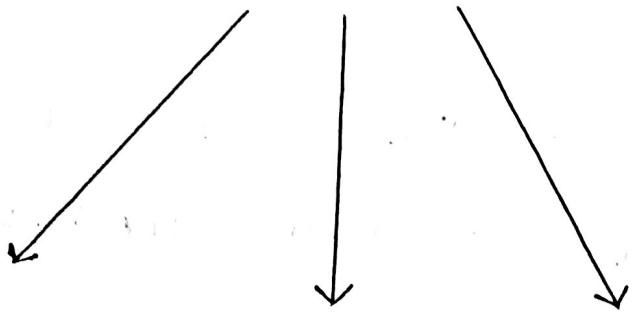
DSD

22 April 2017

CSE 3109
Digital System Design

Book

Digital Logic and Computer Design — M. Morris Mano



ALU — Booth — SAP - I + II

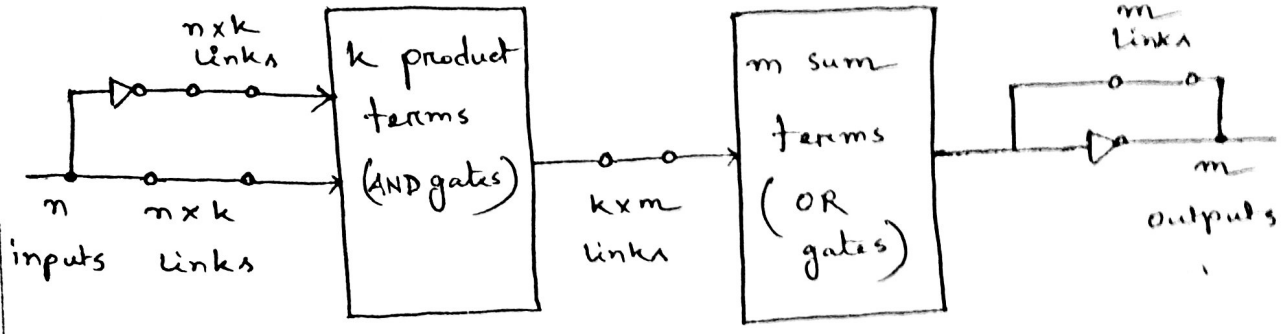
D. M. Anisuzzaman

rajon99@gmail.com

7B04 → 9A05 (LAB room shifted)

23 April 2017

PLA (Programmable Logic Array)



Imp **

* Two simplified functions are given below. show all the steps required for their PLA implementation.

$$F_1(A,B,C) = \Sigma(4,5,7)$$
$$F_2(A,B,C) = \Sigma(3,5,7)$$

$$F_1 = AB' + AC$$

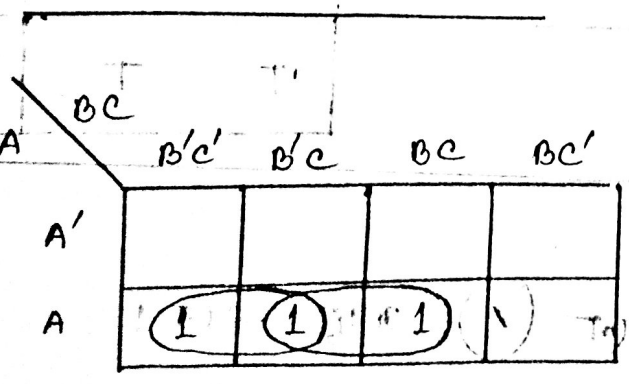
$$F_2 = AC + AB$$

post

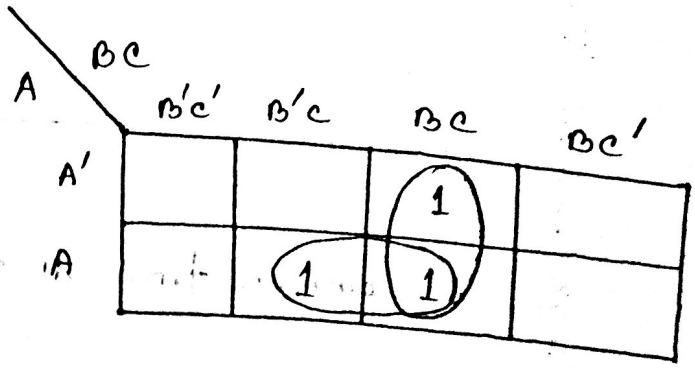
(a) Truth Table

A	B	C	F_1	F_2
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	1
1	0	0	1	0
1	0	1	1	1
1	1	0	0	0
1	1	1	1	1

(b) map simplification



$F_1 = AB' + AC'$



$F_2 = AC + BC$

c) PLA Program table :

	Product terms	Inputs			Outputs		
		A	B	C	F_1	F_2	
AB'	1	1	0	—	1	—	
AC	2	1	—	1	1	1	
BC	3	—	1	1	—	1	
					T	T	T/C

* Common term এর ক্ষেত্রে (1) বসবে।

* (1) এর ক্ষেত্রে T এবং Opposite হলে, C (Complement).

Example 5-6

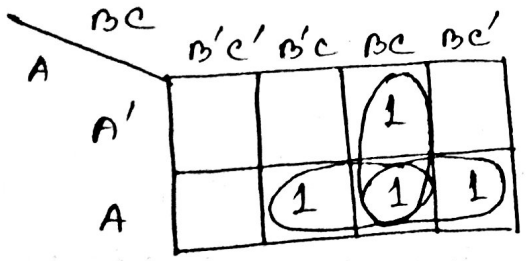
A combinational circuit is defined by the functions:

$$F_1(A,B,C) = \Sigma(3,5,6,7)$$
$$F_2(A,B,C) = \Sigma(0,2,4,7)$$

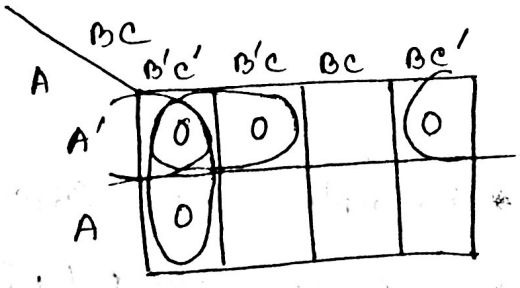
Implement the circuit with a PLA having 3 inputs, 4 product terms and two outputs.

Truth Table: (skip करना है)

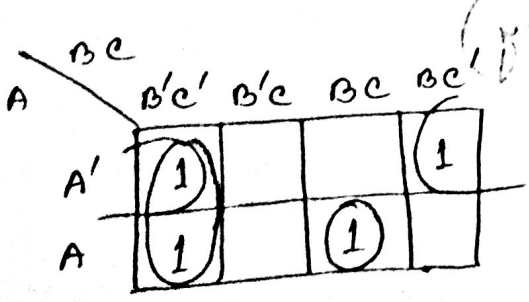
(a) map simplification:



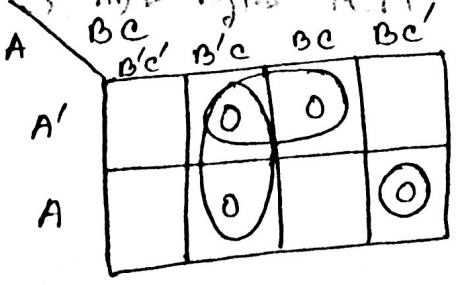
$F_1 = AC + BC + AB$



$F_1' = B'C' + A'B' + A'C'$



$F_2 = B'C' + A'C' + ABC$



$F_2' = B'C + A'C + ABC'$

(b) PLA Program table

	Product terms	Inputs			Outputs	
		A	B	C	F_1	F_2
$B'C'$	1	-	0	0	1	1
$A'C'$	2	0	-	0	1	1
$A'B'$	3	0	0	-	1	-
ABC	4	1	1	1	-	1
					C	T

T/C

* Example - 5.31 থেকে last বাদ গেছিল।
* PLA থেকে একটি (1টি) Question must থাকবে।

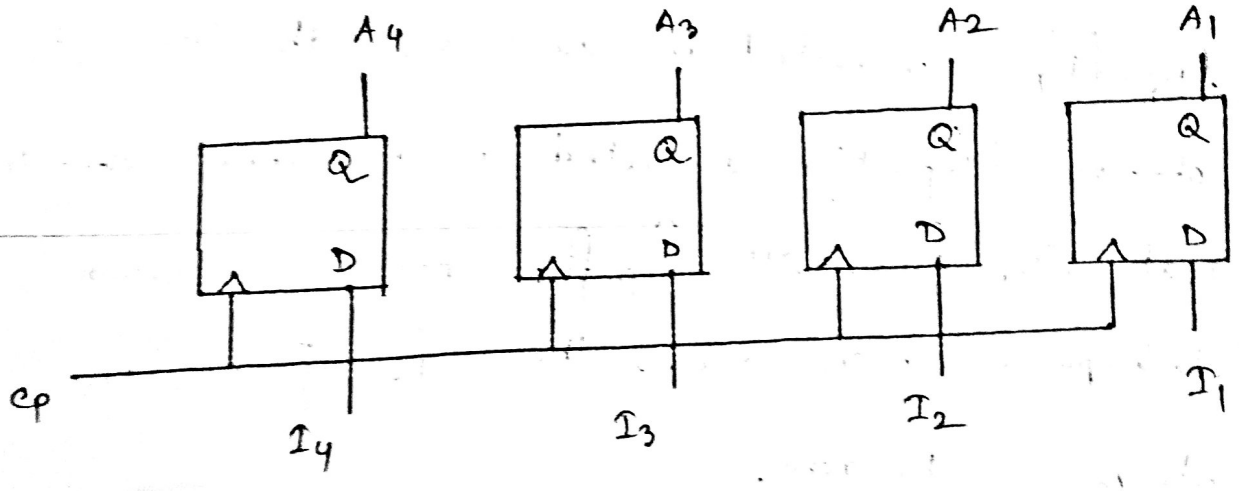
Q. PLA কাকে বলে ? (Theory)

27 April 2017

Register

Register:

A register is a group of binary storage cells suitable for holding binary information. A group of flip-flops constitutes a register, since each flip-flip is a binary cell capable of storing one bit of information. An n-bit register has a group of n flip-flops and is capable of storing any binary information containing n-bits.



4-bit register

Register is enabled as long as cp = 1. when cp goes to 1, the input info is loaded into the register. if cp remains at 0, the content of the register is not changed.

Shift Register:

A register capable of shifting its binary info either to the left or to the right is called a shift register. The logical configuration of a shift-register consists of a chain of flip-flops connected in cascade, with the output of one flip-flop connected to the input of the next flip-flop. All flip-flops receive a common clock-pulse which causes the shift from one stage to the next.

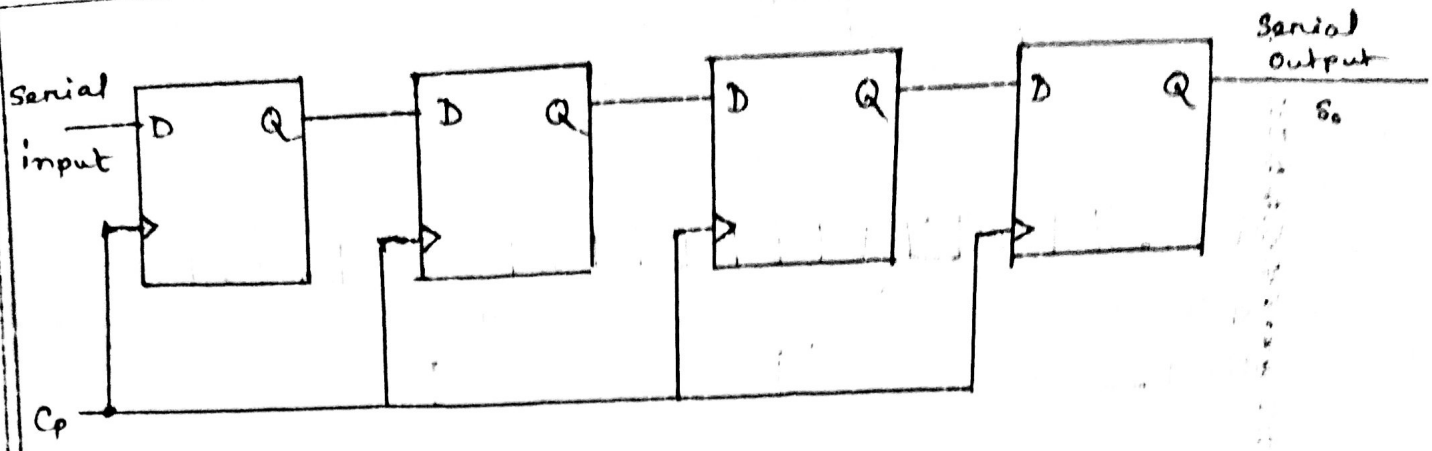
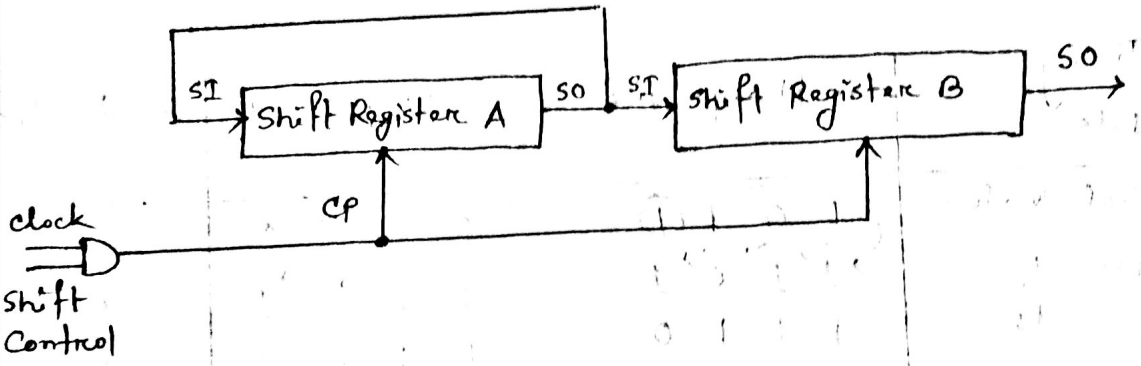


fig. shift Register

Serial Transfer:

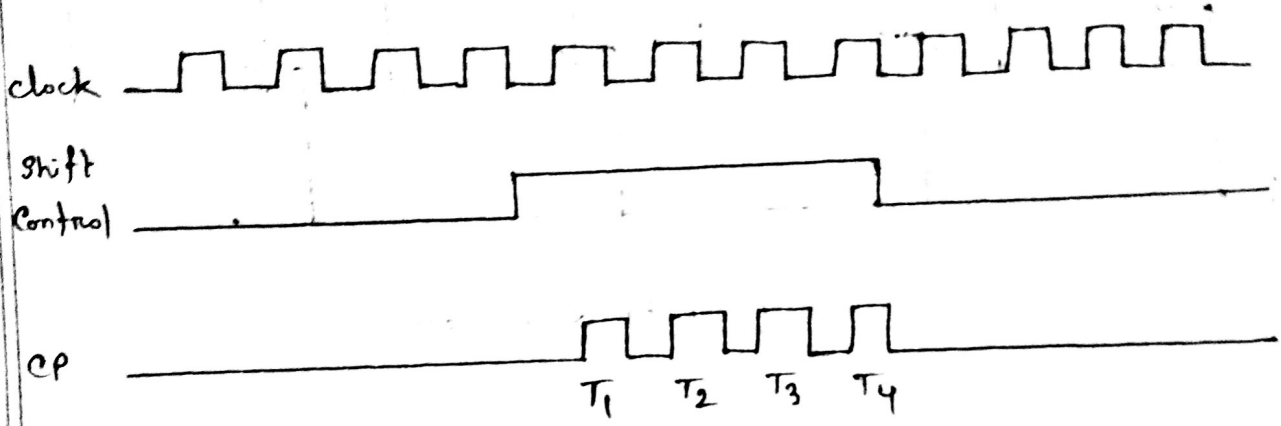


Block Diagram

The shift control input determines when and by how many times the registers are shifted.

Suppose the shift registers have four bits each.

Timing Diagram



Serial Transfer Example: (Logical Question)

Timing Pulse	Shift Register A	Shift Register B	Serial Output of B
Initial Value	1 0 1 1	0 0 1 0	0
After T_1	1 1 0 1	1 0 0 1	1
" T_2	1 1 1 0	1 1 0 0	0
" T_3	0 1 1 1	0 1 1 0	0
" T_4	1 0 1 1	1 0 1 1	1

After the 4th shift, the shift control goes to 0 and both register A and B have the value 1011.

Read Only Memory (ROM)

A ROM is essentially a memory device in which a fixed set of binary info is stored. The binary info must first be specified by the user and is then embedded in the unit to form the required interconnection pattern.

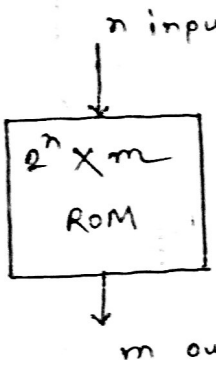
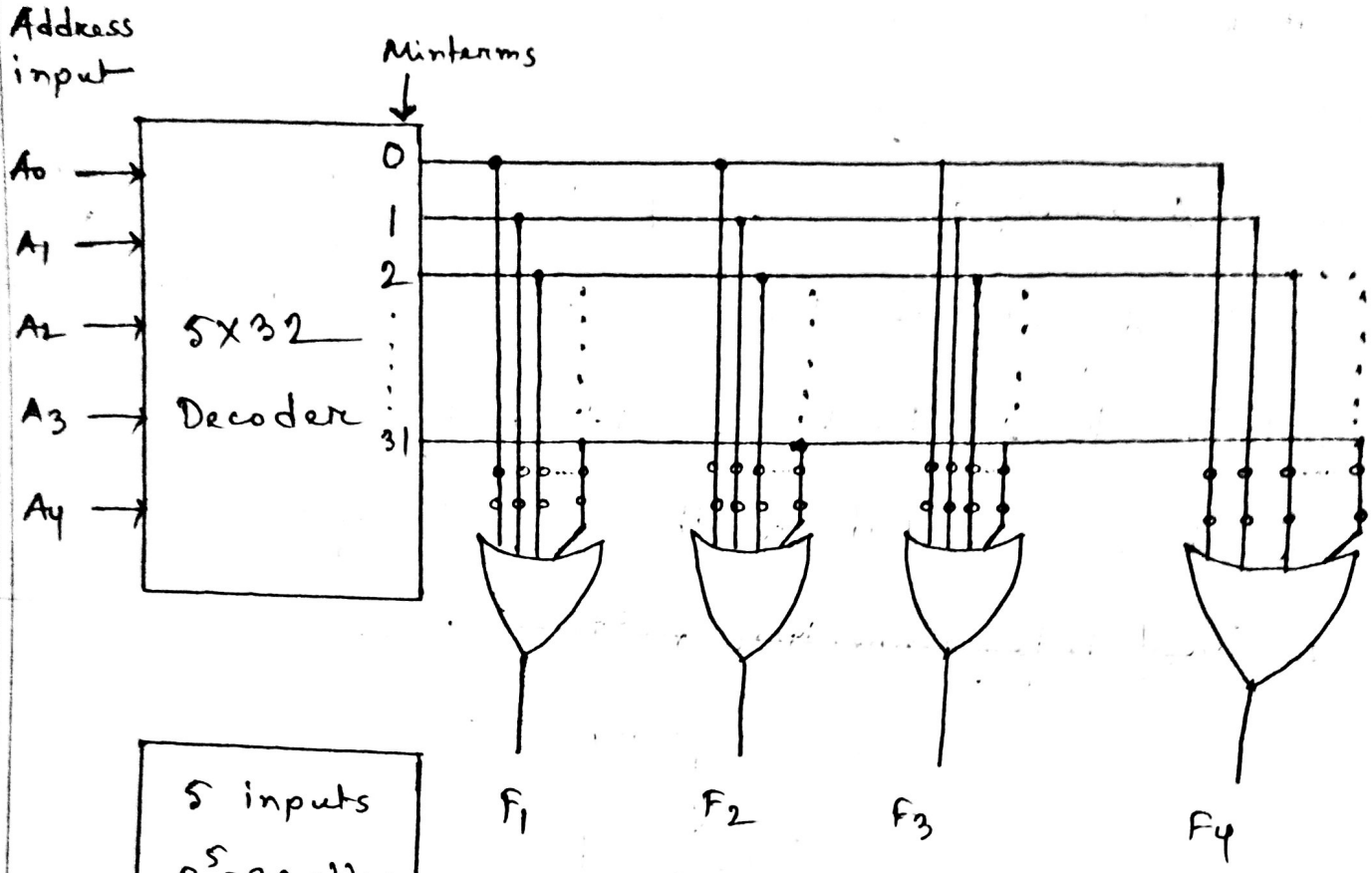


Fig: ROM block diagram

Consider a 32X4 ROM. The unit consists of 32 words of 4-bit each. If the input address is 00000, word no. 0 is selected and it appears on the output line.



5 inputs

$2^5 = 32$ addresses

$4 \times 2^5 = 128$ links

Logic construction of a 32X4 ROM

29 April 2017

Random-Access-Memory (RAM)

RAM is the place in a computer, where the operating system, application programs and data in current use are kept so that they can be quickly reached by the computer's processor.

Vs. Read Only-Memory (ROM)

ROM is a type of memory that is as fast as RAM, but has two important differences: It can't be changed, and it retains its contents even when the computer is shut off. It is generally used to start your computer up and load the OS.

RAM

Memory Unit:

Stores binary information in group of bits called words.

Memory Word:

Group of 1's and 0's may represent a number, character(s), instruction or other binary information.

Most computer memories use words that are multiples of 8-bits (byte).

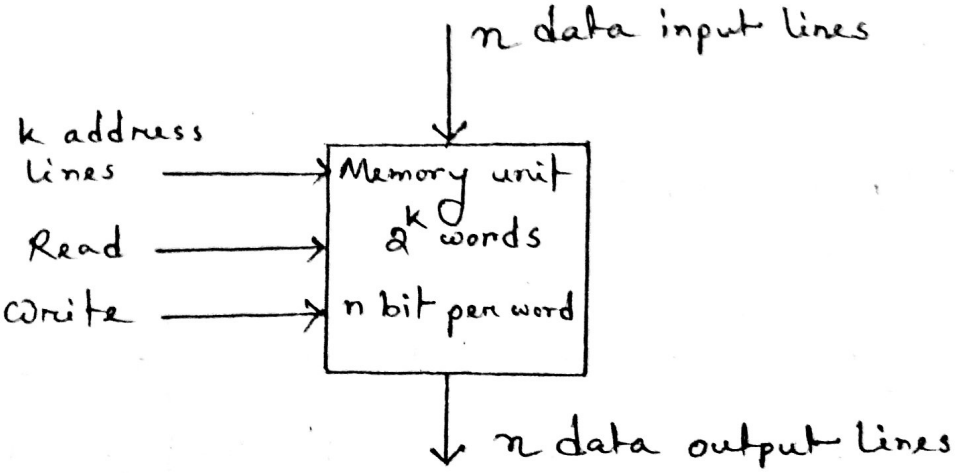


Fig: RAM block diagram

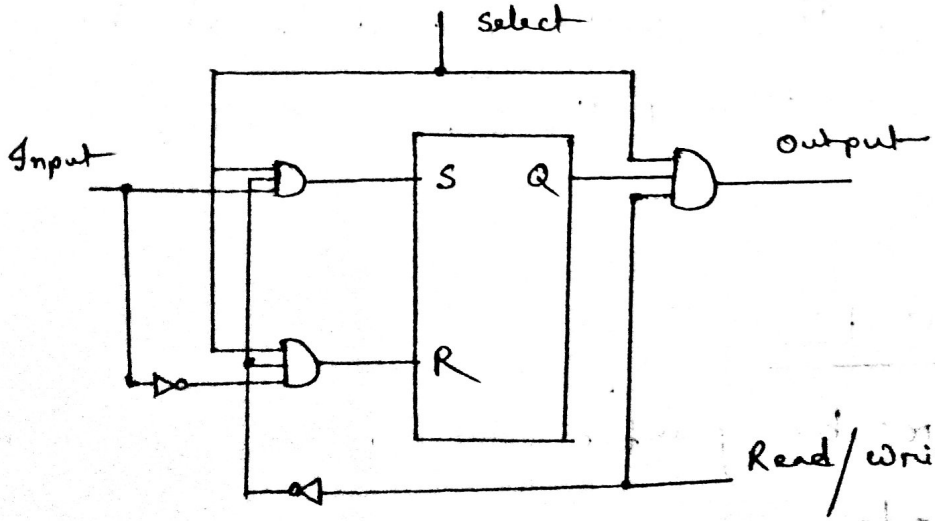


fig: RAM logic diagram

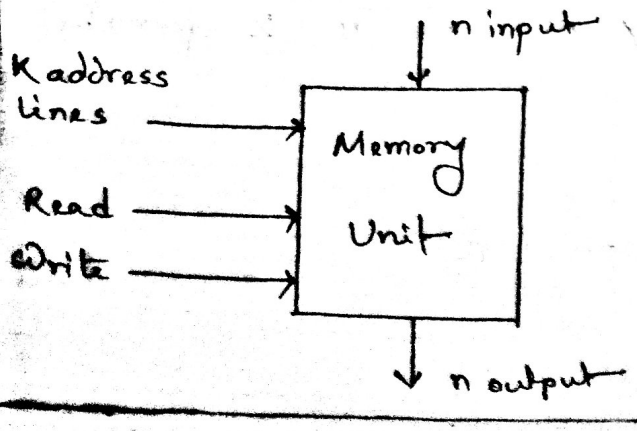
Each word in memory is assigned as address 0 up to $2^k - 1$ (k = no. of address lines)

Memory Address		Memory Content
Binary	Decimal	
0000000000	0	1111011101101111
0000000001	1	1010111011101101
⋮	⋮	⋮
1111111110	1022	1111011011111100
1111111111	1023	0011001100001111

Content of a 1024 x 16 Memory

Random-Access-Memory (RAM)

Write and Read Operation:



To transfer a new word to be stored into memory:

[1] Apply the binary address of the word to address lines.

[2] Apply the data bits that must be stored in memory to the data input lines.

[3] Activate the "Write" input.

To transfer a stored word out of memory:

[1] Apply the binary address of the word to the address lines.

[2] Activate the "Read" input.

Types:

RAM units are mainly available in 2 possible operating modes:-

Static and Dynamic

Static RAM (SRAM)

consists of internal latches that store the binary info. The stored info remains valid as long as power is applied to the unit.

Dynamic RAM (DRAM)

stores the binary info in the form of electric charges on capacitors provided by the MOS transistors. The capacitors must be periodically recharged by refreshing of the dynamic memory every few milliseconds

Vs:

→ DRAM offers reduced power consumption, large integration of units on chip.

→ SRAM is faster, has shorter read and write cycles

SRAM is used in cache.

Disadvantage: high power consumption, low density, expensive.

30 April 2017

ALU Design:

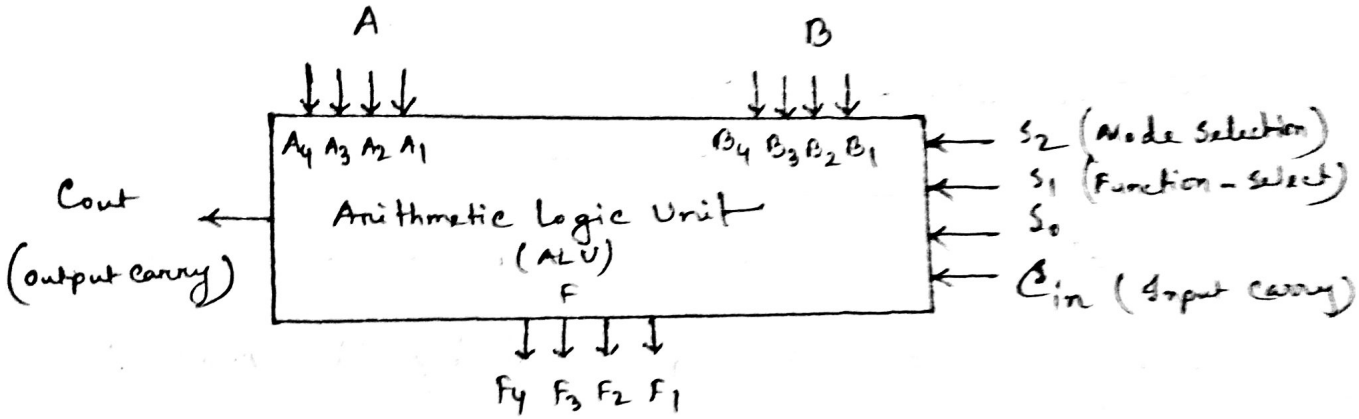


Fig: 4-bit ALU block diagram

(P.T.O.)

9-4

Design of Arithmetic Circuit (xm ও Lab এর জন্য করতে হবে)

Function Select S_1	S_0	C_{in}	X	Y	Output	Function
0	0	0	A	0	$F = A$	Transfer A
0	0	1	A	0	$F = A+1$	Increment A
0	1	0	A	B	$F = A+B$	Add B to A
0	1	1	A	B	$F = A+B+1$	Add B to A plus 1
1	0	0	A	B'	$F = A-B-1$	Add 1's Complement of B to A
1	0	1	A	B'	$F = A-B$	Add 2's Complement of B to A
1	1	0	A	All 1	$F = A-1$	Decrement A
1	1	1	A	All 1	$F = A$	Transfer A

Now,

$X = A_i$

$Y = S_1' S_0 B + S_1 S_0' B' + S_1 S_0$

$= S_0 B_i + S_1 B_i'$

$S_2 = 0 \rightarrow$ Arithmetic Mode

$S_2 = 1 \rightarrow$ Logic Mode

* C_n এর মান দিলে গেলেও "Z" বাদ যাবে final xm এ।
Lab এর বিষয়বস্তুগুলি করতে হবে।

* Design করতে চললে figure আঁকাসহ সব করতে হবে।

04 May 2017

Effect of Output Carry

To investigate the effect of the output carry we expand the arithmetic circuit to n-bits so that Cout = 1, when the output of the circuit is equal to or greater than 2^n. An output carry of 1 after an addition operation denotes an overflow condition. The following table lists the conditions for having an output carry in the circuit.

	Function Set			Arithmetic function	$C_{out} = 1$ if	Comments
	S_1	S_0	C_{in}			
a)	0	0	0	$F = A$		C_{out} is always 0
b)	0	0	1	$F = A+1$	$A = 2^n - 1$	$C_{out}=1$ and $F=0$ if $A = 2^n - 1$
c)	0	1	0	$F = A+B$	$(A+B) \geq 2^n$	Overflow occurs if $C_{out} = 1$
d)	0	1	1	$F = A+B+1$	$(A+B) \geq (2^n - 1)$	Overflow occurs if $C_{out} = 1$
e)	1	0	0	$F = A-B-1$	$A > B$	If $C_{out} = 0$, then $A \leq B$
f)	1	0	1	$F = A-B$	$A \geq B$	If $C_{out} = 0$, then $A < B$
g)	1	1	0	$F = A-1$	$A \neq 0$	$C_{out} = 1$ except when $A = 0$
h)	1	1	1	$F = A$		C_{out} is always 1

Q. Describe the effect of Output Carry.

⇒ Table में देखकर with an example each.

* $A + B + C_{in}$

⇒ $A + 0 + 1$

$$\begin{array}{r} 1\ 1\ 1\ 0 \\ 0\ 0\ 0\ 0 \\ \qquad\quad 1 \\ \hline (1)\ 0\ 0\ 0\ 0 \\ 0\ 1\ 1\ 1\ 1 \end{array}$$

x = 1 1 1 1

y = 0 0 0 0

z = 1

---

(1) 0 0 0 0

* সর্বোচ্চ bit এর ক্ষেত্রেই কেবল Cout = 1 হবে।

* x = 1 0 1 0

y = 0 1 0 1

z = 0

---

(0) 1 1 1 1

(*) · A + B' + 0

x = 1 0 1 0 (10) | A > B

y = 1 0 0 0 (-7) | So, Cout = (1)

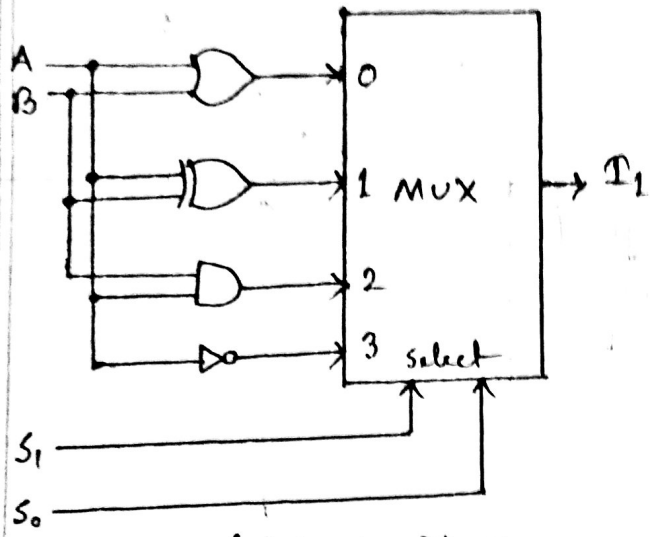
z = 0

---

(1) 0 0 1 0

06 May 2017

Design of Logic Circuit



(a) Logic Diagram

S_1	S_0	Output	Operation
0	0	$F_i = A_i + B_i$	OR
0	1	$F_i = A_i \oplus B_i$	XOR
1	0	$F_i = A_i B_i$	AND
1	1	$F = A_i'$	NOT

(b) Function Table

fig 9-11: One stage of logic circuit

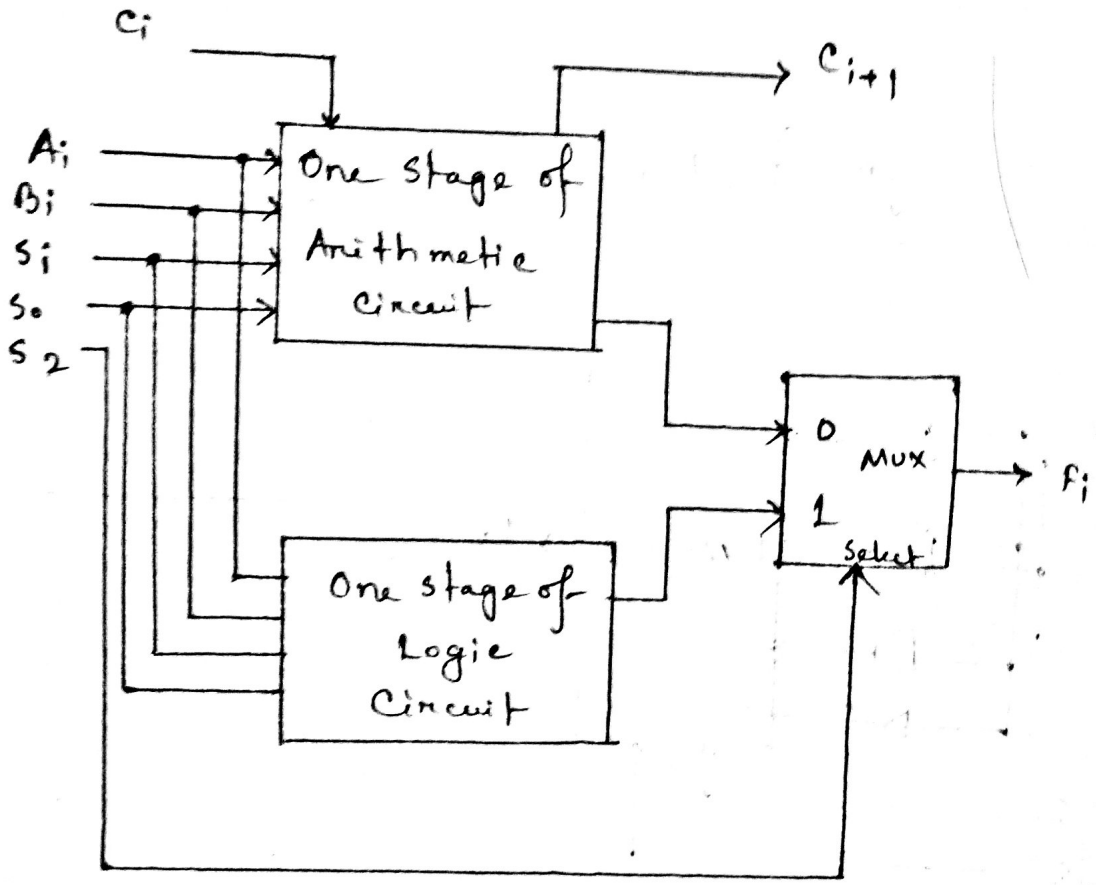


Fig 9-12 : Combining logic and arithmetic circuits

Table ⇒ 9-3: Logic operations in one stage of arithmetic circuits

S_2 S_1 S_0	x_i y_i C_i	$F_i = x_i \oplus y_i$	Operation	Required Operation
1 0 0	A_i 0 0	$F_i = A_i$	Transfer A	OR
1 0 1	A_i B_i 0	$F_i = A_i \oplus B_i$	XOR	XOR
1 1 0	A_i B_i' 0	$F_i = A_i \odot B_i$	Equivalence	AND
1 1 1	A_i 1 0	$F_i = A_i'$	NOT	NOT

* Design of Arithmetic Logic Unit (ALU)

The steps involved in the design of an ALU are as follows:

(1) Design the arithmetic section independent of the logic section.

(2) Determine the logic operations obtained from the arithmetic circuit in step 1, assuming that the input carries to all stages are 0.

(3) Modify the arithmetic circuit to obtain the required logic operations.

Function Table of the ALU

Selection S_2 S_1 S_0 C_{in}	Output	x_i	y_i	Function
0 0 0 0	$F = A$	A	0	Transfer A
0 0 0 1	$F = A+1$	A	0	Increment A
0 0 1 0	$F = A+B$	A	B	Addition
0 0 1 1	$F = A+B+1$	A	B	Add with carry
0 1 0 0	$F = A-B-1$	A	B'	Subtract with borrow
0 1 0 1	$F = A-B$	A	B'	Subtraction
0 1 1 0	$F = A-1$	A	1	Decrement A
0 1 1 1	$F = A$	A	1	Transfer A
1 0 0 X	$F = A \vee B$	A+B	0	OR
1 0 1 X	$F = A \oplus B$	A	B	XOR
1 1 0 X	$F = A \wedge B$	A+B'	B'	AND
1 1 1 X	$F = A'$	A	1	Complement A

(*) Logic always bit-by-bit করে করে, Cin এর কোন কাজ নেই।

OR

$$\begin{aligned} F &= x \oplus y \\ &= (A + B') \oplus B' \\ &= (A + B')' \cdot B' + (A + B') \cdot B \\ &= A' \cdot B \cdot B' + AB + B' \cdot B \\ &= AB \end{aligned}$$

(*) "x" এর সকল logic operation দিয়ে দিবা Lab এর method এর মত।

Proper Way: (Result) [Ultimate]

$$X_i = A_i + S_2 S_1' S_0' B_i + S_2 S_1 S_0' B_i'$$

$$Y_i = S_0 B_i + S_1 B_i'$$

$$Z_i = S_2' C_{in}$$

Problems

9-8, 10, 11, 12, 13, 16, 17 [Assignment]

* Amp

07 May 2017

Problem 9-17

$$x_i = A_i B_i + (S_2 S_1' S_0')' A_i + S_2 S_1 S_0' B_i$$

$$y_i = S_0 B_i + S_1 B_i' (S_2 S_i S_0')'$$

$$Z = S_2' C$$

Soln: if $S_2 = 0$ then function reduce to →

$$X = AB + A = A(B+1) = A \quad [\because B+1 = 1]$$

$$Y = S_0 B + S_1 B'$$

$$Z = C_i$$

---

when $(S_1 S_0) = (00)$,

$X = A$

$Y = 0$

now, if $C_{in} = 0$ | if $C_{in} = 1$

$F = A$ | $F = A+1$

when, $(S_1 S_0) = (01)$;

$X = A$

$Y = B$

now if $C_{in} = 0$ | if $C_{in} = 1$

$F = A+B$ | $F = A+B+1$

---

when, $(S_1 S_0) = (10)$;

$X = A$

$Y = B'$

now, if $C_{in} = 0$ | if $C_{in} = 1$

$F = A + B'$ | $F = A + B' + 1$

when $(S_1, S_0) = (1\ 1)$,

$X = A$

$y = B + B' = 1$ (All 1)

now if $C_{in} = 0$ | if $C_{in} = 1$

$F = A - 1$ | $F = A$

---

when $S_2 = 1$, $x \oplus y$ operation will occur in each stage.

Now if $(S_2\ S_1\ S_0) = (1\ 0\ 0)$

$x = AB + (1.1.1)'A + 1.0.0'.B$

$= AB$

$y = 0.B + 0.B'(1.0.0')' = 0$

$= AB \oplus 0 = AB = A \cap B$

if $(S_2\ S_1\ S_0) = (1\ 0\ 1)$,

$$x = AB + (1.1.0)'A + 1.0.0.B$$

$$= AB + A = A(B+1) = A$$

$$y = 1.B + 0.B'(1.0.0)' = B$$

$$F = A \oplus B$$

if $(S_2\ S_1\ S_0) = (1\ 1\ 0)$

$$x = AB + (1.0.1)'A + 1.1.1.B$$

$$= AB + A + B$$

$$= A + B$$

$$y = 0.B + 1.B'(1.1.1)' = 0$$

$$F = (A+B) \oplus 0$$

$$= A+B = A \cup B$$

if $(S_2\ S_1\ S_0) = (1\ 1\ 1)$

$$x = AB + (1.0.0)'A + 1.1.D.B$$

$$= AB + A = A$$

$$y = 1.B + 1.B'(1.1.0)'$$

$$= B + B' = 1$$

$\therefore F = A \oplus 1$

$= A \cdot 1' + A' \cdot 1$

$= A'$

So, the 12 function of the ALU ⇒

	Selection						
	S_2	S_1	S_0	C_{in}	X	Y	Output F
1	0	0	0	0	A	0	A
2	0	0	0	1	A	0	A+1
3	0	0	1	0	A	B	A+B
4	0	0	1	1	A	B	A+B+1
5	0	1	0	0	A	B'	A+B'
6	0	1	0	1	A	B'	A+B'+1
7	0	1	1	0	A	All 1	A-1
8	0	1	1	1	A	All 1	A
9	1	0	0	X	A·B	0	$A \wedge B$
10	1	0	0	X	A	B	$A \oplus B$
11	1	1	0	X	A+B	0	$A \cup B$
12	1	1	1	X	A	1	A'

Quiz-1 [14.05.17] (Sunday)

Up to chapter-9

বড় Question -টি আসা হবে না।

11 May 2017

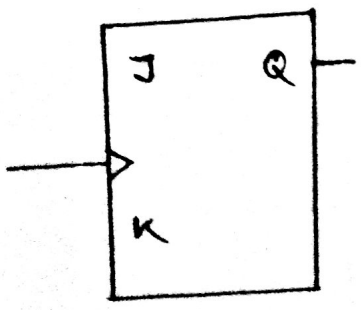
J-K flip flop

T.T (Truth Table)

0	0	#id	0 1	0 1
0	1	Reset	0 1	0 0
1	0	Set	0 1	1 1
1	1	Toggle	0 1	1 0

Excitation Table

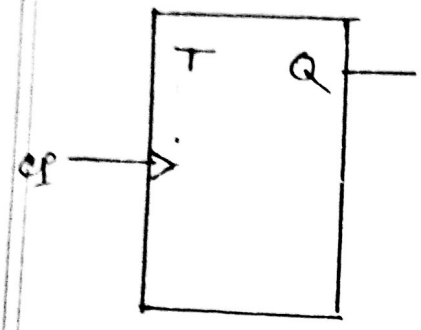
P.S	N.S	J	K
0	0	0	x
0	1	1	x
1	0	x	1
1	1	x	0



T- flip flop

0 → Hold

1 → Toggle

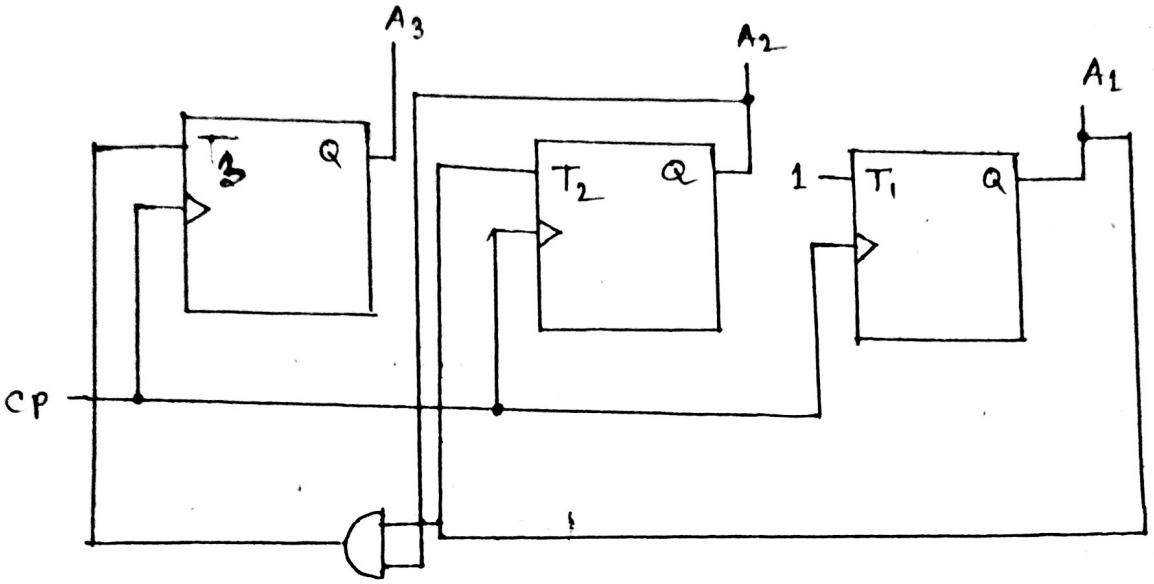


Excitation Table

P.S	N.S	T
0	0	0
0	1	1
1	0	1
1	1	0

3-bit binary Counter

R.S			N.S		
A_3	A_2	A_1	A_3	A_2	A_1
0	0	0	0	0	1
0	0	(1)	0	(1)	0
0	1	0	0	1	1
0	1	(1)	(1)	(0)	0
1	0	0	1	0	1
1	0	(1)	1	(1)	0
1	1	0	1	1	1
1	1	(1)	(0)	(0)	0



$T_1 = 1$

$T_2 = A_1$

$T_3 = A_2 A_1$

Q. Design a 4-bit / 5-bit counter or specify করে flip flop (নির্দিষ্ট) দিয়ে করতে বলতে পারে।

* Booth ~ "D-flip flop" কেমন use করা হয়।

with JK flip flop:

P.S			N.S								
A_3	A_2	A_1	A_3	A_2	A_1	J_3	K_3	J_2	K_2	J_1	K_1
0	0	0	0	0	1	0	X				
0	0	1	0	1	0	0	X				
0	1	0	0	1	1	0	X				
0	1	1	1	0	0	1	X				
1	0	0	1	0	1	X	0				
1	0	1	1	1	0	X	0				
1	1	0	1	1	1	X	0				
1	1	1	0	0	0	X	1				

(1+6)

13 May 2017

Booth's Multiplication Algorithm

Booth's algorithm can be implemented by repeatedly adding one of two predetermined values A and S to a product P, then performing a rightward arithmetic shift on P.

Let m and n be the multiplicand and multiplier, respectively and let x and y represent the number of bits in m and n.

1 Determine the value of A and S; and the initial value of P. All of these numbers should have a length equal to $(x+y+1)$.

① A: Fill the most significant (left most) bits with the value of m. Fill the remaining $(y+1)$ bits with zeros.

(ii) S: Fill the most significant bits with the value of $(-m)$ in two's complement notation. Fill the remaining $(y+1)$ bits with zeros.

(iii) P: Fill the most significant x bits with zeros. To the right of this, append the value of n. Fill the least significant (right most) bit with a zero.

[2] Determine the two least significant bits of P.

(i) If they are 01, find the value of P+A. Ignore any overflow.

(ii) If they are 10, find the value of P+S. Ignore any overflow.

(iii) If they are 00, do nothing, use P directly in next step.

(iv) If they are 11, do nothing, use P directly in next step.

[3] Arithmetically shift the value obtained in the 2nd step by a single place to the right. Let p now equal this new value.

[4] Repeat steps 2 and 3 until they have been done y times.

[5] Drop the least significant (right most) bit from P. This is the product of m and n.

* Find $13 \times (-6)$ using Booth's algorithm wi

$m = 13$ and $n = -6$, and $x = 5$, $y = 5$

Sol^n:

$m = 01101$ $\quad -m = 10011$

$$\begin{array}{r} 01101 \\ 11001 \\ \hline n = 11010 \end{array}$$

A:	01101	00000 0
S:	10011	00000 0
P:	00000	11010 0

Step 1

P: 00000 11010 0

→ P: 00000 01101 0 ; [Right shift]

Step 2:

P: 00000 01101 0

P+S: 10011 01101 0

→ P: 11001 10110 1

Step 3:

P+A : 00110 10110 1

→ P : 00011 01011 0

Step 4:

P+S : 00011 01011 0

10011 00000 0

10110 01011 0

→ P : 11011 00101 1

Step 5:

→ P : 11101 10010 1

Ans. 11101 10010

-78

* xm ↳ multiplication ↳ ↳ math must ↳

Booth's algorithm ↳ procedure ↳ ↳

* Design করার সময় যেকোন example নিয়ে design করা যায় algorithm should be followed.

14 May 2017

0 0	→ 0
0 1	→ +y
1 0	→ -y
1 1	→ 0

5X5 bit Booth Multiplier

X	Y	-Y
01101 (13)	11010 (-6)	00110

	U	V	X	X_{-1}	
	00000	00000	01101	0	1st step
	00110				
	00110	00000	01101	0	
(Right shift)	00011	00000	00110	1	
(Right shift)	11010				
	11101	00000	00110	1	2nd step
(R.S.)	11110	10000	00011	0	
(R.S.)	00110				
	00100	10000	00011	0	3rd step
(R.S)	00010	01000	00001	1	
(R.S)	00000				
	00010	01000	00001	1	4th step
(R.S)	00001	00100	00000	1	
(R.S)	11010				
	11011	00100	00000	1	5th step
(R.S)	11101	10010	00000	0	

* xm ও এইটা করলে না।

1 Initialization :

U ← 0

V ← 0

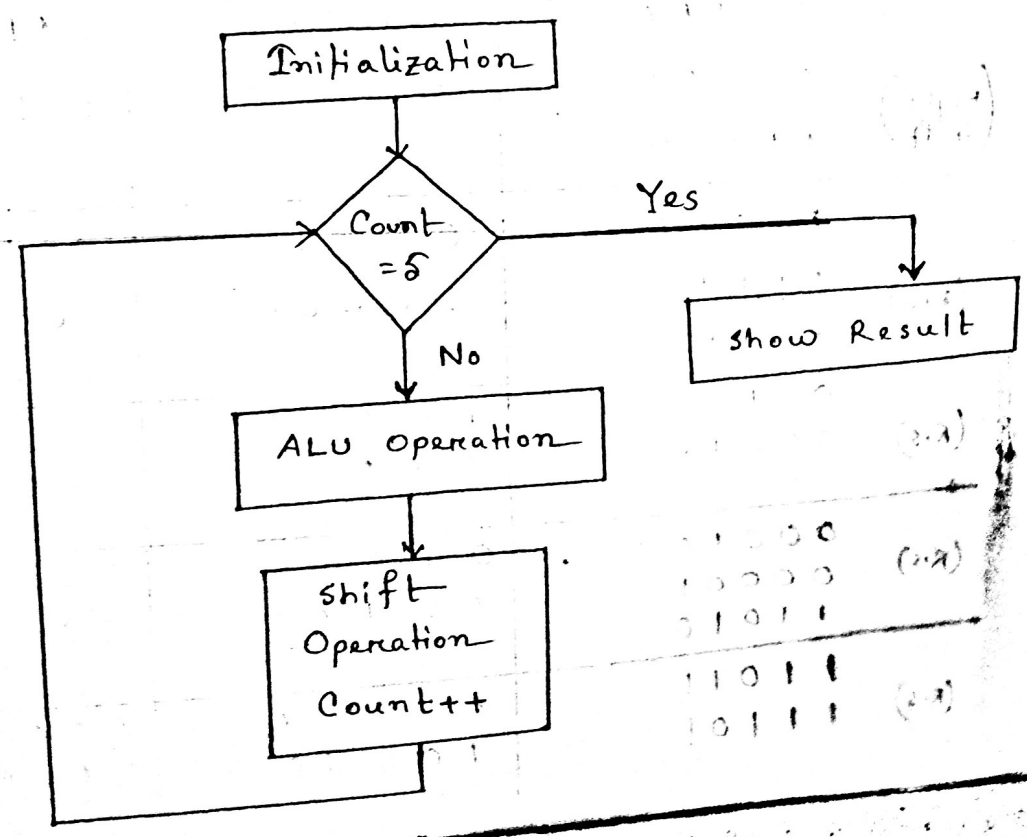
X ← Input

Y ← Input

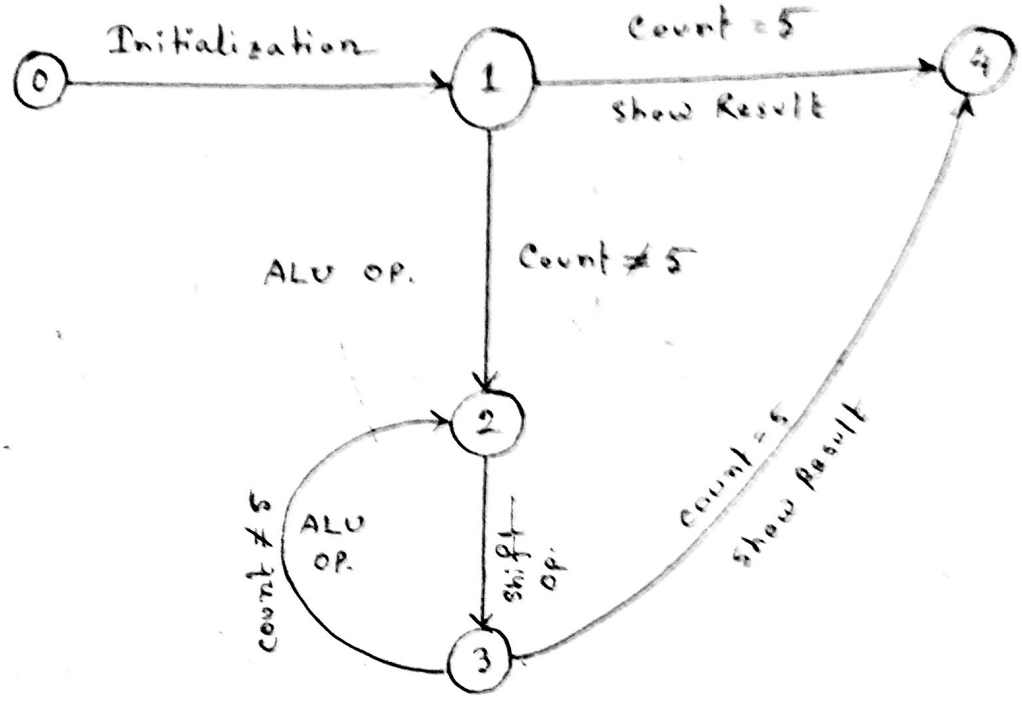
X_{-1} ← 0

Count ← 0

2 Flow Chart :



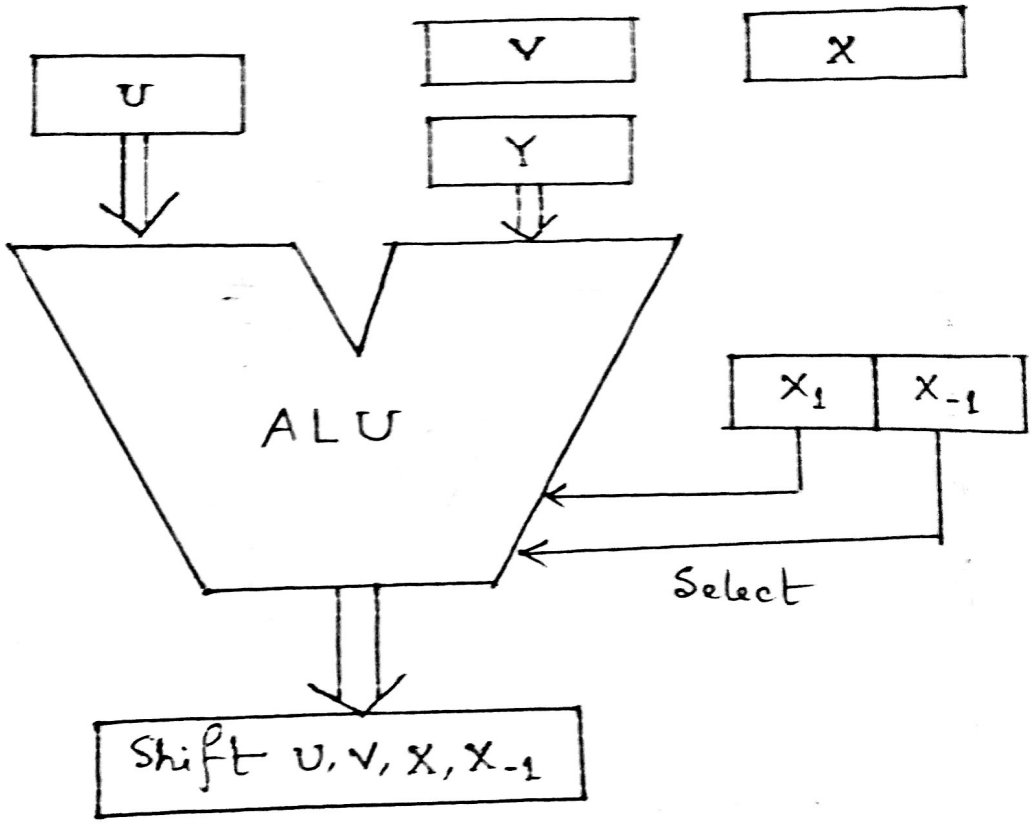
[3] State Diagram :



ALU Op.

S_1	S_0	Operation
0	0	U + 0
0	1	U + Y
1	0	U - Y
1	1	U + 0

4 Architecture :



18 May 2017

10-3

Hardwired Control

Example – 1

The design is carried out in 5 consecutive steps:

[1] The problem is stated.

[2] An initial equipment configuration is assumed.

[3] An algorithm is formulated.

[4] The data processor part is specified

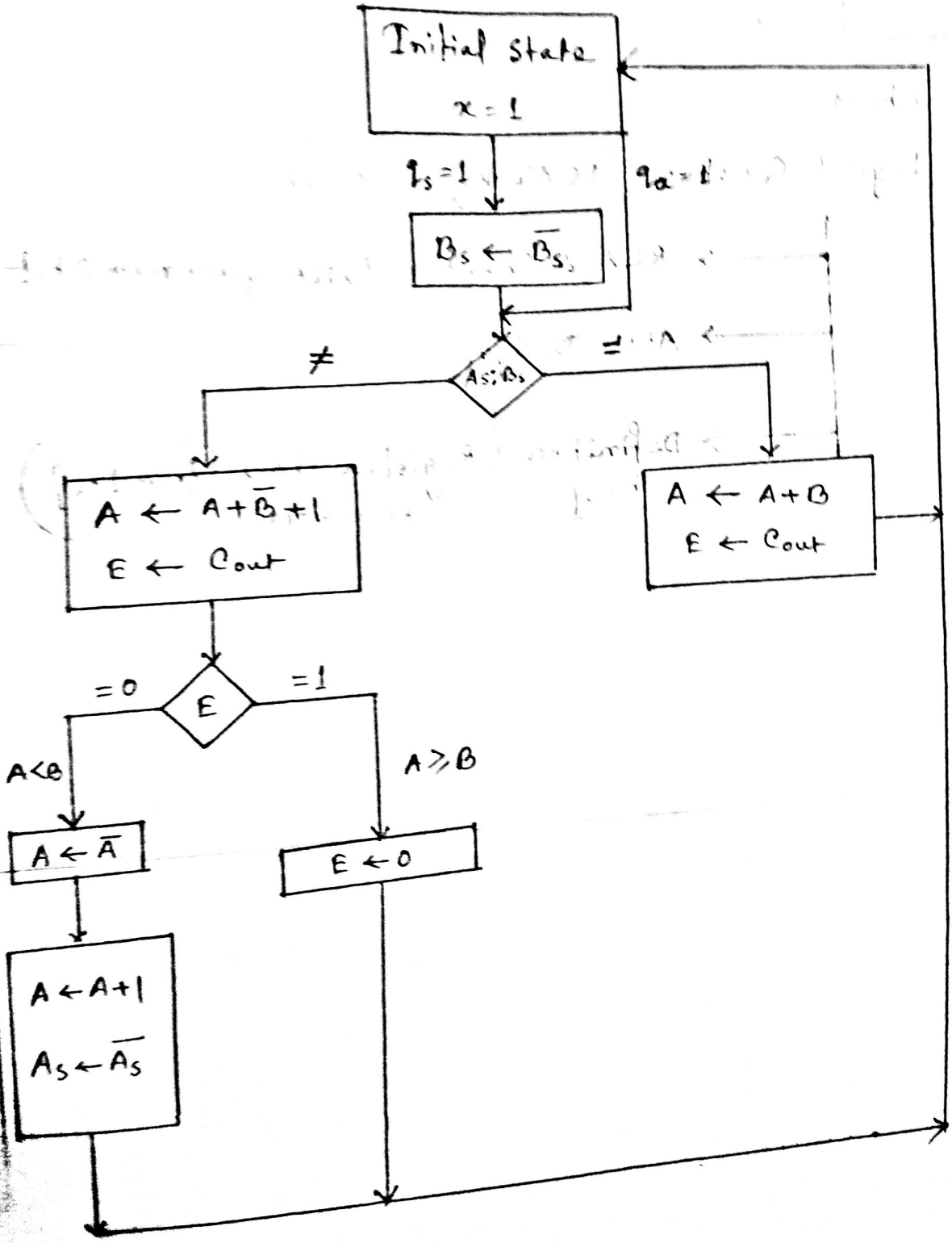
[5] The control logic is designed.

[1] Statement of the problem

The problem here is to implement with hardware the addition and subtraction of two fixed-point binary numbers represented in sign-magnitude form.

sign bit
(0) 0101 → +5
(1) 0101 → −5

Flow Chart


Initial State
x = 1
$q_s = 1$
$q_a = 1$
$B_s \leftarrow \overline{B_s}$
$A_s : B_s$
≠
=
$A \leftarrow A + \overline{B} + 1$
$E \leftarrow C_{out}$
$A \leftarrow A + B$
$E \leftarrow C_{out}$
E
=0
=1
A<B
A≥B
$A \leftarrow \overline{A}$
$E \leftarrow 0$
$A \leftarrow A + 1$
$A_s \leftarrow \overline{A_s}$

[2] Equipment Configuration

As = Result

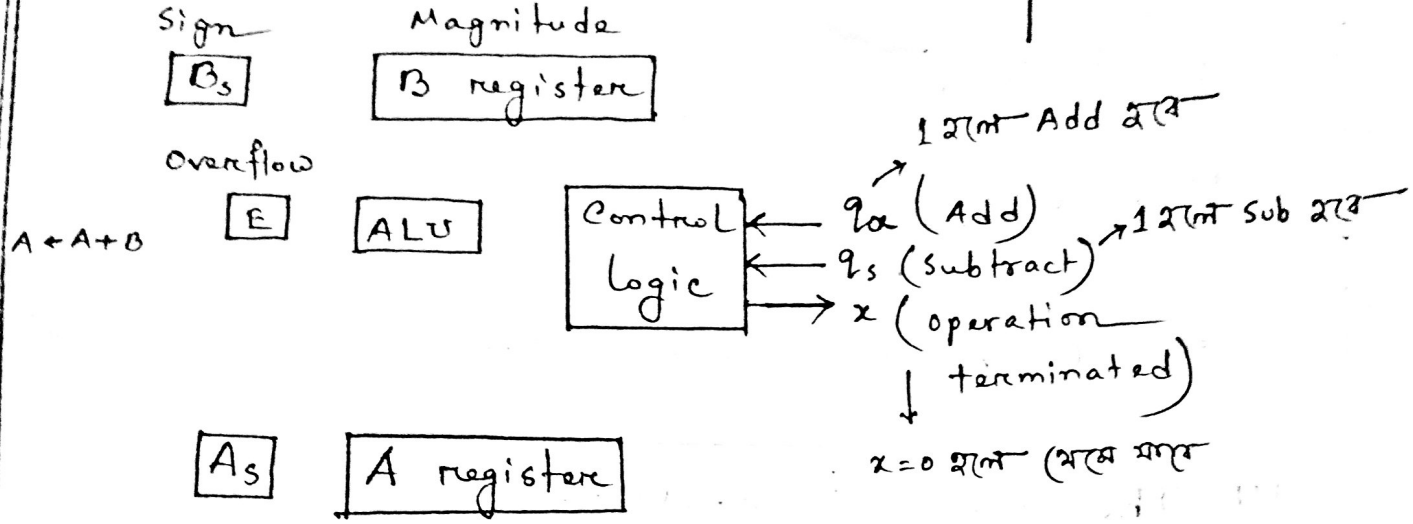


Fig 10-6 : Register Configuration for the adder–subtractor

[3] Derivation of the Algorithm:

$$(\pm A) \pm (\pm B)$$

* B এর sign change হবে

$$(\pm A) - (+B) = (\pm A) + (-B)$$

$$(\pm A) - (-B) = (\pm A) + (+B)$$

	if $A \geq B$	if $A < B$
$(+A) + (+B)$	$= +(A+B)$	
$(+A) + (-B)$	$= +(A-B)$	$= -(B-A)$
$(-A) + (+B)$	$= -(A-B)$	$= +(B-A)$
$(-A) + (-B)$	$= -(A+B)$	

same হবে, sign আলাদা

* Example (গত Question এসেছে)

Sunday - Quiz-01

Ch-9

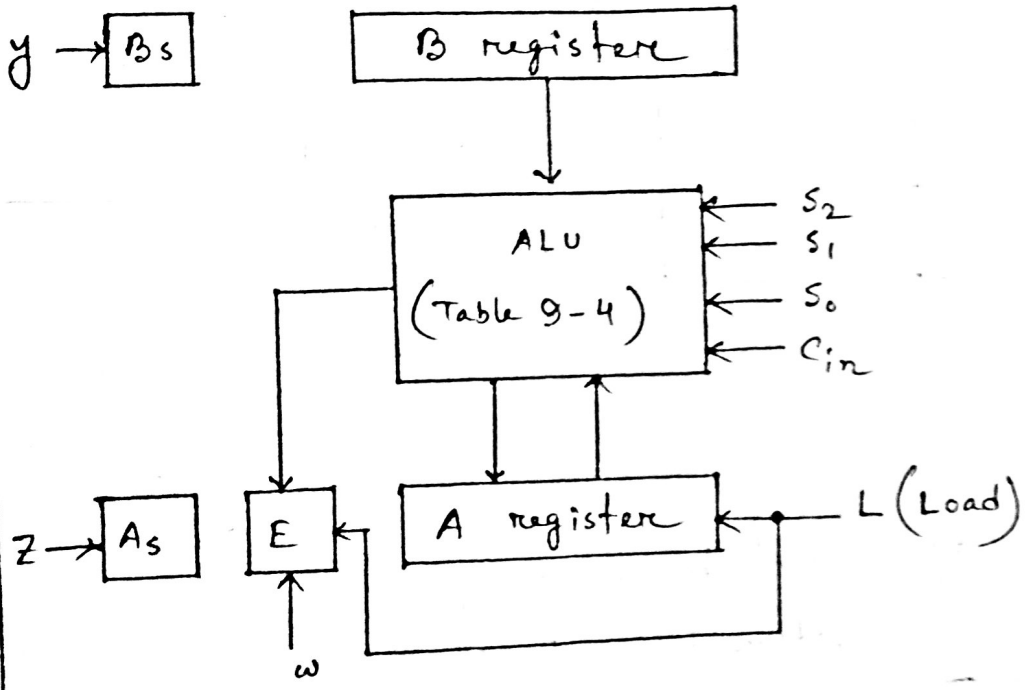
Logical Question (15 Marks) — Math

- PLA & Chapter-9, PLA square num 2 bit
- ALU - Design
- Definition (Register কি? [from Lec] Theory RAM, ROM

25 May 2017

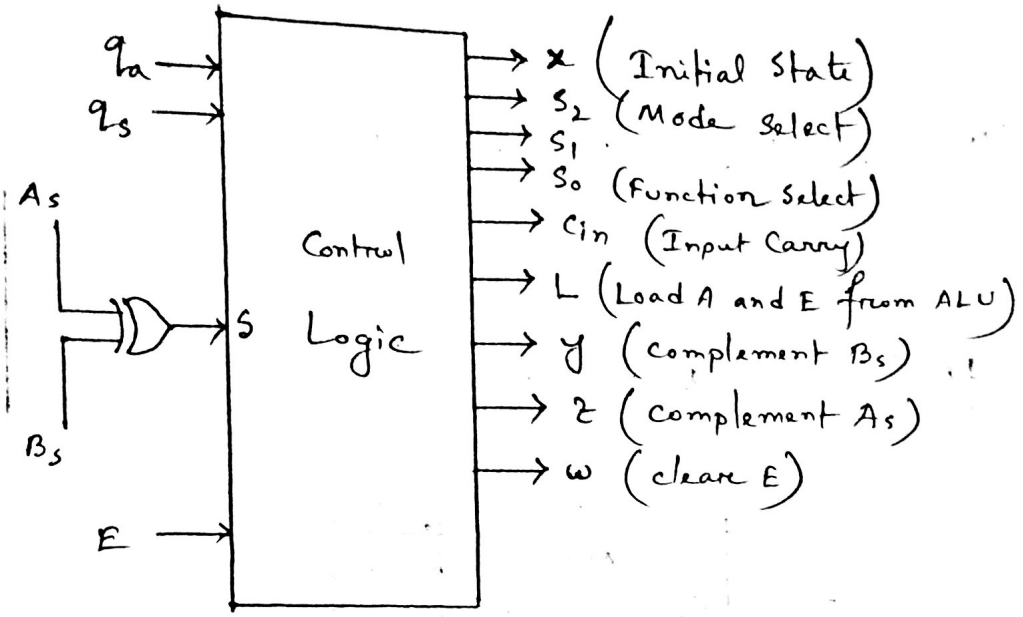
[4] The data-processore part:

A, B => Magnitude



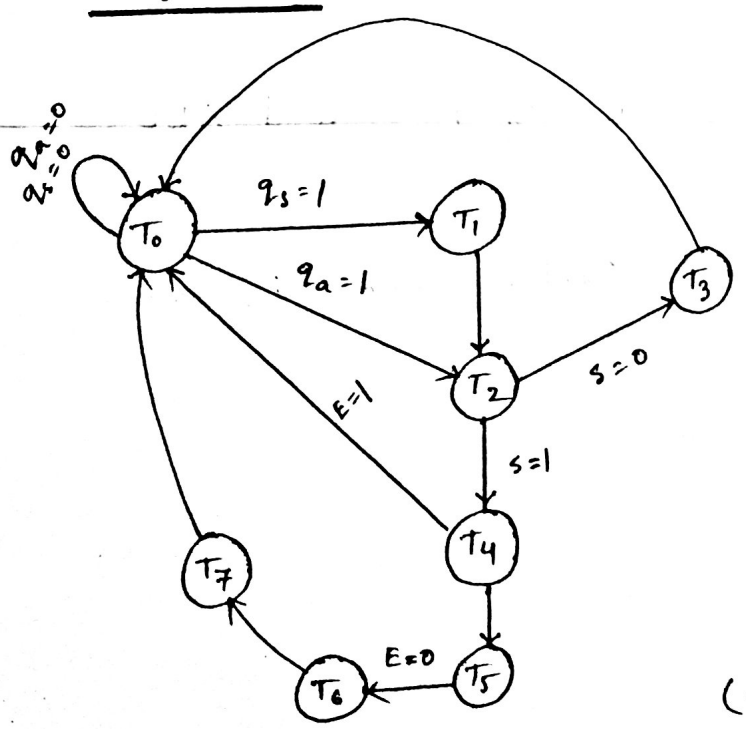
(a) Data processore registers and ALU

Fig 10-8: System block Diagram



(b) Control block diagram

Fig: 10-8 system block diagram



(a) State Diagram

Control outputs

	x	S_2	S_1	S_0	C_{in}	L	y	z	w
T_0 : Initial stage $x=1$	1	0	0	0	0	0	0	0	0
T_1 : $B_s \leftarrow \overline{B_s}$	0	0	0	0	0	0	1	0	0
T_2 : nothing	0	0	0	0	0	0	0	0	0
T_3 : $A \leftarrow A+B$, $E \leftarrow C_{out}$	0	0	0	1	0	1	0	0	0
T_4 : $A \leftarrow A+\overline{B}+1$, $E \leftarrow C_{out}$	0	0	1	0	1	1	0	0	0
T_5 : $E \leftarrow 0$	0	0	0	0	0	0	0	0	1
T_6 : $A \leftarrow \overline{A}$	0	1	1	1	0	1	0	0	0
T_7 : $A \leftarrow A+1$, $A_s \leftarrow \overline{A_s}$	0	0	0	0	1	1	0	1	0

(b) Sequence of register transfers;

Fig : 10-9 : Control (state) Logic Diagram and Sequence of micro-operations

27 May 2017

5 Design of Hard-wired Control (Control Logic Design)

The control is designed by one flip-flop per state method

Table 10-1 : Boolean function for Control logic

Flip-flop input function	Boolean function for output control
$DT_0 = q_s' \cdot q_a' \cdot T_0 + T_3 + ET_5 + T_7$	$x = T_0$
$DT_1 = q_s T_0$	$s_2 = T_6$
$DT_2 = q_a T_0 + T_1$	$s_1 = T_4 + T_6$
$DT_3 = s' T_2$	$s_0 = T_3 + T_6$
$DT_4 = s T_2$	$C_{in} = T_4 + T_7$
$DT_5 = T_4$	$L = T_3 + T_4 + T_6 + T_7$
$DT_6 = E' T_5$	$y = T_1$
$DT_7 = T_6$	$z = T_7$
	$w = T_5$

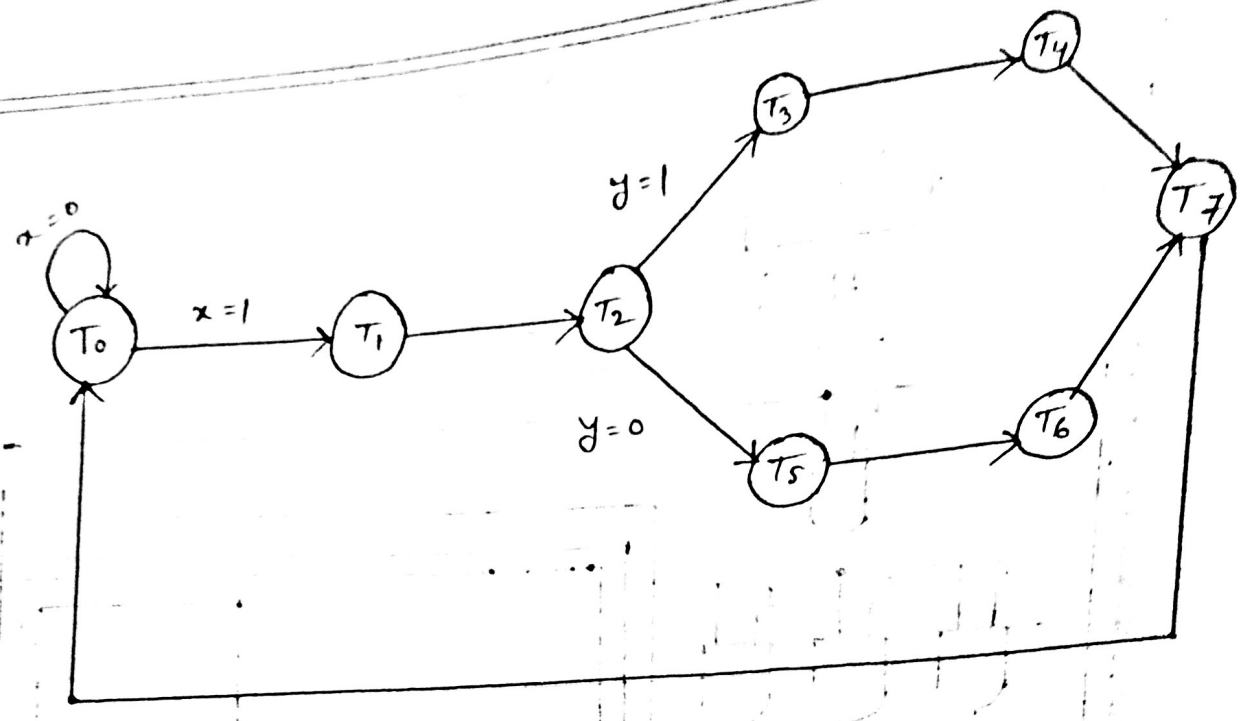


Fig : P10-27

Q. Design the control using eight D-flip flops. (ডিজাইন করতে)

=> $DT_0 = x'T_0 + T_7$

$DT_1 = xT_0$

$DT_2 = T_1$

$DT_3 = yT_2$

$DT_4 = T_3$

$DT_5 = y'T_2$

$DT_6 = T_5$

$DT_7 = T_4 + T_6$

(fig - 10.28, পরের পাতায়)

28 May 2017

10-4 Microprogram Control

Hardware Configuration

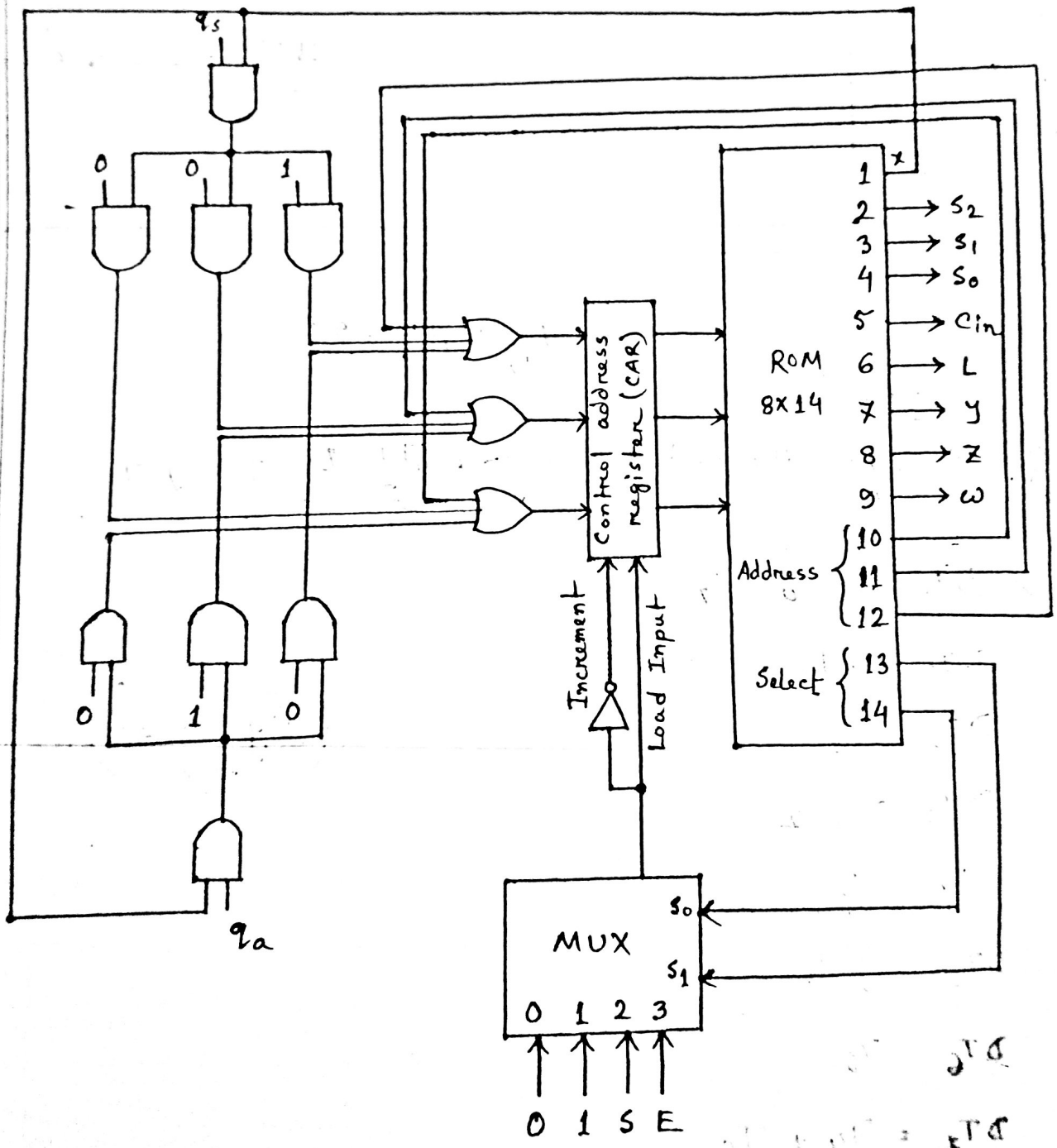


Fig = (10-10) Microprogram control block diagram

CAR = Control Address Register

ROM bits 13 14	MUX select function
0 0	Increment CAR
0 1	Load Input to CAR
1 0	Load Input to CAR if S=1, increment CAR if S=0
1 1	Load Input to CAR if E=1, increment CAR if E=0

(with figure)

The Microprogram

Table: 10-2 Symbolic Microprogram for control memory

ROM address	Micro-instruction	Comments
0	x=1, if (q_s=1) then (go to 1), if (q_a=1) then (go to 2), if ($q_a \wedge q_s$)=0 then (go to 0)	Load 0 or external address
1	$B_s \leftarrow \bar{B}_s$	q_s=1, start subtraction
2	If (s=1) then (go to 4)	q_a=1, start addition
3	$A \leftarrow A+B$, $E \leftarrow C_{out}$, go to 0	Add magnitudes and return
4	$A \leftarrow A+\bar{B}+1$, $E \leftarrow C_{out}$	subtract magnitudes
5	If (E=1) then (go to 0), $E \leftarrow 0$	Operation terminated if E=1
6	$A \leftarrow \bar{A}$	E=0, complement A
7	$A \leftarrow A+1$, $A_s \leftarrow \bar{A}_s$, go to 0	Done, return to address 0

Table : 10-3 Binary Microprogram for Control Memory

ROM Address	x S_2 S_1 S_0 C_{in} L y z w	Address	Select
	1 2 3 4 5 6 7 8 9	10 11 12	13 14
000	1 0 0 0 0 0 0 0 0	0 0 0	0 1
001	0 0 0 0 0 0 1 0 0	0 1 0	0 1
010	0 0 0 0 0 0 0 0 0	1 0 0	1 0
011	0 0 0 1 0 1 0 0 0	0 0 0	0 1
100	0 0 1 0 1 1 0 0 0	1 0 1	0 1
101	0 0 0 0 0 0 0 0 1	0 0 0	1 1
110	0 1 1 1 0 1 0 0 0	1 1 1	0 1
111	0 0 0 0 1 1 0 1 0	0 0 0	0 1

* xm ↪ Table : 10-2 থেকে ALU এর circuit selection নির্দেশ করবে, Table - 10-3 কাকে বলে ? [7 Marks]

01 June 2017

10-6 : Hard-Wired Control

Example – 2

1 Statement of the Problem

We wish to design an arithmetic circuit that multiplies two fixed point binary numbers in sign-magnitude representation

multiplicand : 10111

multiplier : 10011

1st multiplier bit = 1, copy multiplicand 10111

shift right to obtain 1st partial product (pp) 01.0111

2nd multiplier bit = 1, copy multiplicand 10111

---

add multiplicand to previous pp 1000101

shift right to obtain 2nd pp 0100010 1

3rd multiplier bit = 0, shift right to obtain 3rd pp 0010001 01

4th " " = 0, " " " " 4th pp 0001000 101

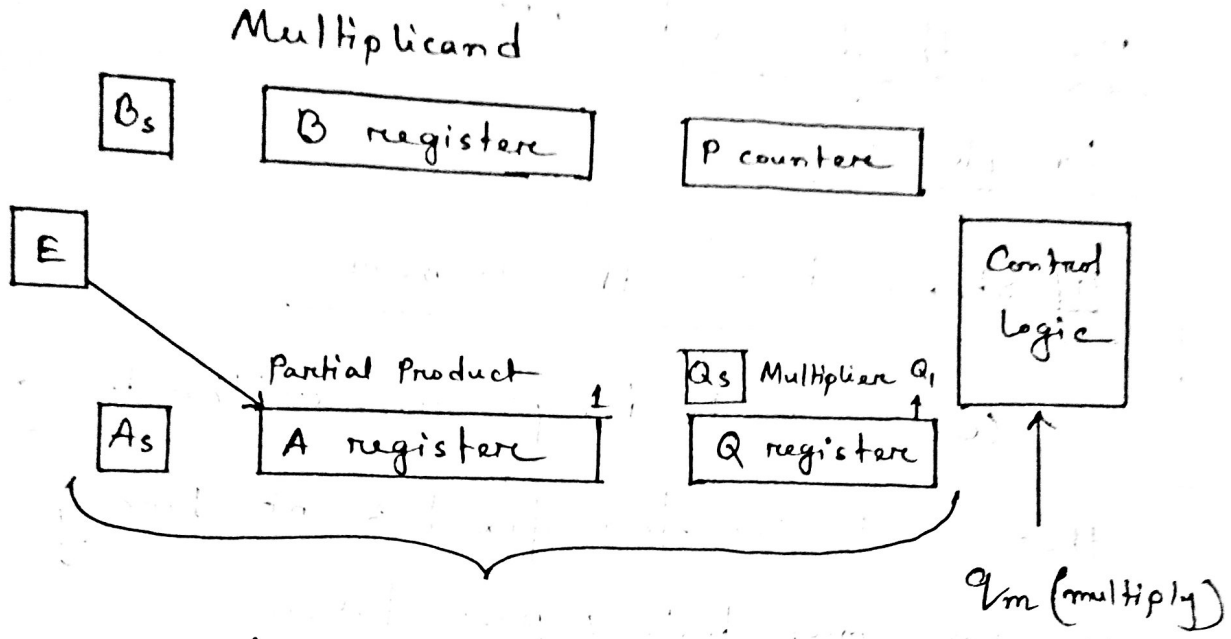
5th " " = 1, copy multiplicand 10111

---

add multiplicand to previous pp 11011 0101

shift right to obtain 5th pp = final product 01101 1010 1

[2] Equipment Configuration :



* E এর value SHIFT ও চলে আসবে!
* E এর value A এর msB তে আসবে।

[3] Derivation of Algorithm : (বই থেকে দেখবে)

B ← Multiplicand magnitude, Bs ← sign

Q ← Multiplier magnitude, Qs ← sign

A ← A+B (Multiplicand + Partial Product)

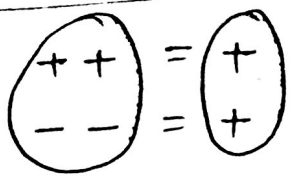
E ← Cout

P ← k, k = number of bits in multiplier

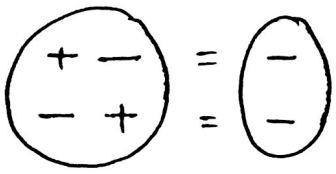
Shift Operation :

AQ ← shr EAQ, E ← 0

[Details → Book]

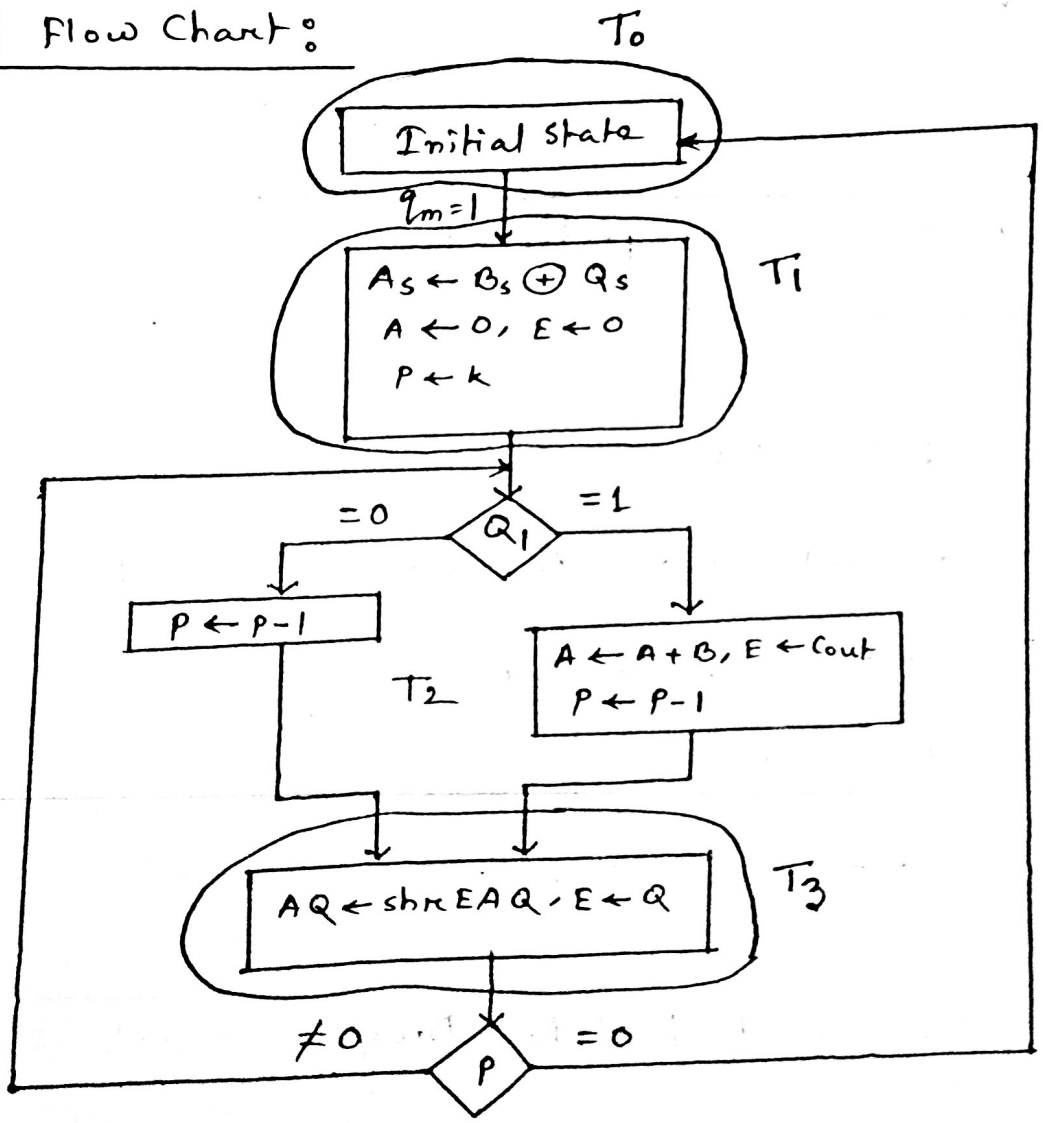


0



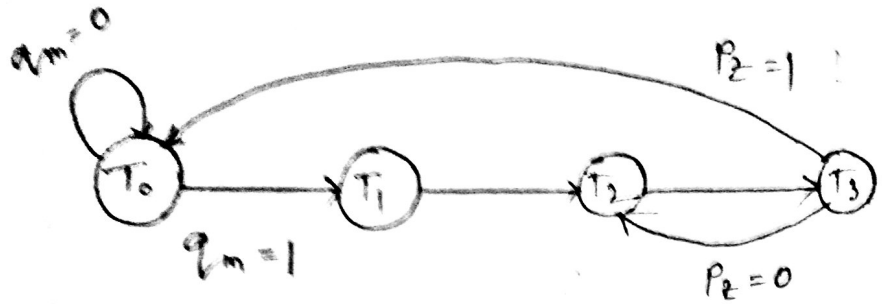
1

[4] Flow Chart:



03 June 2017

(a) State Diagram



(b) Sequence of register transfer

T_0 : Initial state

T_1 : $A_S \leftarrow B_S \oplus Q_S$, $A \leftarrow 0$, $E \leftarrow 0$, $P \leftarrow k$

$Q_1 T_2$: $A \leftarrow A + B$, $E \leftarrow C_{out}$

T_2 : $P \leftarrow P - 1$

T_3 : $AQ \leftarrow shr\,EAQ$, $E \leftarrow 0$

fig: 10-15 : Control state Diagram and sequence of micro-operations for multiplier.

Fig: 10-16

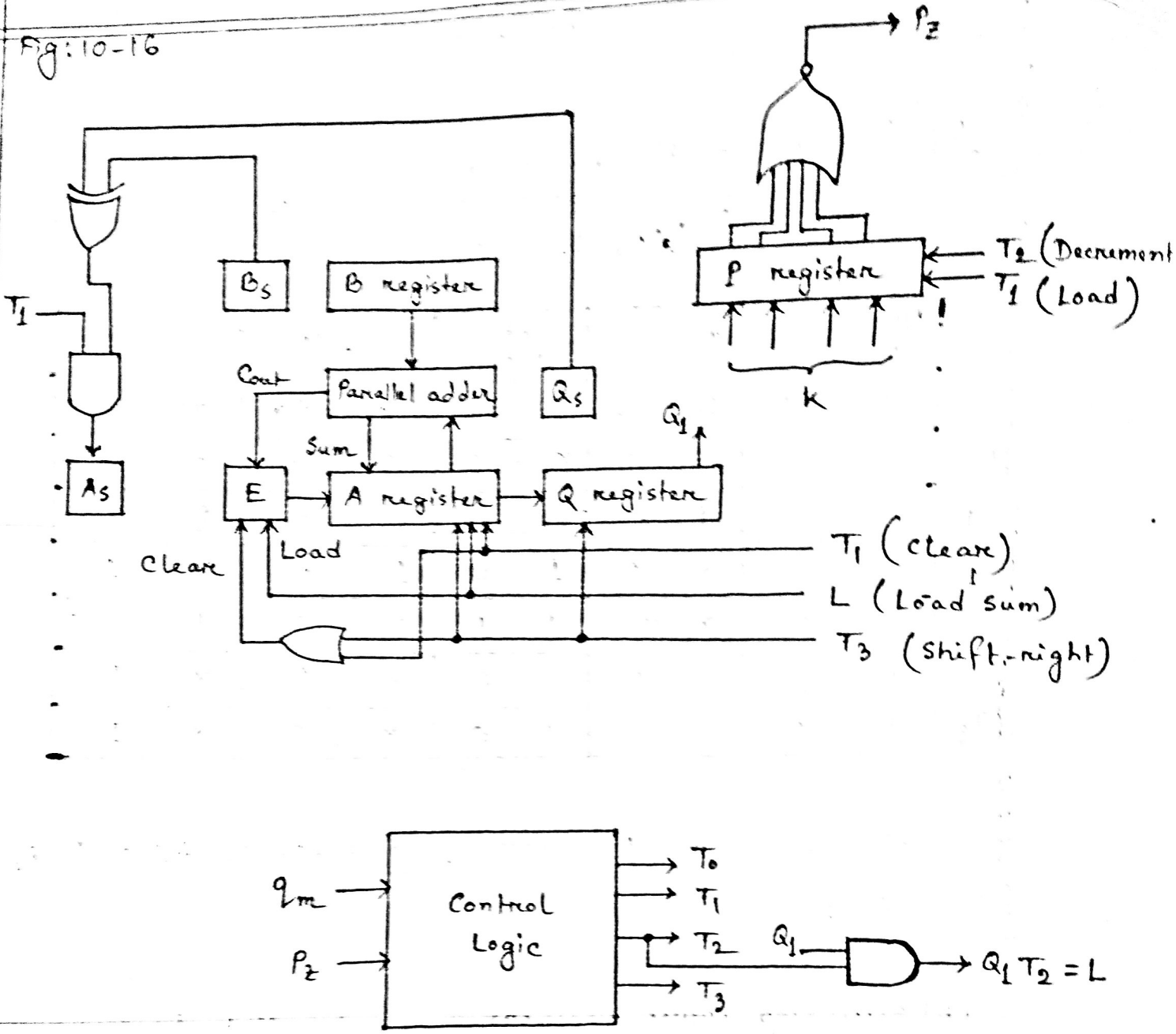


Figure: 10-16 Data processor for binary multiplier

04 Jun 2017

5 Design of Hard-wired Control

(a) Excitation Table

	Present state		Inputs		Next state		Flip-flop inputs			
	G_2	G_1	q_m	P_2	G_2	G_1	JG_2	KG_2	JG_1	KG_1
T_0	0	0	0	X	0	0	0	X	0	X
T_0	0	0	1	X	0	1	0	X	1	X
T_1	0	1	X	X	1	0	1	X	X	1
T_2	1	0	X	X	1	1	X	0	1	X
T_3	1	1	X	1	0	0	X	1	X	1
T_3	1	1	X	0	1	0	X	0	X	1

(2টি flip flop লাগবে, কারণ 3 কে represent করতে 2 bit লাগে)

(b) Flip-flop input functions

State P	State N	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

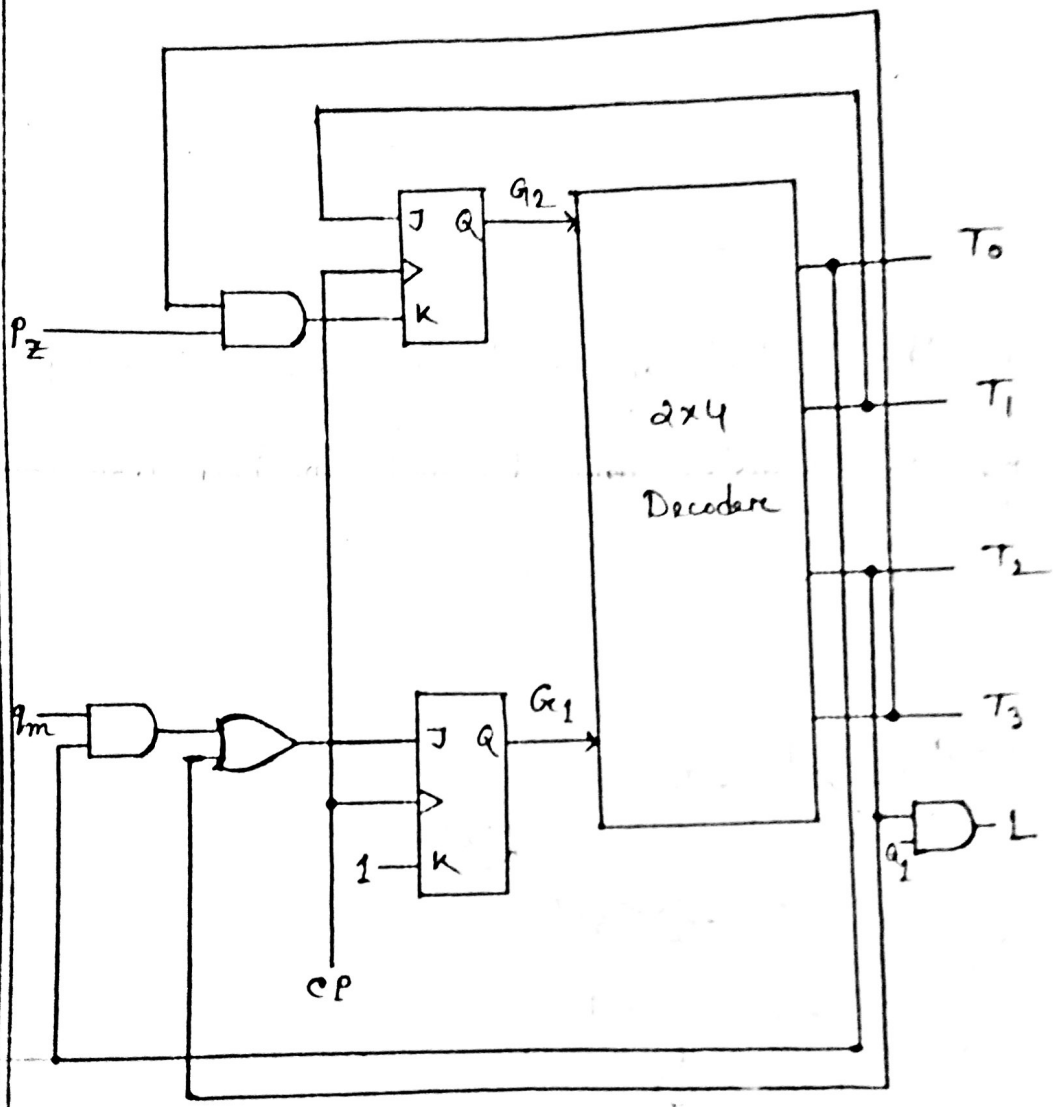
$$JG_2 = T_1$$

$$KG_2 = T_3 P_2$$

$$JG_1 = T_0 q_m + T_2$$

$$KG_1 = 1$$

(c) Logic Diagram (Book) [10-17 (fig)]



$P_z \rightarrow$ P register zero

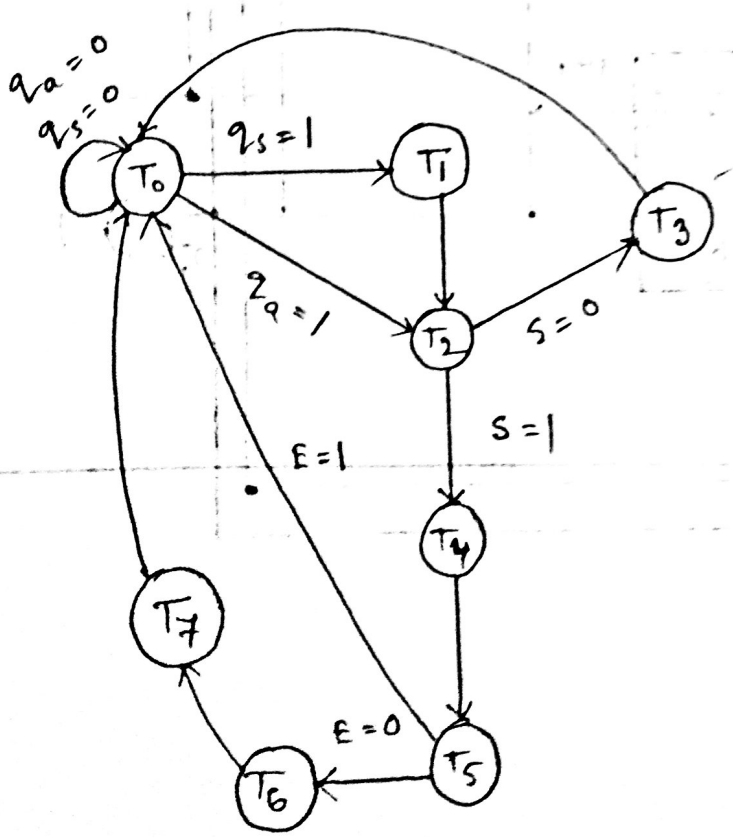
$q_m \rightarrow$ User Input

fig: 10-17: Design of control for binary multiplier

* figure - p 10.28 & p 10.27 - JK flip flop দিয়ে Solve করতে হবে।

08 Jun 2017

* Design the control specified in the following figure by the register and decoder method use 3 J-K flip flops.



(a) Excitation Table

	Present state			Input				Next state			flip flop inputs					
	G_3	G_2	G_1	q_a	q_s	E	S	G_3	G_2	G_1	JG_3	KG_3	JG_2	KG_2	JG_1	KG
T_0	0	0	0	0	0	x	x	0	0	0	0	x	0	x	0	x
T_0	0	0	0	x	1	x	x	0	0	1	0	x	0	x	1	x
T_0	0	0	0	1	x	x	x	0	1	0	0	x	1	x	0	x
T_1	0	0	1	x	x	x	x	0	1	0	0	x	1	x	x	1
T_2	0	1	0	x	x	x	0	0	1	1	0	x	x	0	1	x
T_2	0	1	0	x	x	x	1	1	0	0	1	x	x	1	0	x
T_3	0	1	1	x	x	x	x	0	0	0	0	x	x	1	x	1
T_4	1	0	0	x	x	x	x	1	0	1	x	0	0	x	1	x
T_5	1	0	1	x	x	0	x	1	1	0	x	0	1	x	x	1
T_6	1	1	0	x	x	x	x	1	1	1	x	0	x	0	1	x
T_7	1	1	1	x	x	x	x	0	0	0	x	1	x	1	x	1
T_5	1	0	1	x	x	1	x	0	0	0	x	1	0	x	x	1

$JG_3 = ST_2$

$KG_3 = T_7 + T_5E$

$JG_2 = q_aT_0 + T_1 + E'T_5$

$KG_2 = ST_2 + T_3 + T_7$

$KG_1 = 1$

$JG_1 = q_sT_0 + S'T_2 + T_4 + T_6$

10 Jun 2017

PLA Control

Table 10-6 : state table for control circuit

	Input					Output					
	Present state		Inputs			Next state		Outputs				
	G_2	G_1	q_m	P_2	Q_1	G_2	G_1	T_0	T_1	T_2	L	T_3
T_0	0	0	0	x	x	0	0	1	0	0	0	0
T_0	0	0	1	x	x	0	1	1	0	0	0	0
T_1	0	1	x	x	x	1	0	0	1	0	0	0
T_2	1	0	x	x	0	1	1	0	0	1	0	0
T_2	1	0	x	x	1	1	1	0	0	1	1	0
T_3	1	1	x	0	x	1	0	0	0	0	0	1
T_3	1	1	x	1	x	0	0	0	0	0	0	1

* $T_2 \rightarrow T_3$ দুইভাবে যাওয়া যায়, $Q_1 = 1, 0$ এর জন্য।
কারণ, L এর জন্য তা সমাধান করা যায়।

* $T_3 \Rightarrow$ কোন state এ কাজ করে তা determine করে

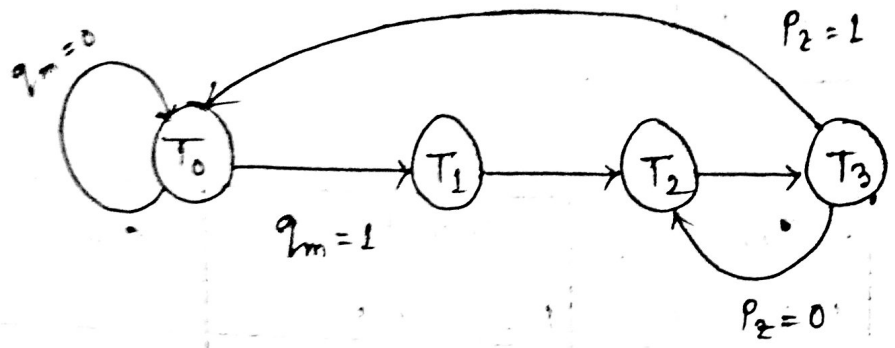
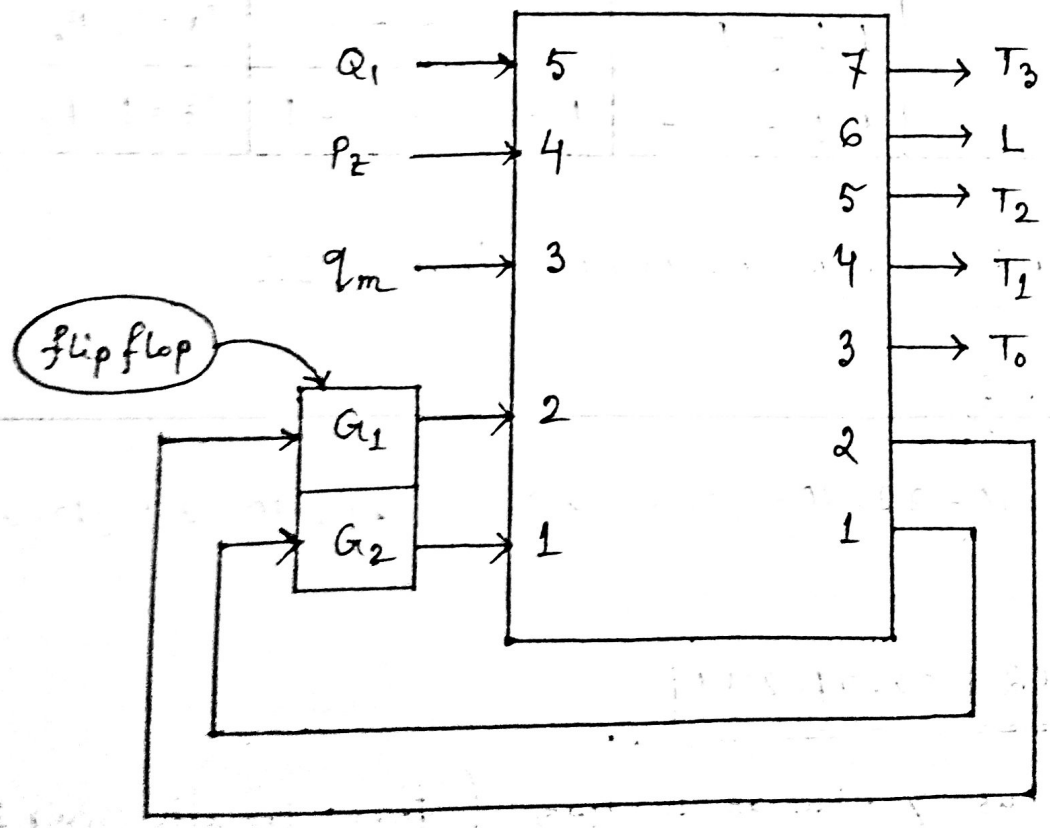


Fig 10-18 PLA control for binary multiplier

(a) block diagram



* $Q_1 = 1$ হলে L, T_2 active

$Q_1 = 0$ হলে, T_2 active

(b) PLA Program table

Product terms	Inputs 1 2 3 4 5	Outputs 1 2 3 4 5 6 7	Comments
1	0 0 0 - -	- - 1 - - - -	$T_0 = 1$, $q_m = 0$
2	0 0 1 - -	- 1 1 - - - -	$T_0 = 1$, $q_m = 1$
3	0 1 - - -	1 - - 1 - - -	$T_1 = 1$
4	1 0 - - 0	1 1 - - 1 - -	$T_2 = 1$, $Q_1 = 0$
5	1 0 - - 1	1 1 - - 1 1 -	$T_2 = 1$, $L = 1$, $Q_1 = 1$
6	1 1 - 1 -	- - - - - - 1	$T_3 = 1$, $P_2 = 0$
7	1 1 - 0 -	1 - - - - - 1	$T_3 = 1$, $P_2 = 1$

Question: Design the control logic PLA.

Exercise:

10-14, 10-22, 10-23, 10-25, 10-26, 10-27, 10-28

Quiz-02 [02.07.2017]

syllabus => Chapter 10 পর্যন্ত (from Lec after Quiz 1)

[Counter, Booth, Chapter-10]

06 July 2017

SAP - I

Simple - As - Possible Computer

10-1 Architecture

Program Counter:

It is a part of the control unit, counts from 0000 to 1111. Its job is to send to the memory the address of the next instruction to be fetched and executed.

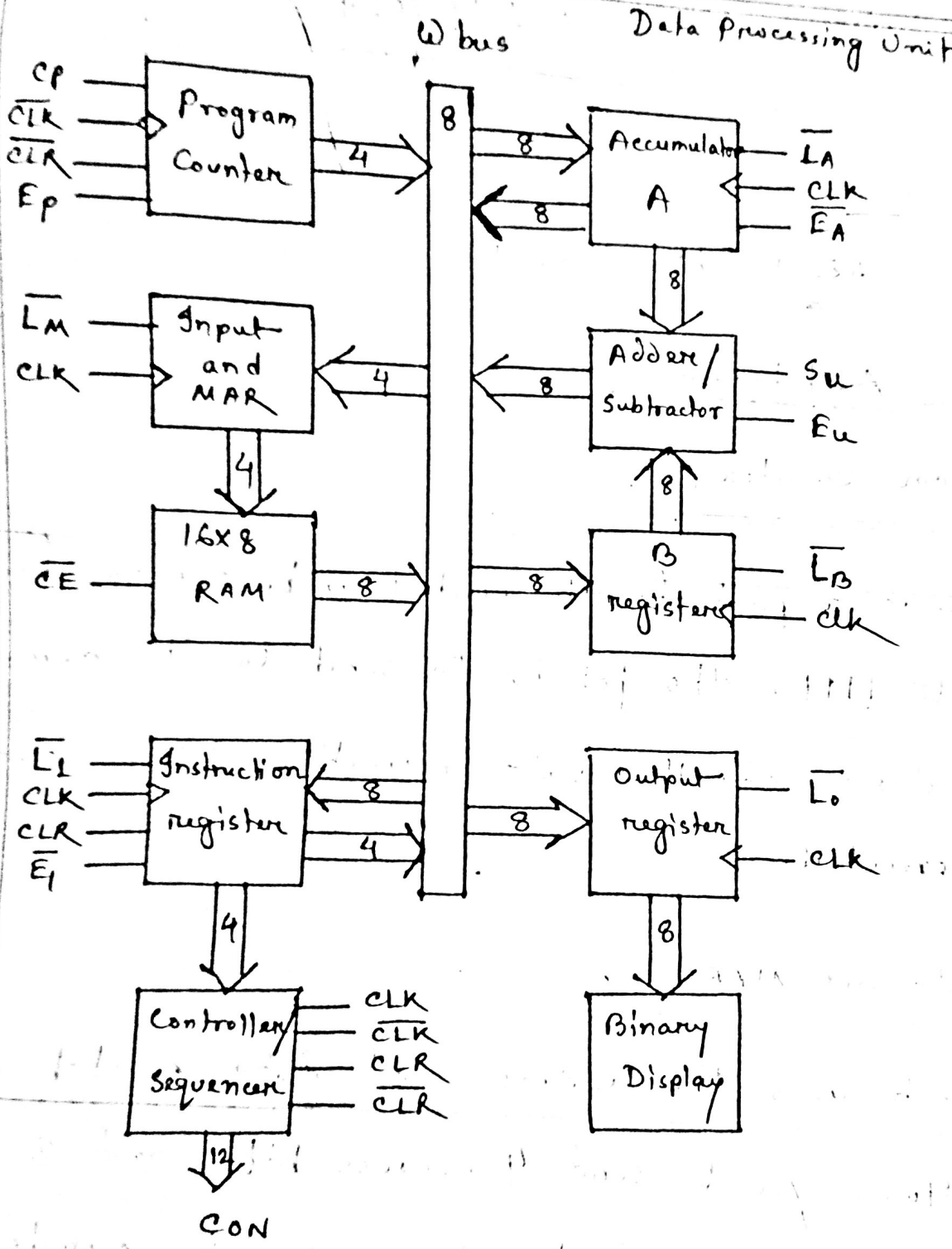
Input and MAR:

It includes the address and data switch registers. It allows you to send 4 address bits and 8 data bits to the RAM. Instructions and data words are written into the RAM before a computer run.

The RAM:

The RAM is a 16X8 static TTL RAM. You can program the RAM by means of the address and data switch registers.

Control Unit



Instruction Register :

It is a part of control unit. To fetch an instruction from the memory, the computer does a memory read operation. This places the content of the addressed memory location on the ω bus.

Controller - Sequencer :

Before each computer run, a $\overline{CLR}$ signal is sent to the PC and CLR signal to the IR.

This reset the PC to 0000 and wipes out the last instruction in the IR.

A clock signal is sent to the buffer registers, this synchronizes the operation of the computer.

A 12-bit word comes out of the controller sequencer that controls the rest of the computer.

$$CON = C_P E_P \overline{L_M} \overline{CE} \overline{L_I} \overline{E_I} \overline{L_A} E_A S_U F_U \overline{E_B} \overline{L_O}$$

[2] Accumulator:

It is a buffer register that stores the intermediate answer during a computer run.

[3] The Adder - Subtractor:

SAP-I uses a 2's-complement adder-subtractor. When Su = low, A = A + B and when Su = high, A = A + B'

[4] B register:

It is a buffer register that is used in anithmetic operation.

[5] Output Register:

When E_R is high and $\overline{L_O}$ is low, the next positive clock edge loads

the accumulator word into the output register.

Binary Display:

It is a row of light-emitting diodes (LEDs).

08 July 2017

* Instruction set 4-bit হবে
* Data 8-bit হবে

10-2 | Instruction Set

LDA
ADD
SUB
HLT ⟶ Whole ckt off করে দিবে।
OUT ⟶ Accumulator এর Data Output Register এ নিয়ে যাবে।

* SUB এর (আগে SU High করবে।
* ADD এর (আগে SU Low করবে।

0H	LDA	9H	
1H	ADD	FH	
2H	SUB	8H	
:	HLT		
:	OUT		
8H	0000	0011 | 03H	
9H	0010	0010 | 22H	24H − 03H = 21H
AH	0000	0101 | 05H	
FH	0000	0010 | 02H	

A + B

22H + 05H

```
    01H
    02H
    03H
   ----
EA  04H
    05H
     :
     :
    FH
```

CON = Cp Ep

00

(*)

Mnemonics

LDA ADD

Example

Address	Mnemonics
0H	LDA 9H
1H	ADD AH
2H	SUB BH
3H	ADD CH
4H	HLT
5H	OUT
6H	FFH
7H	FFH
8H	FFH
9H	ABH
AH	0AH
BH	05H
CH	A0H
DH	FFH
EH	FFH
FH	FFH

Output = ?

15 July 2017

10-3 Programming SAP-1

Mnemonics	Op Code
LDA	0000
ADD	0001
SUB	0010
OUT	1110
HLT	1111

Example 10-2

Translate the following program into SAP-1 machine language.

Address	Instruction
0H	LDA 9H
1H	ADD AH
2H	ADD BH
3H	SUB CH
4H	OUT
5H	HLT

Address	Function
0000	0000 1001
0001	0001 1010
0010	0001 1011
0011	0010 1100
0100	1110 xxxx
0101	1111 xxxx

Example - 10-3

How would you program SAP-I to solve this arithmetic problem?

$16 + 20 + 24 - 32$

The numbers are in decimal form.

Soln:

0H	LDA 6H	→ Address দিবে
1H	ADD 7H	
2H	ADD 8H	
3H	SUB 9H	
4H	OUT	
5H	HLT	
6H	10H	
7H	14H	
8H	15H	
9H	20H	

* আগে instruction যাবে, then data.

* Max^m 8টি Address যাবে।

* HLT ⇒ Instruction set শেষ, আর কিছু না Data!

Example 10-4

Convert the code into hexadecimal representation:

Address	Bus (Hex)
0H	06H
1H	17H → operation
2H	18H → Address
3H	29H
4H	ExH → যাঁকা হারকমত না তাই— Don't Care!
5H	FXH → HLT
6H	10H
7H	14H
8H	18H
9H	20H

16 July 2017

10-4 Fetch Cycle

Ring Counter

Fetch

$T = \overbrace{T_6\ T_5\ T_4}\ T_3\ T_2\ T_1$

$T = 0\ 0\ 0\ 0\ 0\ 1$

$T = 0\ 0\ 0\ 0\ 1\ 0$

⋮

$T = 1\ 0\ 0\ 0\ 0\ 0$

* 1টি Instruction Complete করতে 6টি (Max) state লাগে।
* Ring Counter এর একটি instruction এর জন্য বারবার চালাতে হয়।

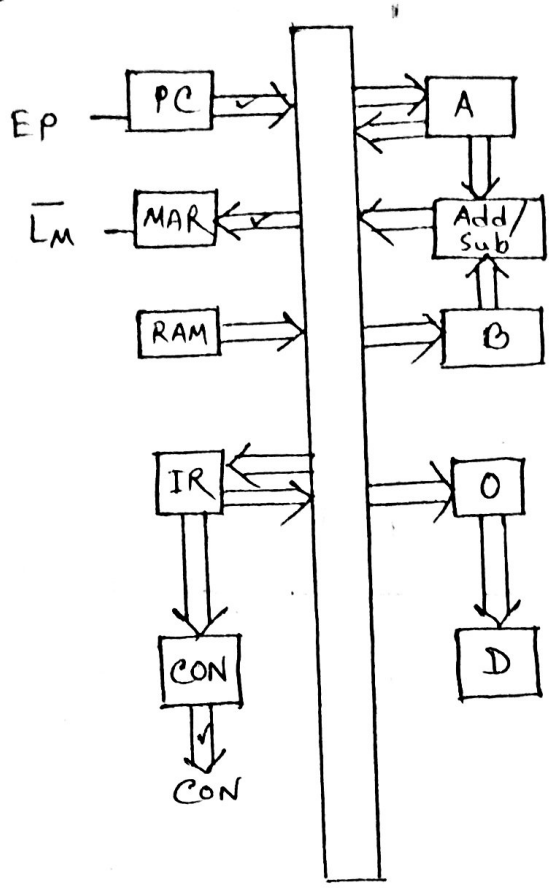
Q. Ring Counter এর Timing Diagram, Block Diagram (fig-10.2) (Imp for Xm).

* PC ও Ring Counter এর মধ্যে পার্থক্য।

Here fetch means ⇒ Read করে নিয়ে আসা প্রস্তুত করা।

Address State (Fig-10.3) T_1 state

Ep = Enable Program Counter



$CON = C_p\ E_p\ \overline{L_M}\ \overline{C_E}\ \ \overline{L_I}\ \overline{E_I}\ \overline{L_A}\ E_A\ S_U\ E_U\ \overline{L_B}\ \overline{L_O}$

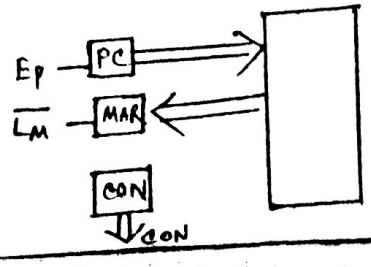
$0\ 1\ 0\ 1\ \ 1\ 1\ 1\ 0\ \ 0\ 0\ 1\ 1$

* xm ৬ fig আঁকতে হবে [মুখস্থ Active গুলো আঁকলে চলবে]

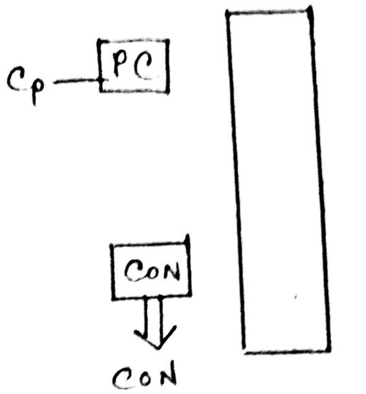
+ Timing Diagram [6 Marks total]

[No Explanation needed]

* T_1 state (only Actives)



* Increment State

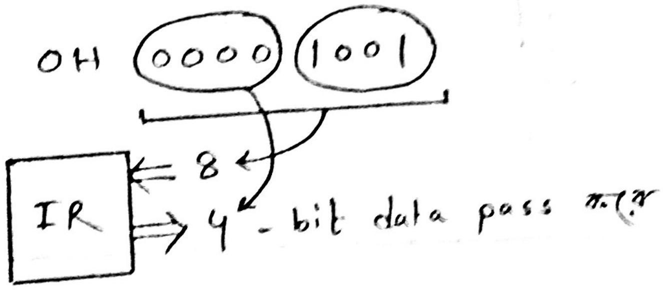


* যদি (–) দেয় সেখানে 0 এবং যেখানে যদি নেই, সেখানে 1 দিতে হবে।

* Memory Address এ যে Address থাকে, সেটাই RAM এর Address.

	C_p E_p $\overline{L_M}$ $\overline{C_E}$	$\overline{L_I}$ $\overline{E_I}$ $\overline{L_A}$ E_A	S_U E_U $\overline{L_B}$ $\overline{L_O}$
CON =	1 0 1 1	1 1 1 0	0 0 1 1

* T_3 State :

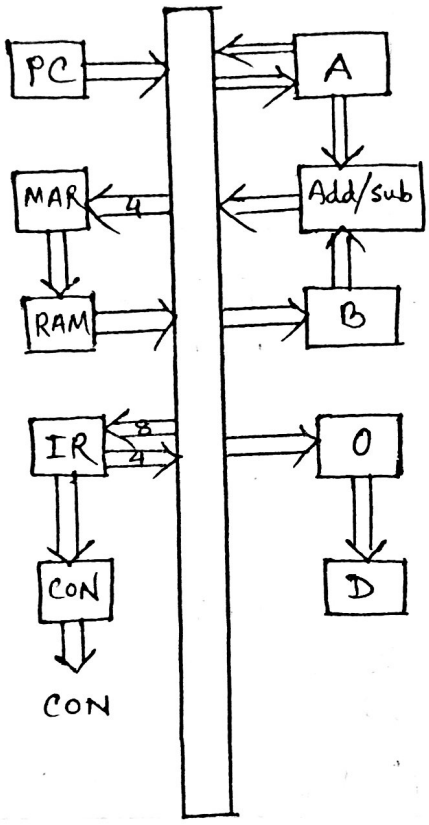


* Fetch cycle শেষ !

20 July 2017

SAP-I

LDA Routine:

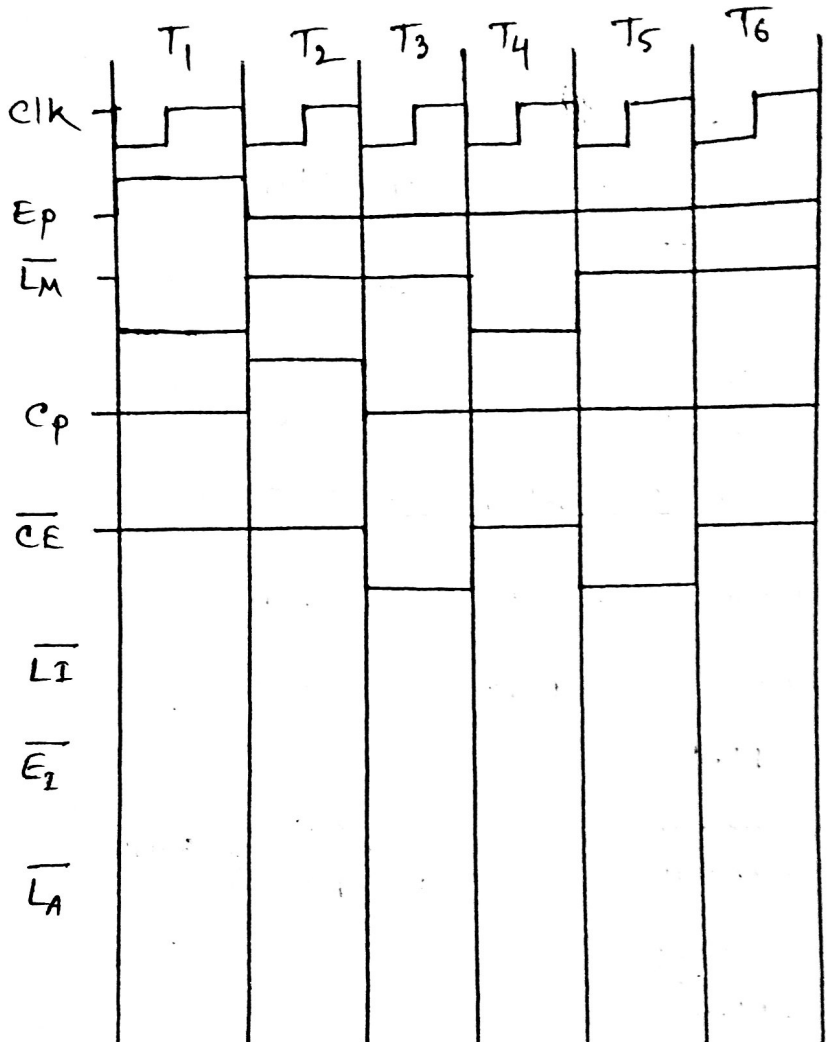


⊞ 0H LDA 9H

0000 1001

* T4 state ↪ MAR ↪ 1001 (Address) Load হবে!
* IR থেকে ৪ই Address, W-bus এবং MAR ↪ যাবে Last ↪ ৪ 4-bit.
* RAM ↪ MAR address ৪ই Data আসবে। তারপর W-bus ↪ যাবে। W-bus থেকে Accumulator ↪ যাবে।
* LDA ৪ই ক্ষেত্রে T5 পর্যন্ত অপেক্ষা করবে!

Follow sheet's Fig – 10.4



(fig - 10.5)

* SUB এর ক্ষেত্রে Su, Eu দুটোই চেঞ্জ হবে।
* OUT Routine এখানে 1 টি state আঁকলেই হবে।
* HLT Routine থেকে এখানে কিছু বের হয় না।
* fig - 10.7 ও T_6 ও Su High হবে।
* Fall - 16 Question

Fetch – 3
LDA – 2
ADD – 3

8টি (চারটি) Diagram [final ও 5 Mark]

0H – 7H ⇒ Instruction
8H – FH ⇒ Data

Quiz ↳ Architecture নিয়ে খুব গুরুত্ব।

22 July 2017

SAP-II

* SAP-II এর main কাজ 'JUMP করা'
* SAP-II এর Instruction set 42টি।
* Rom সাধারণত 0000H থেকে 07FFH পর্যন্ত
* RAM সাধারণত 0800H থেকে FFFFH পর্যন্ত
* Table - 11.1
* Rom → 0000H – 07FFH
  RAM → 0800H – FFFFH
* Example – 11.2
* Example - 11.4

MOV → 6টি → 1 cycle Machine instruction

ADD/SUB → 2টি

ADD B SUB B
ADD C SUB C

* Direct Register এ Address!
* Example – 11.3, 11.4
* Sign flag = 1 হলে Jump করবে!

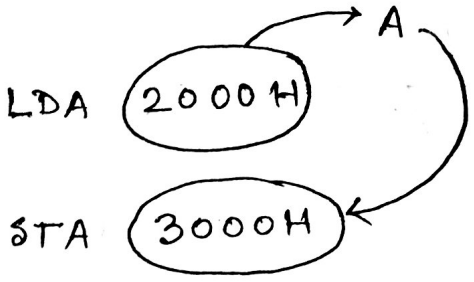
JZ → ZF = 1
JNZ → Z = 0

29 July 2017

11-3

Memory Reference

Introduction :



8-bit

3 Cycle

0000H	3AH
0001H	00H
0002H	20H
0003H	
0004H	

T3

```
MVI  A, 10H
MVI  B, 10H
MVI  C, 10H
```

* Fig - 11.2
* Example - 11.2 (Example - 11.1 এর - Conversion)
* Opcode যেভাবে রাখবে, Address start এর - 2000H থেকে।
* Example - 11.1 দেখবে।

DSD Lab (Quiz)

- 4 set Question
  20 MCQ, 15 min time

* 4-bit Adder এর ক্ষেত্রে 15 এবং 1 যোগ করলে Zero flag কী হবে?
* Binary Display কী কয়?
  — Read / Write
* 5-bit এর ক্ষেত্রে কয়টা Clock pulse লাগে?

(*)

2000H JM ← S=0

2000H

sign flag = 1 হলে Jump করে।

JZ ✓ Z=1

JNZ ✓ Z=0

03 August 2017

JNZ

2000H CALL 5000H

2001H

CALL 5000H

RET

FFFEH	01H
FFFFH	20H

5000H ---

5001H ---

5002H ---

5003H RET — HLT

* SAP-II তে কোনো multiplier নেই।

* Example – 11.9

Example – 11.10

Example – 11.11, 11.12

(function)

Example – 11.5, 11.6, 11.7, 11.8 → comment (চাওয়া লাগবে না!

JZ 2009H

JMP 2002H

* 12×8 ⇒ 8 বার 12 যোগ করা!

CMA

1's Complement

A = 0101 1100
→ 1010 0011

*'A' থাকলে,
⇒ Register এর নাম

*'I' থাকলে,
⇒ Immediate

ANA

ANA B
ANA C

A = 0011 1110
B = 0100 0110

A = 0000 0110

ORA A
XRA A

RAL ← 10 011011
RAR → 00110111
11100110

ANI 33H
0011 0011

ANI
ORI
XRI

NOP

T_3/T_4

NOP
NOP
5 × 4 = 20

HLT

IN

IN byte
02H
01H

(2 No. Port use করা যায়)

OUT

OUT byte
OUT 03H
04H

42 Instructions (OPCODE ⇒ Table II.1)

LDA
STA
mvI A,
mvI B,
mvI C,

mov A,B
mov A,C
mov B,A
mov B,C
mov C,A
mov C,B

ADD B
ADD C
SUB B
SUB C
INR A
INR B
INR C
DCR A
DCR B
DCR C

JMP
JM
JZ
JNZ
CALL
RET
CMA
ANA B
ANA C
ORA B
ORA C
XRA B
XRA C

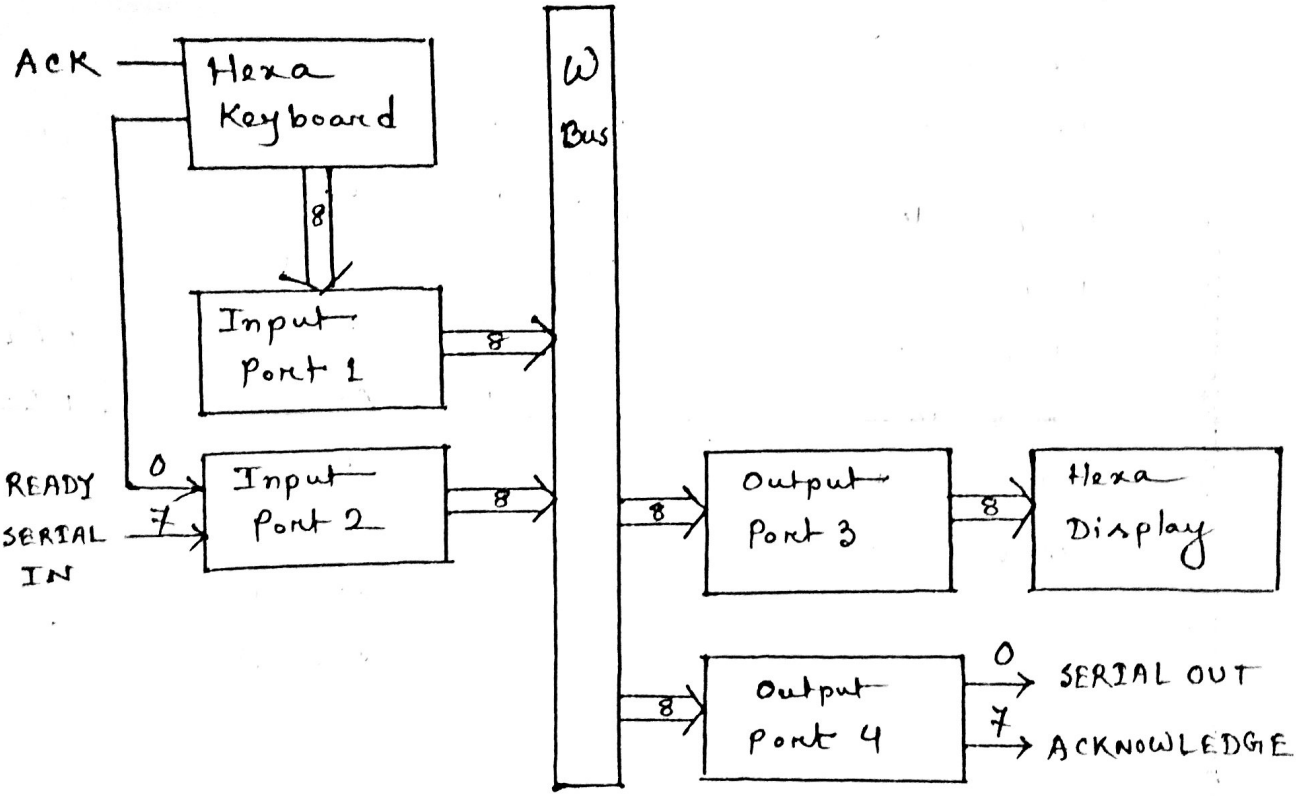
ANI —
ORI —
XRI —

NOP
HLT
IN
OUT
RAL
RAR

* Chapter-9 পৃষ্ঠা- 9.10 দেখ

* CMA ⇒ প্রত্যেকটা bit আগের Accumulator এর,
-প্রত্যেকটার 1's Complement করবে।

05 Aug 2017



** Q. Describe the Handshaking of SAP-II.

SAP-II Handshaking :

1. READY bit (bit 0, port 2) goes high.
2. Input the data in port 1 to the CPU.
3. ACKNOWLEDGE bit (bit 7, port 4) goes high to reset READY bit.

4. Reset the ACKNOWLEDGE bit.

Example - 11.13

N কে ASCII করতে, (4E)

Y কে ASCII করতে (59)

0000 000① (ASCII চিহ্নিত করতে)

Example – 11.14

A7 ---------- A0

RAR → shift right

Table 11.3

Page – 188

```
      MVI  C, 46H  ──→ 70
AGAIN: DCR  C
       JNZ  AGAIN
       NOP
       RET
```

* Time Delay এর ক্ষেত্রে must করতে আসবে!

```
        MVI B, 0AH          → 10        1 × 7 × 1 μs = 7 μs
LOOP1:  MVI C, 47H          → 71        10 × 7 × 1 μs = 70 μs
LOOP2:  DCR C                           71 × 4 × 1 μs = 284 μs
        JNZ LOOP2                       70 × 10 × 1 μs = 700 μs
        DCR B                           1 × 7 × 1 μs = 7 μs
        JNZ LOOP1                       10 × 4 × 1 μs = 40 μs
        RET                             9 × 10 × 1 μs = 90 μs
                                        1 × 7 × 1 μs = 7 μs
                                        1 × 10 × 1 μs = 10 μs
```

$$
\begin{aligned}
71 \times 4 \times 1\,\mu s &= 284\,\mu s \\
70 \times 10 \times 1\,\mu s &= 700\,\mu s \\
1 \times 7 \times 1\,\mu s &= 7\,\mu s \\
\hline
&991\,\mu s
\end{aligned}
$$

* Table 11.3 → Follow / Sheet
* Important বলতে → T-state
* 10/7 → যদি Jump করে আসলে, 10টি T-state লাগবে।
  যদি Jump না করে আসলে, 7টি T-state লাগবে!
* 1টি T-state ↔ 1 μs.
* xm ↔ T-state দেওয়া থাকবে!

MVI C, 46H ⟶ $1 \times 7 \times 1\ \mu s = 7\ \mu s$

AGAIN:

DCR C ⟶ $70 \times 4 \times 1\ \mu s = 280\ \mu s$

JNZ AGAIN ⟶ $69 \times 10 \times 1\ \mu s = 690\ \mu s$

Loop

$1 \times 7 \times 1\ \mu s = 7\ \mu s$

NOP ⟶ $1 \times 4 \times 1\ \mu s = 4\ \mu s$

RET ⟶ $1 \times 10 \times 1\ \mu s = 10\ \mu s$

$998\ \mu s$

* Example — 11.16, 11.18 ⟶ ধরনা না দিয়ে final Result বের করব!

Example — 11.19 ⟶ Less Important!!

Example — 11.21 পর্যন্ত দেখব!

Ch Chapter 9, 10 & SAP-I, II থেকেই 5 set Question Common পড়বে!