

So, parallal adder এ 4 bit calculation এর জন্য
3টা cycle লাগবে, Serial এর জন্য 4 bit এ
4টা cycle লাগত। So time save হচ্ছে।



<u>total টা parallal Adder</u>

1st cycle : generate & propagate produce
2nd cycle : carry "
3rd cycle : Addition

I/p : $x = x_0\ x_1\ x_2\ x_3$
$y = y_0\ y_1\ y_2\ y_3$ & C_{in}

O/p : $z = z_0\ z_1\ z_2\ z_3$ & C_0

Sweet Bird

Subject.....

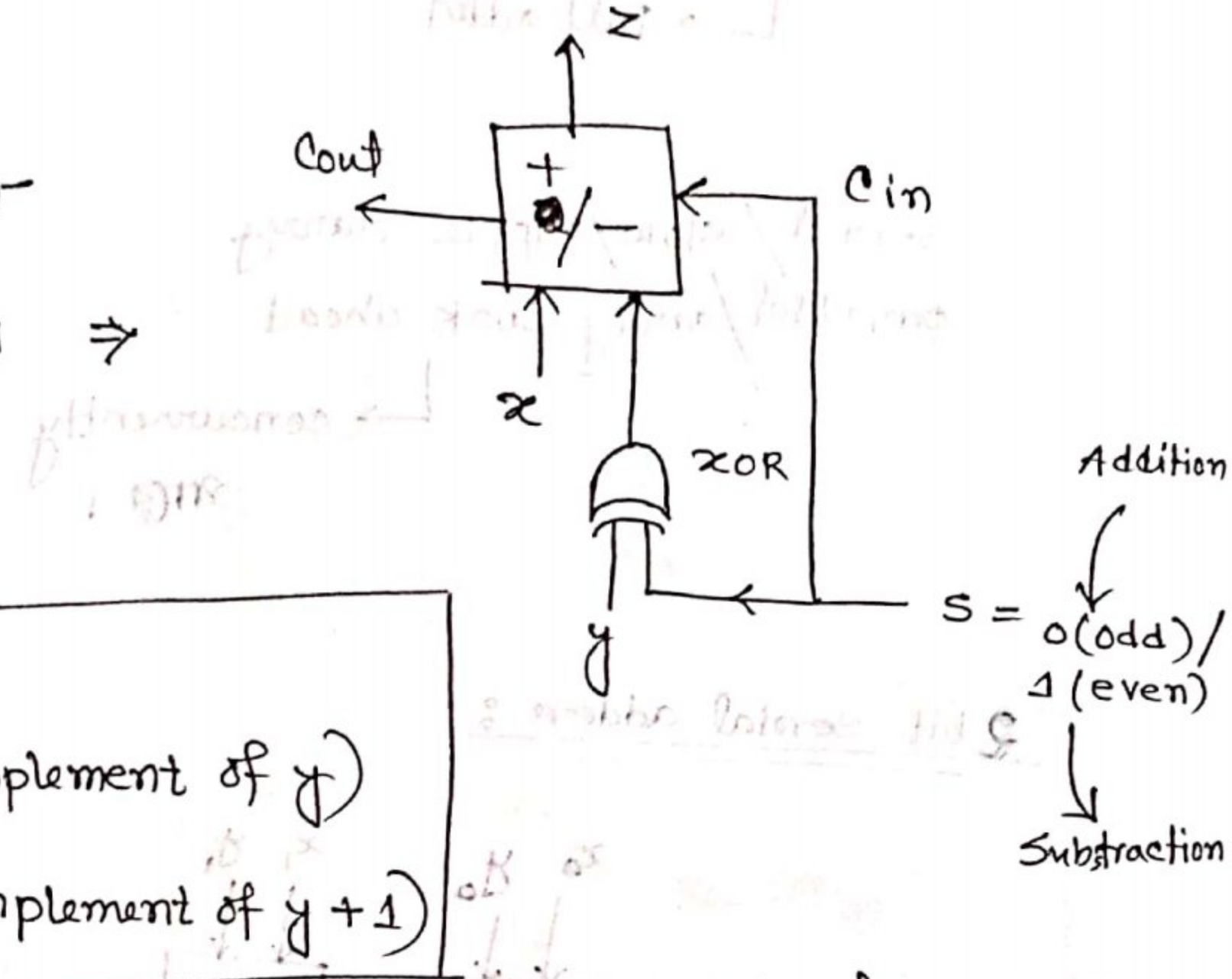
Date: Time:

Subtractor :

Subtractor এর জন্যে আলাদা কোনো circuit নেই। Adder দিয়েই কাজ করি।

Adder দিয়েই subtract এর কাজ করা হয়,

4/16 bit এর জন্য আসলে এই চিত্র একেই just describe করে দিব :



$x - y$

$= x + (2's\ complement\ of\ y)$

$= x + (1's\ complement\ of\ y + 1)$

Subtractor এর জন্য আলাদা কোনো circuit নেই। Adder দিয়েই করি।

* S = 1 হলে subtraction হবে। কিভাবে?
* S = 0 হলে Addition হবে।

XOR

0	0	0
0	1	1
1	0	1
1	1	0

কোনো num কে 0 এর সাথে XOR করলে ঐ num ই পাব আর 1 এর সাথে XOR করলে inverse পাব।

"Week 2 Done"

Sweet Bird

Subject 28.03.2022
Date: Lec - (6) Time:

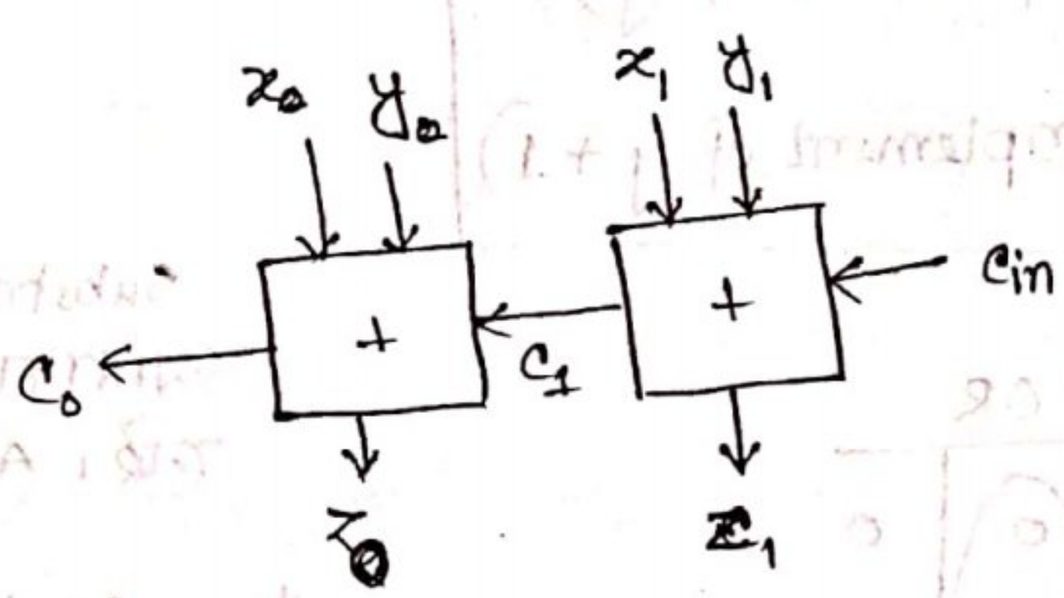
Q: Half Adder থেকে কেন Full adder এ switch করি।

Q: x number কে Half adder করতে পারবে? না পারলে কী করণীয়।

↳ Full adder

serial / slow / ripple carry
parallel / carry look ahead
↳ concurrently process করতে পারে।

2 bit serial adder :



~~i/p : x_1, y_1, c_{in}, x_0, y_0~~

~~o/p : z_0, z_1~~

i/p : $x = x_0 \quad x_1$ & c_{in}
$y = y_0 \quad y_1$

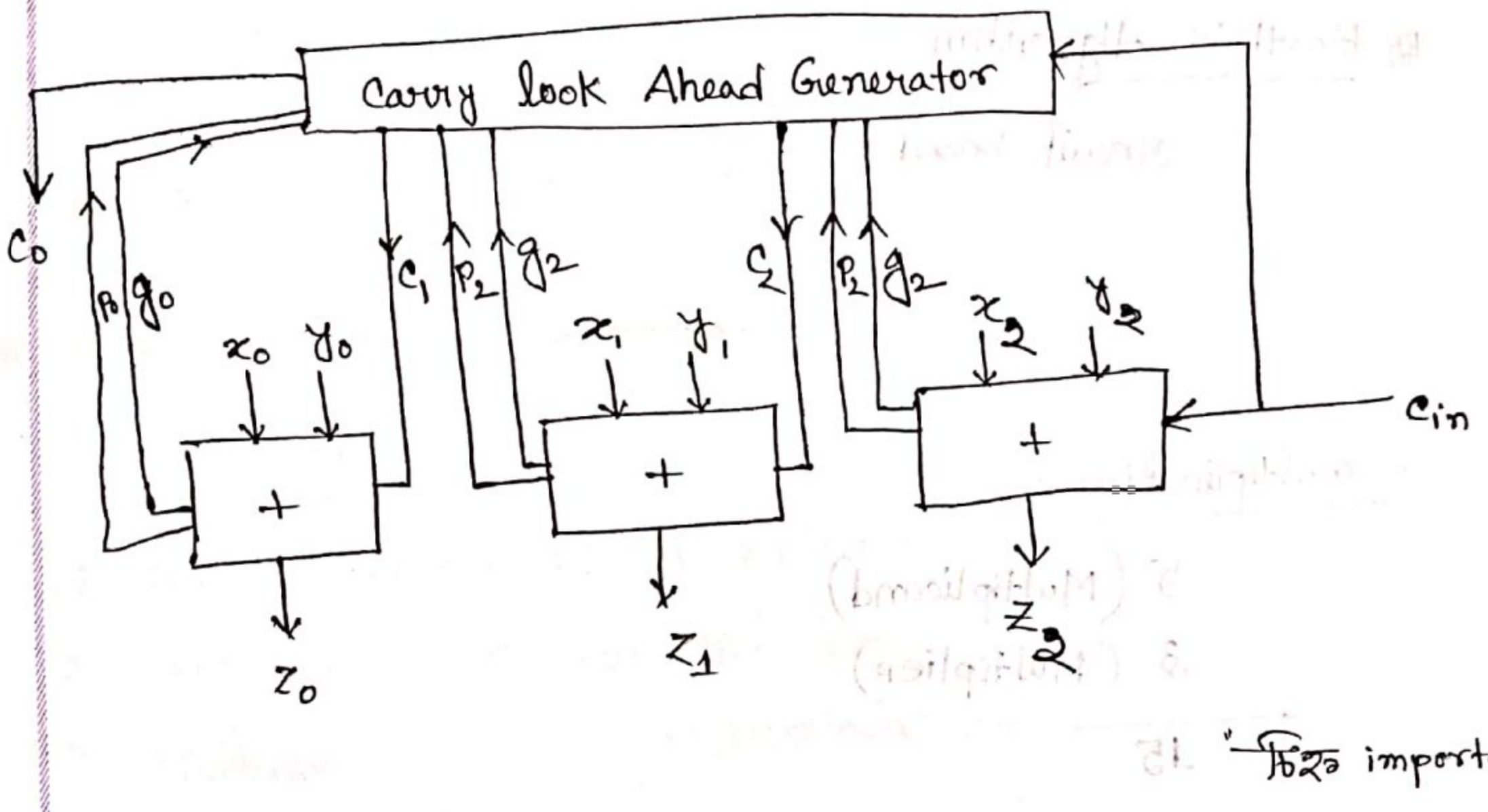
o/p : $z = z_0 \quad z_1$ & c_0

Sweet Bird

Subject.....

Date:.....Time:.....

3 bit Parallal adder :



"খুব important"

i/p : $x = x_0 \quad x_1 \quad x_2$ & c_{in}

$y = y_0 \quad y_1 \quad y_2$

o/p : $z = z_0 \quad z_1 \quad z_2$ & c_0

MSB থেকে operation start হবে

Sweet Bird

30.03.2022
Wed
Lec-(7)

Multiplication :

Booth's Algorithm

circuit based

---

multiplication :

5 (Multiplicand)
3 (Multiplier)

15

255×255 ← 255 কে 255 বার যোগ করা which is is step consuming.
এই নিয়মকে বলে repeated addition.
এটা time consuming

To solve this, Basic Multiplication :

5 → 0101
3 → 0011
---
15 → 0101
0101x
00 00xx
00 0 0xxx
---
0 0 0 1 1 1 1 (15)

partial product

একেত্রে 4 step এই হয়ে যায়, অনেক বড় নম্বর এর ক্ষেত্রে এটা calculation easy.
** এখানে mainly addition হচ্ছে ॥

$2^8 \longrightarrow$ 8 steps

$2^4 \longrightarrow$ 4 steps

$2^n \longrightarrow$ n steps

Addition + Shifting + Addition + ... :

* steps of Basic Multiplication,

partial product (যার কারণে মাল্টিপ্লেই আমরা কাঙ্ক্ষিত result পাই।

Steps

1. Num of steps = bit of multiplier
2. Check the bit of multiplier from LSB.
3. Multiplier = 1 $\longrightarrow$ Multiplicand Itself
   Multiplier = 0 $\longrightarrow$ 0
4. Add all partial products.

Register required :

We don't store partial product in register.

Q: $2^3 \times 2^5$ গুণের জন্য accumulator size কত হবে?

$2^3 \times 2^5 \Rightarrow 5+3 = 8$

→ store করতে

Accumulator

0000	0000	→ 8 bit

প্রতিটা step এ store করে add করি, এই storing কে procedure এর পুরু থেকে বলে accumulator.

Accumulator

0000	0000

এক্ষেত্রে right shift

0101

\+

0101 ⑧ } step 1

0010 1000 }

\+ 0101

0111 1000 } step 2

(shift) 0011 1100 }

0001 1110 } step 3

0000 1111 } step 4

zero থাকলে add করব না just shift করব

(15)

Q: Reg 1টা → 4 bit → Addition

Reg 1টা → 4 bit → Multiplication

" 1টা → 8 bit → Accumulator

Sweet Bird

Subject .....

Date: ..... Time .....

Circuit

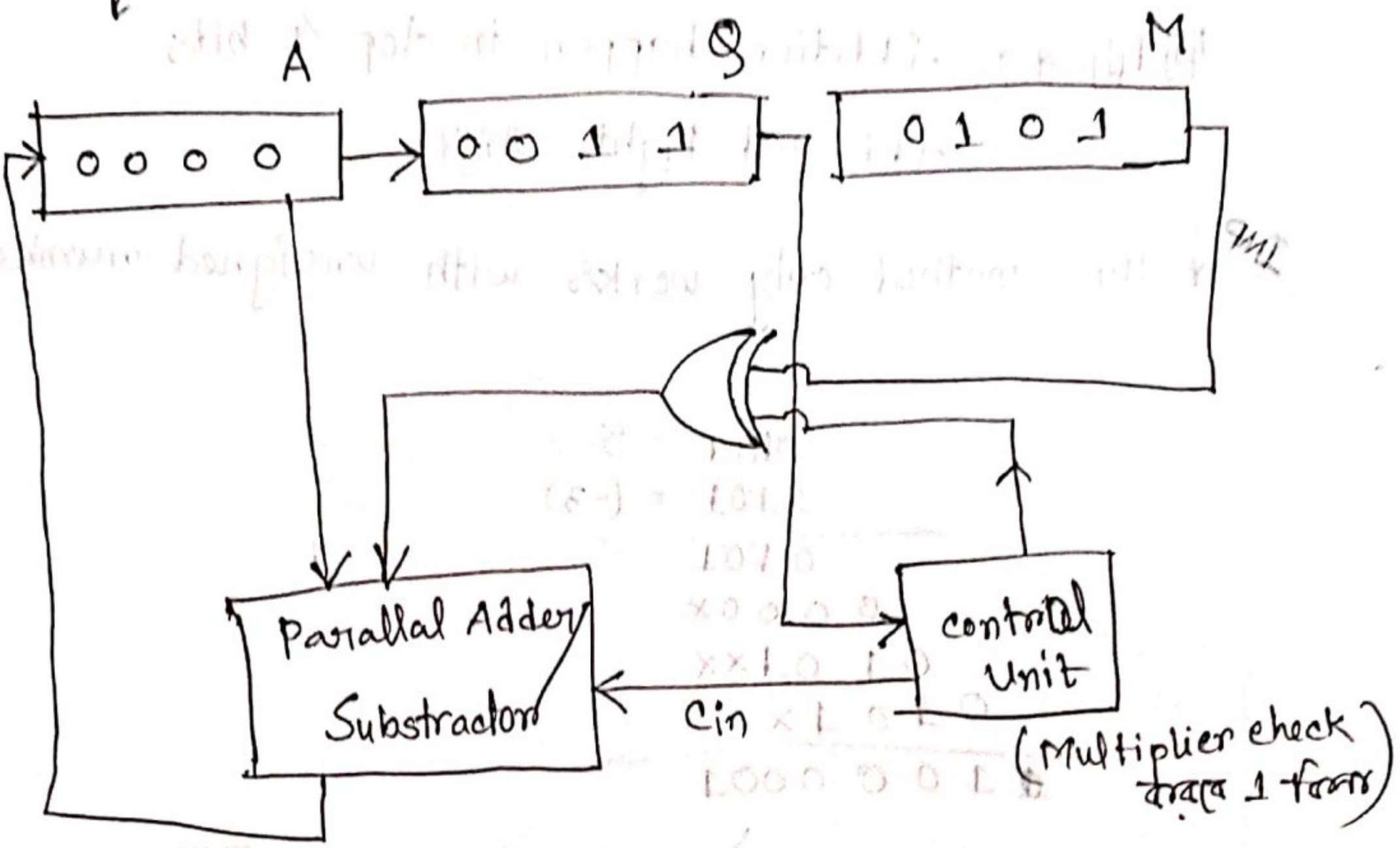
4bit + 4bit = 8 bit ← [last step এ A & Q মিলে result আসে]

এখানে 8 bit না নিয়ে 4 bit কেন নিই?

Multiplicant → M

Multiplier → Q

Accumulator → A



A	Q	M
0000	0011	0101
0101	1001	
0010		
0111	1100	
0011		
0001	1110	
0000	1111	

Sweet Bird

Subject
Date Time

$Z = 0101$

$Z = 0011$

Accumulator Calculation

** Addition happen in top 4 bits

Add and Right Shift

INP * this method only works with unsigned number.

```
    0101 = 5
    1101 = (-3)
-------------
    0101
  0 0 0 0x
  0 1 0 1xx
  0 1 0 1x
-------------
1 1 0 0 0001
```

→ (-63) আসে যা ভুল

negative number এর partial product ও negative আসে।

কিন্তু আমরা partial product এ zero আনতাম।

এখানে mainly one use করা হয় (এর আগে থেকে partial product এর MSB কত। যদি 1 হয় তখন shifting এর time এ 1 add নয়)

Q: Why fail?

Sweet Bird

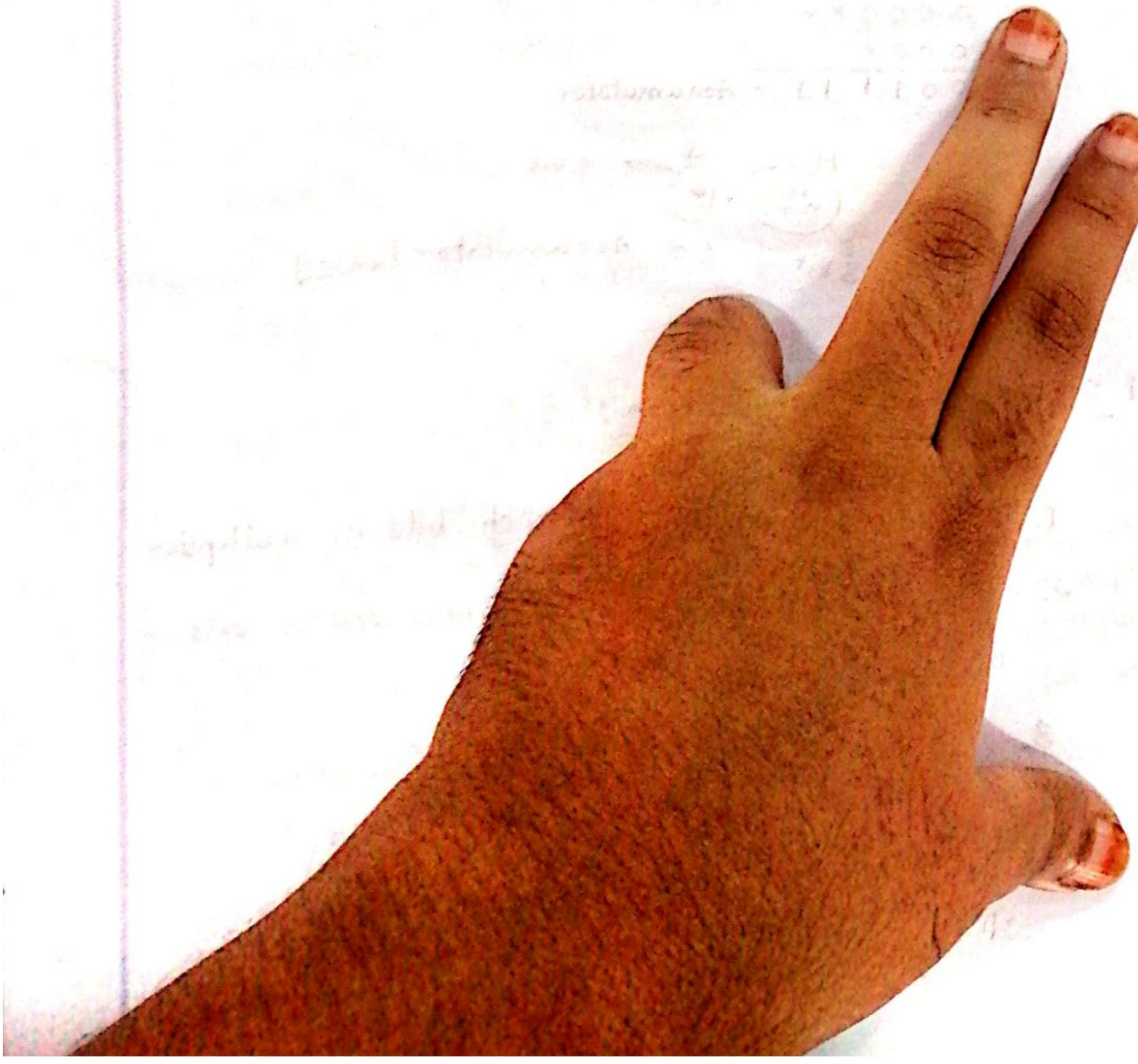
Subject .....

Date: ..... Time: .....

Q: Why fail ?

→ When you multiply neg number, your partial product will be negative.

negative numbers has infinite leading 1 .



Subject: Lec - (8)

Date: 14.05.22 Time: Sat

After Eid-ul-Fitre Vacation.

5 → Multiplicant (M)

3 → Multiplier (Q)

0101 = 5

0011 = 3

0101

0101x

0000xx

0000x

00011111 = Accumulator

M → 255 × 8 ← Q → 4 bit

8 bit

= Accumulator 12 bit

Step's of Booth's Algorithm:

1. Number of steps = Number of bits in multiplier.
2. At each step, examine two consecutive bits of multiplier from LSB.

If transition $0 \longrightarrow 1$ (Substract M)

$1 \longrightarrow 0$ (Add M)

প্রতিটা step এ Right Shift হবে।

Sweet Bird

Subject..

Date:........................Time:........................

Booth's Algorithm

(5 × 7)

+5 : 0101 7 : 0111

−5 : ~~1101~~ 1011 −7 : 1001

	A (0) Accumulator	Q(7) Multiplier	Q_{-1}	M(5) Multiplicand
	0000	0111		0101
0 to 1 Sub M RS	1011 1101	1011	1	
1 to 1 RS	1110	1101	1	
1 to 1 RS	1111	0110	1	
1 to 0 Add M RS	+0101 0100 0010	 0110 0011		

right shift করে 1 হবে।

carry ignore করি। Why?

সামনে zero দিল তাই সামনে এটা positive Number

কয়টা step হবে তা depend করে Multiplier কত bit এর।

Ex (5 × -7)

	A (0)	Q (-7)	Q_{-1}	M (5)
	0000	1001	0	0101
	+1011			
0→1	1011	1001		
SubM				
RS	1101	1100	1	
	+1010			
1→0	0111	1100		
AddM				
RS	0011	1110	0	
0→0	0001	1111	0	
RS				
	+1011			
0→1	1100	1111		
SubM				
RS	1110	0111	1	

(Math ঠিক করতে হবে)

Main Ans:

11011101 = Accumulator

Sweet Bird

Lec - (9)
Date: 16.05.22
Mon

(ii) $5 \times -7 = ?$

$+5 = 0101$ $+7 = 0111$

$-5 = 1011$ $-7 = 1001$

Steps	A(6)	Q(7)	Q_{-1}	M(5)
	0 0 0 0	1 0 0 1	0	0 1 0 1
0 to 1	1 0 1 1			
Sub M	1 0 1 1	1 0 0 1		
RS	1 1 0 1	1 1 0 0	1	
1 to 0	0 1 0 1			
Add M	0 0 1 0	1 1 0 0	1	
RS	0 0 0 1	0 1 1 0	0	
0 to 0 RS	0 0 0 0	1 0 1 1	0	
0 to 1	1 0 1 1			
Sub M	1 0 1 1	1 0 1 1		
RS	1 1 0 1	1 1 0 1	1	

Answer
Accumulator : 1101 1101 = -35 → (Ans)
Ans নেওয়ার সময় 2's complement করে ans নিতে হবে।

0010 0011 = 35

4th Week

Subject

Date Time

Ques: $14 \times 7 = ?$ by Booth's algo.

5 bit 4 bit

Steps = num. of bits in Multiplier

অর্থাৎ, Multiplier 7 কে বলে 5 bit এ represent করবি।

So, step 5 টা।

Example: $(-5 \times 7) = ?$

Booth's Devision → Restoring

→ Non-Storing

Dividend

Divisor ← 6) 25 (4 → Quotient

24

1 → Remainder

We don't use repeated subtraction, as ~~show~~ slow.

(Left shift)

Exam Important MUST ⟹ Booth's Algo

Subject.....

Date:.....Time:.....

STORING Or When step successful or unsuccessful.

$$
\begin{array}{l|l|l|l}
4 & \overleftarrow{3}\ 2 & 0 & \text{LS} \\
 & 4 & & \text{Sub} = \rightarrow +ve,\ S,\ Q=1 \\
 & \hline -1 & & \quad = \rightarrow -ve,\ US,\ Q=0 \\
 & +4 & & \\
 & \hline 3 & &
\end{array}
$$

Time loss হয়, ৫ store করে & অনেক বেশি case এই unsuccessfull হয়।

Steps of Booth's Division Restoring :

I) ~~Step = bit num of Dividend~~

Number of steps = Num. of bits in Dividend

II) At each step, Left Shift the Dividend and then Subtract the Divisor.

4 | 6 4 | 1

6

$$
\begin{array}{l|l|l}
4 & \overleftarrow{6}\ \overleftarrow{4} & 1 \\
 & 4 & \\
\hline & 2 &
\end{array}
$$

If, R = (+ve), Successful : Quotient = 1

R = (-ve), Unsuccessful : Quotient = 0

Restoration required, by adding back the divisor

Sweet Bird

Microprocessor add back কেন করে?
→ ∴ overwrite করে তাই add back করে restore করে। এতে একটু time বেশি লাগে।

ex: 7 ÷ 3 = ?

+7 = 0111   +3 = 0011

−7 = 1001   −3 = 1101

	Accumulator A (0)	Dividend Q (7)	Divisor M (3)
	0000	0111	0011
LS	0000	111_	
Sub (Divisor) M	1101 1101	1110 ↓ unsuccessful	
∵ MSB=1, −ve unsuccessful			
Restoration	0000	1110	
LS	0001	110_	
Sub M	1101 1110		
MSB=1, −ve unsuccess → Restore	0001	1100	
LS	0011	100_	
Sub M	1101 0000		
MSB=0, +ve Successful	0000	1001	
LS	0001	001_	
Sub M	1101 1110	001_	
MSB=1, −ve unsu Restore	0001	0010	

LS করার ~~...~~ পর যেটা পাইছি তা হবে। (← Restoration)

0001 → ভাগশেষ − R

0010 → ভাগফল − Q

Re-storing division :

By the time procedure is over, the dividend will be empty, at every step, we start filling the quotient over here at last, the reg itself will be quotient.

By the time procedure is over, Sum is completely out, if anything remains there it will be remainder.

```
5 | 6 | 1
    5
  ---
    1
```

Q: $6 \div 2 = ?$

Ans:
Accumulator : 0000 ⟶ Remainder
Dividend : 0011 ⟶ Quotient

P.T.O→

Subject
Date: Time

(*) $6 \div 2 = ?$

$+6 = 0110$ $\quad$ $+2 : 0010$

$-6 = 1010$ $\quad$ $-2 : 1110$

	Accumulator A(0)	Q Dividend Q(6)	

Week - 5
Subject Lec (11)
Date 11.05.2022 Time

Non Restoring Devison:

1. Num of bit steps = num. of bits in devidend
2. Left Shift & subtract M (divisor).
3. SubM: +ve. Success, Q = 1
   -ve. Unsuccessful, Q = 0, Next step. Add M
4. for last step,
   -ve, Unsuccessful, Q = 0, Restore.

ex: $7 \div 3 = ?$

7 : 0111 3 : 0011

-7 : 1001 -3 : 1100

Accumulator A(0)	Dividend : 7 Q (0111)	Divisor (3) (M	
0000	0111	0011	
0000 1101 1101	111_ 1110		LS Sub M MSB - -ve, Unsuccessful Next Add M
1011 0011 1110	~~0111~~ 111_ ~~0011~~ 1100		LS Add M MSB - -ve, Unsuccessful Next Add M
1110 0011 0000	100_ 1001		LS Add M MSB = -ve, Successful

Week - 5

Subject Lec-(11)

Date 11.05.2022 Time

Non Restoring Devison:

1. Num of steps = num. of bits in dévidend
2. Left Shift & subtract M (divisor).
3. SubM: +ve, Success, Q = 1
   -ve, Unsuccessful, Q = 0, Next step, Add M
4. for last step,
   -ve, Unsuccessful, Q = 0, Restore

ex: $7 \div 3 = ?$

7 : 0111 3 : 0011

-7 : 1001 -3 : 1100

Accumulator A(0)	Dividend : 7 Q (0XXX)	Divison (3) (M	
0000	0111	0011	
0000 1101 1101	111_ 1110		LS Sub M MSB - -ve, Unsuccessful Next Add M
1011 0011 1110	~~0111~~ 111_ ~~0011~~ 1100		LS Add M MSB = -ve, Unsuccessful Next Add M
1110 0011 0000	100_ 1001		LS Add M MSB = -ve, Successful

	A	Q
13	0001	0000
Sub M	1101	0011
	1110	
MSB=-ve, unsuccessful Restore	0001	0010
	1	2
	Remainder	Quotient

এখানে আর unsigned. Signed হলে বা ex division এর ক্ষেত্রে sign number এর algorithm different.

// এভাবে বা পরে পড়বে "

এই process is only for unsigned number.

4th Non-Restoring :

$6 \div 2 = ?$

$+6 = 0110$ $\quad +2 = 0010$

$-6 = 1010$ $\quad -2 = 1110$