COMPUTER ORGANIZATION

MODERN COMPUTER ARCHITECTURE

BY

MOHAMED RAFIQUZZAMAN AND RAJAN CHANDRA

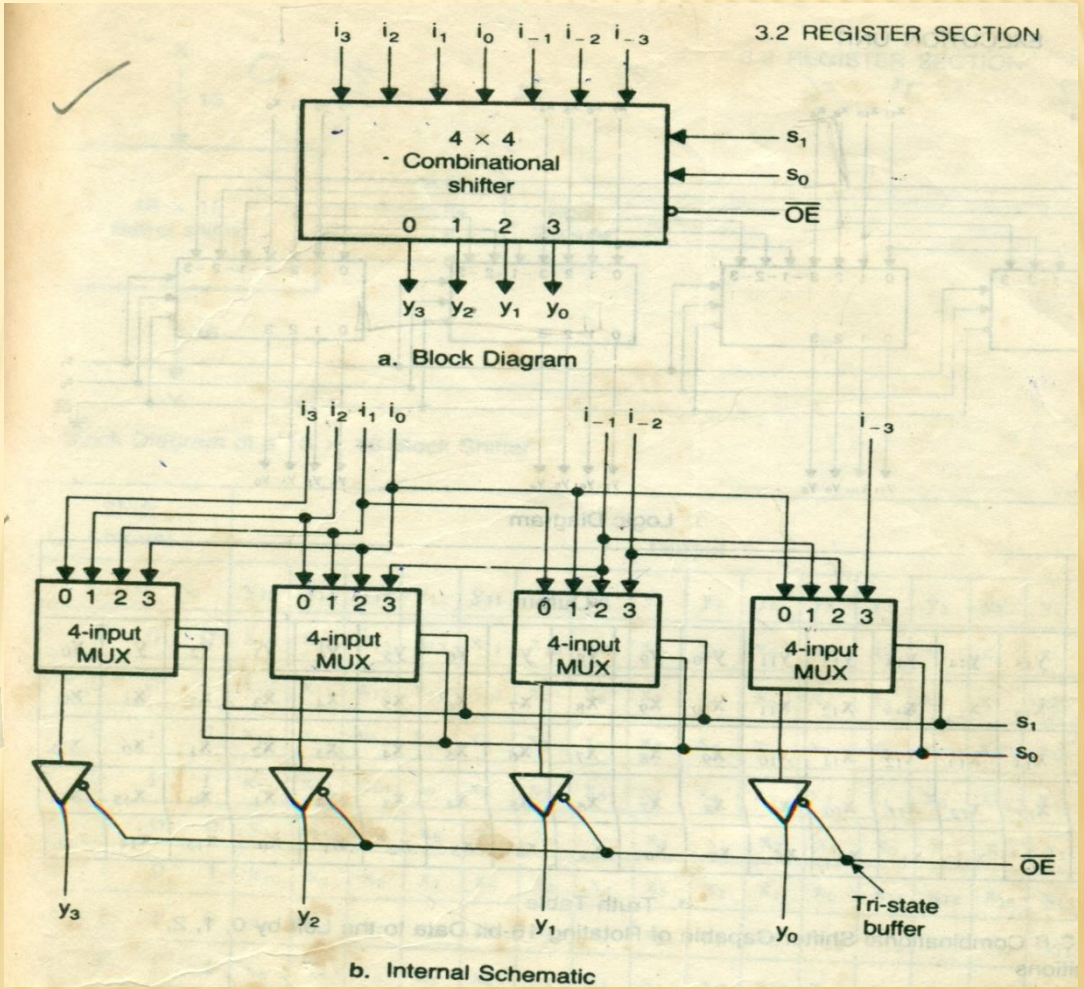
COMBINATIONAL SHIFTER DESIGN

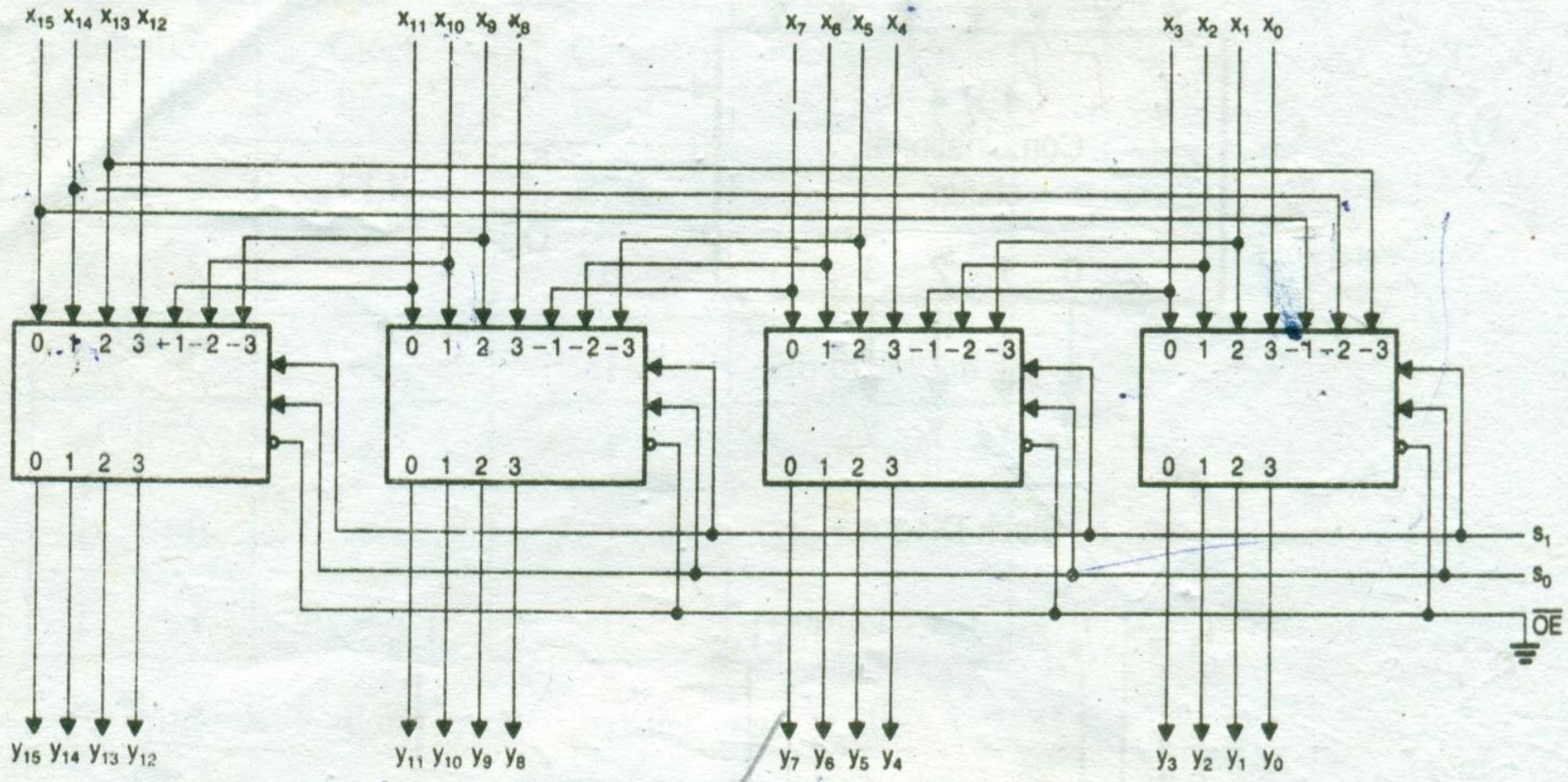
	Shift Count		Output				
$\overline{OE}$	s_1	s_0	y_3	y_2	y_1	y_0	Comment
1	X	X	Z	Z	Z	Z	Output lines float
0	0	0	i_3	i_2	i_1	i_0	Pass (no shift)
0	0	1	i_2	i_1	i_0	i_{-1}	Left shift once
0	1	0	i_1	i_0	i_{-1}	i_{-2}	Left shift twice
0	1	1	i_0	i_{-1}	i_{-2}	i_{-3}	Left shift three times

Note: Z—High-impedance state.
X—Don't care.

c. Truth Table

Figure 3.7 4 × 4 Combinational Shifter





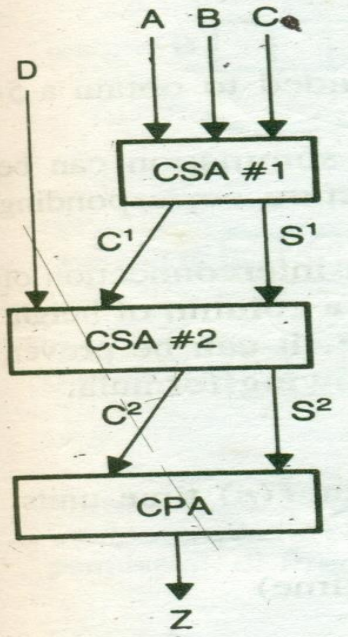
a. Logic Diagram

Shift Count		Output															
s_1	s_0	y_{15}	y_{14}	y_{13}	y_{12}	y_{11}	y_{10}	y_9	y_8	y_7	y_6	y_5	y_4	y_3	y_2	y_1	y_0
0	0	x_{15}	x_{14}	x_{13}	x_{12}	x_{11}	x_{10}	x_9	x_8	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0
0	1	x_{14}	x_{13}	x_{12}	x_{11}	x_{10}	x_9	x_8	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	x_{15}
1	0	x_{13}	x_{12}	x_{11}	x_{10}	x_9	x_8	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	x_{15}	x_{14}
1	1	x_{12}	x_{11}	x_{10}	x_9	x_8	x_7	x_6	x_5	x_4	x_3	x_2	x_1	x_0	x_{15}	x_{14}	x_{13}

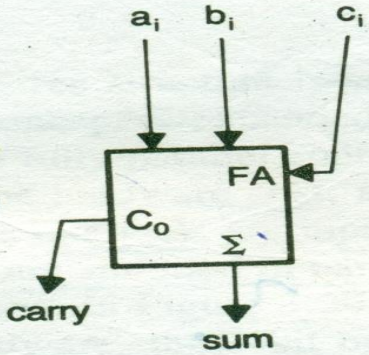
b. Truth Table

Figure 3.8 Combinational Shifter Capable of Rotating 16-bit Data to the Left by 0, 1, 2, or 3 Positions

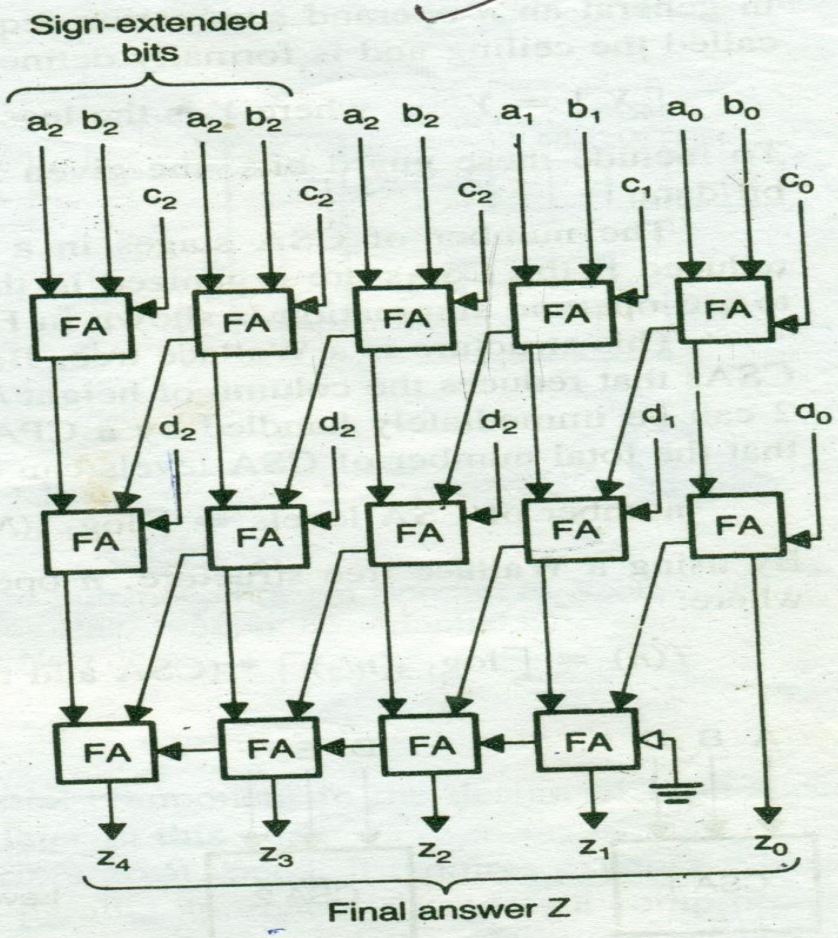
Carry Save Adder



a. Block Diagram



b. Basic Cell

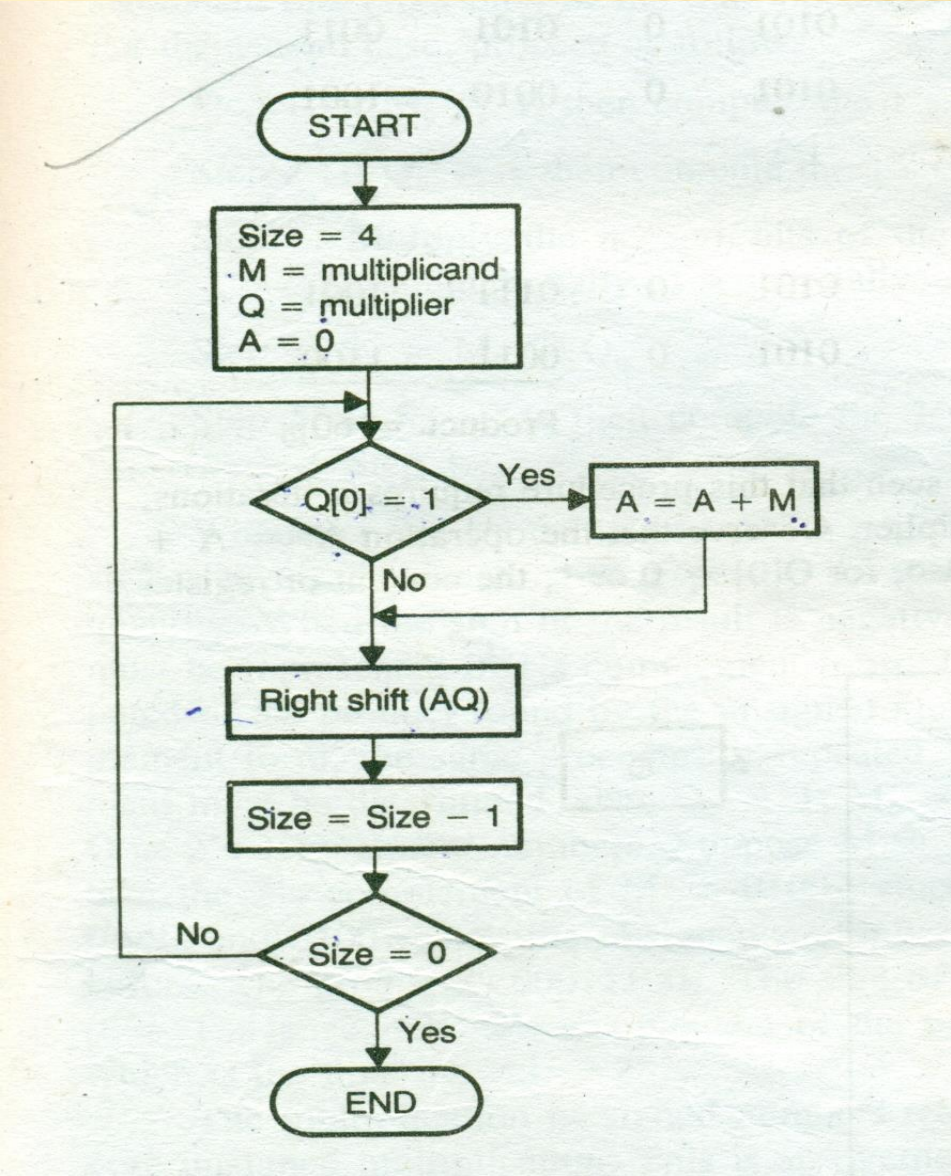


c. Hardware Schematic

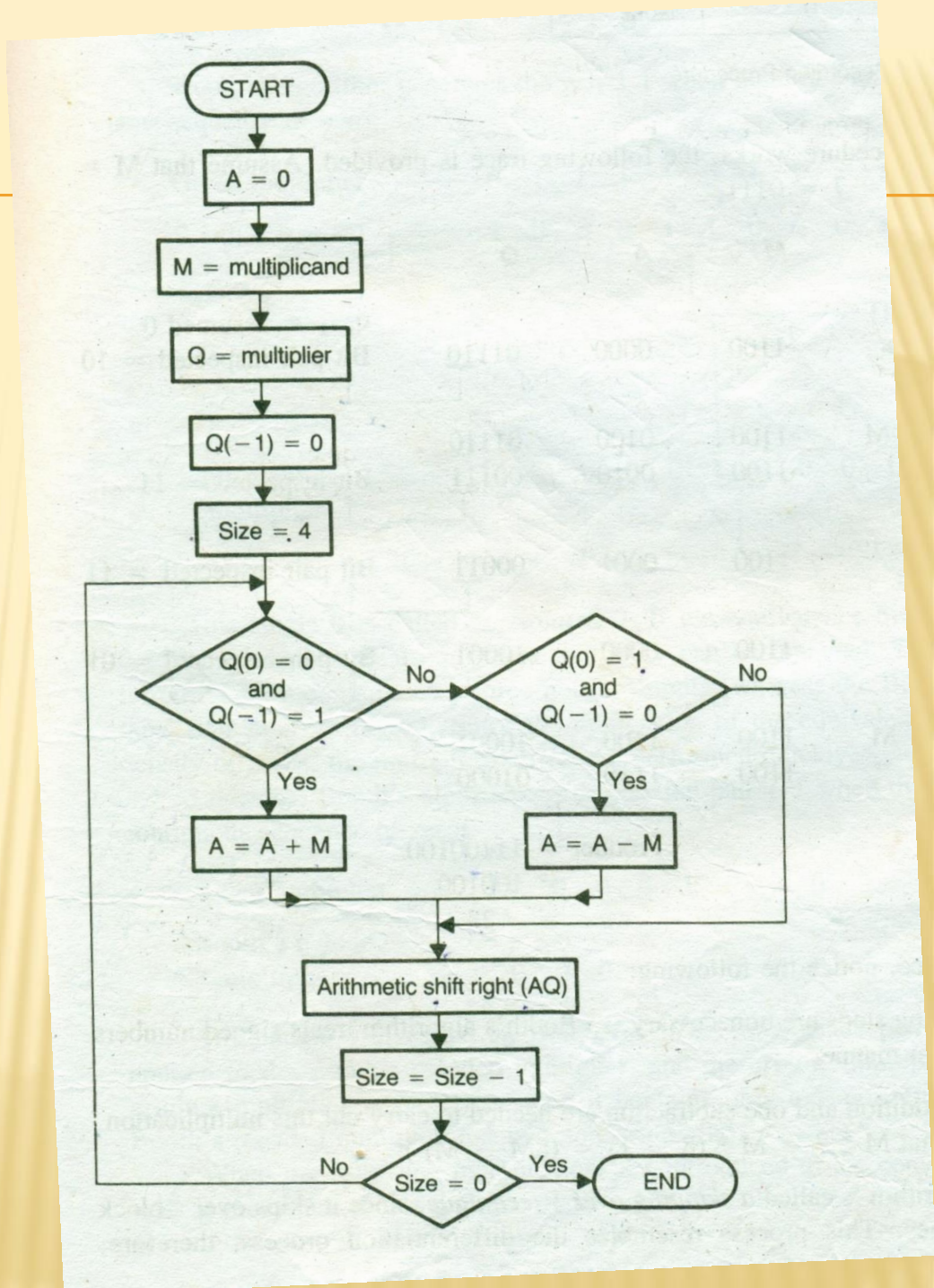
Figure 3.18 5-operand Summation

ALU

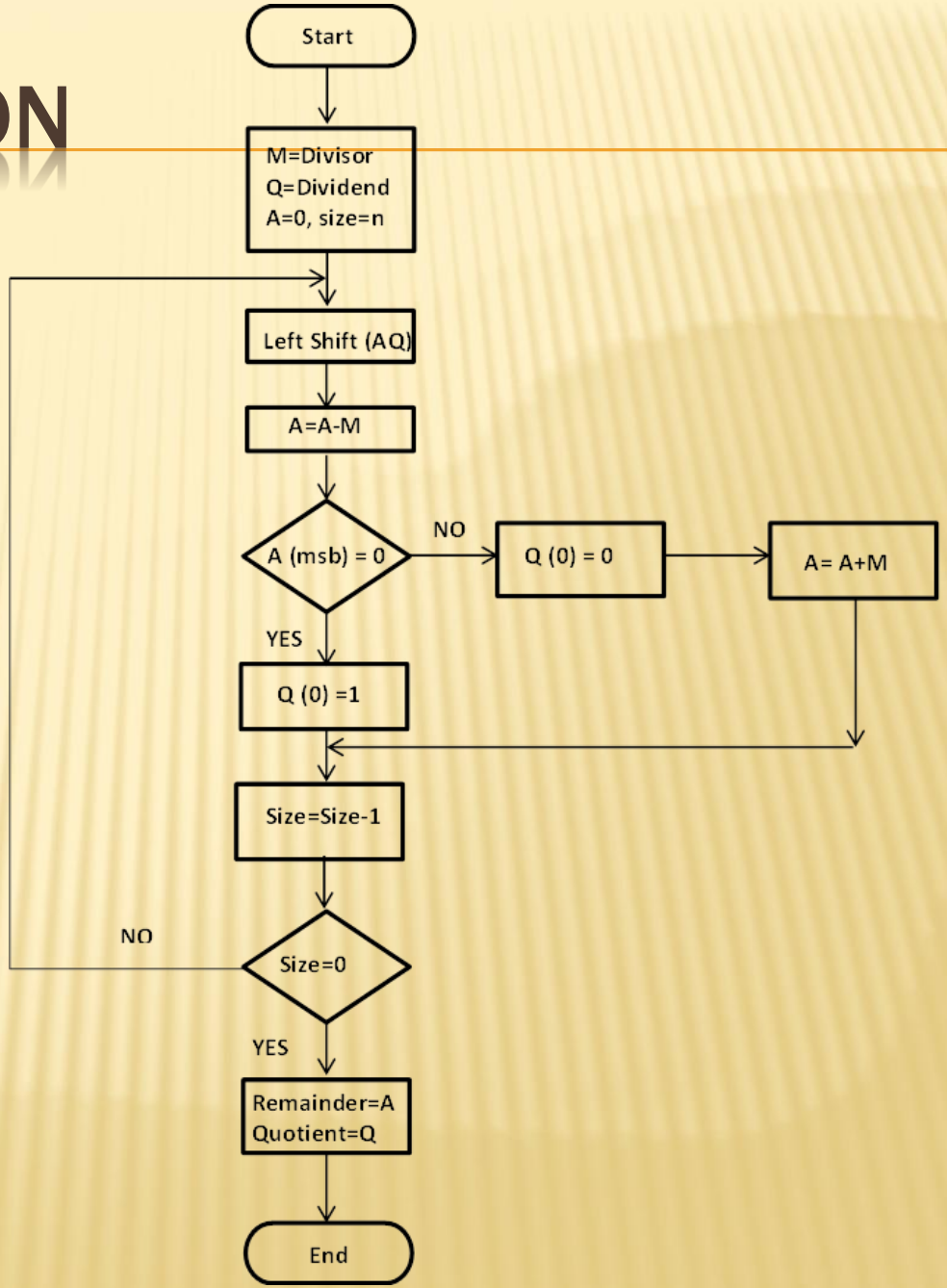
ADD AND SHIFT METHOD



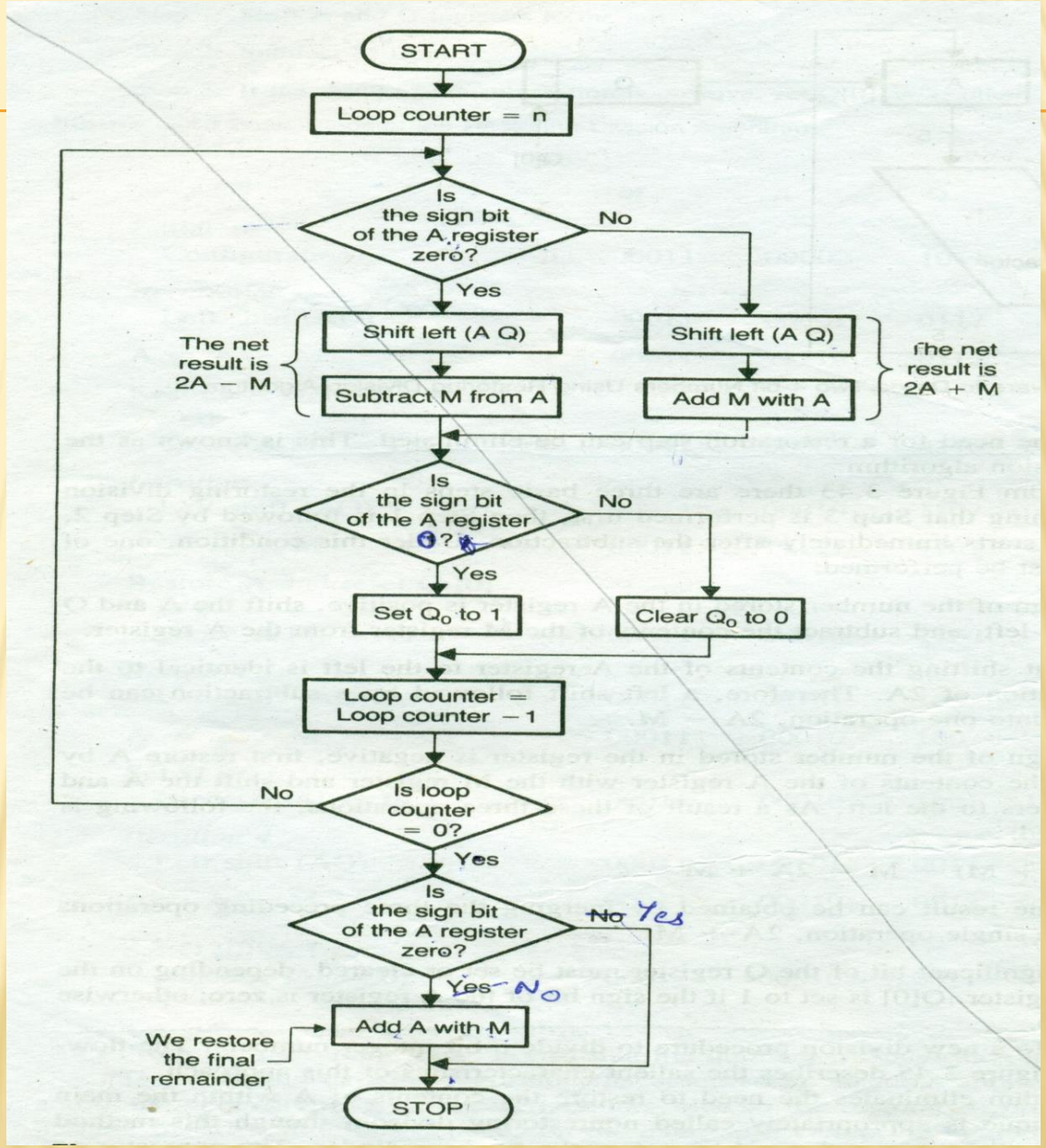
BOOTH'S ALGORITHM



RESTORING DIVISION



NONRESTORING DIVISION



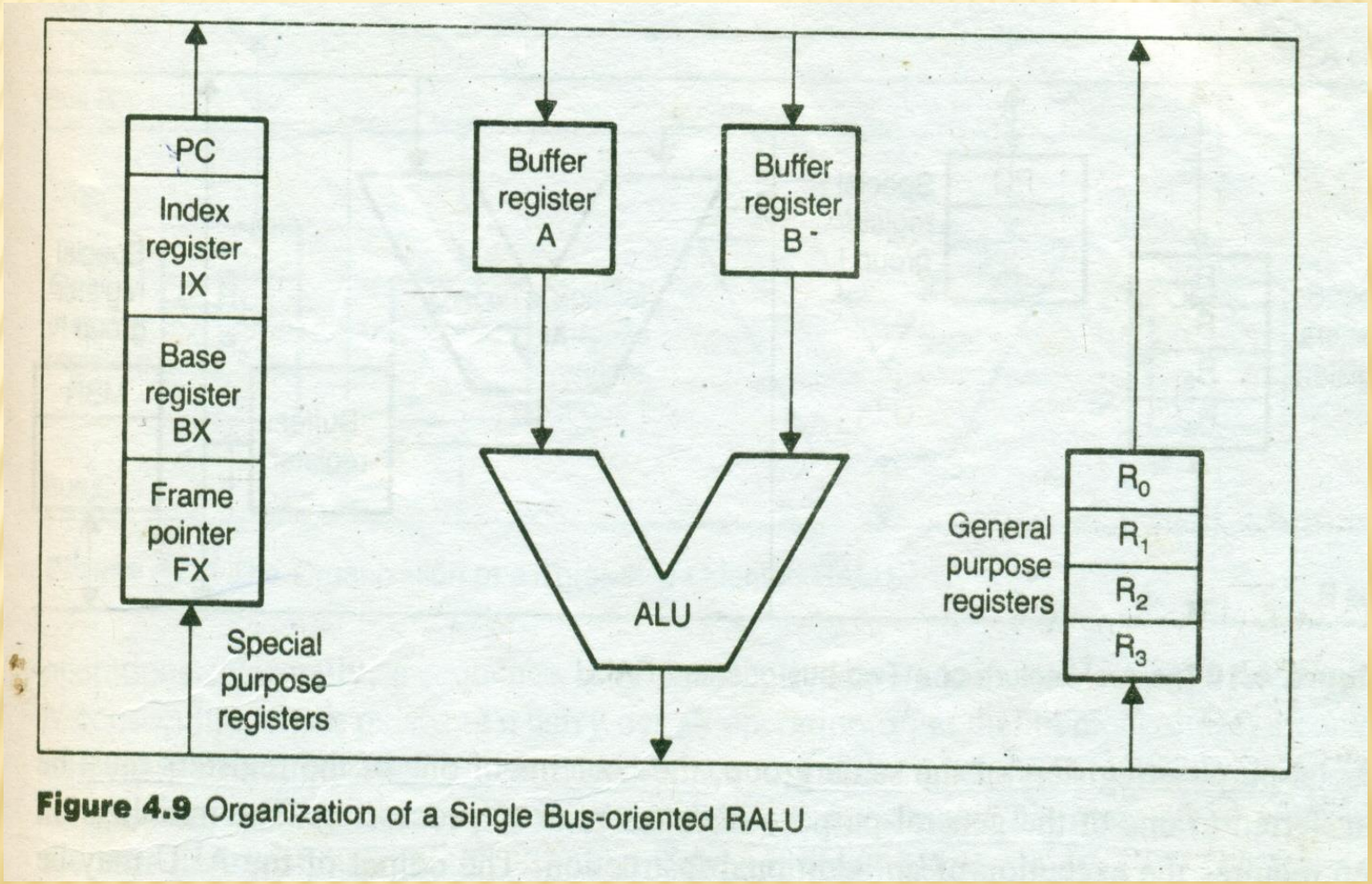


Figure 4.9 Organization of a Single Bus-oriented RALU

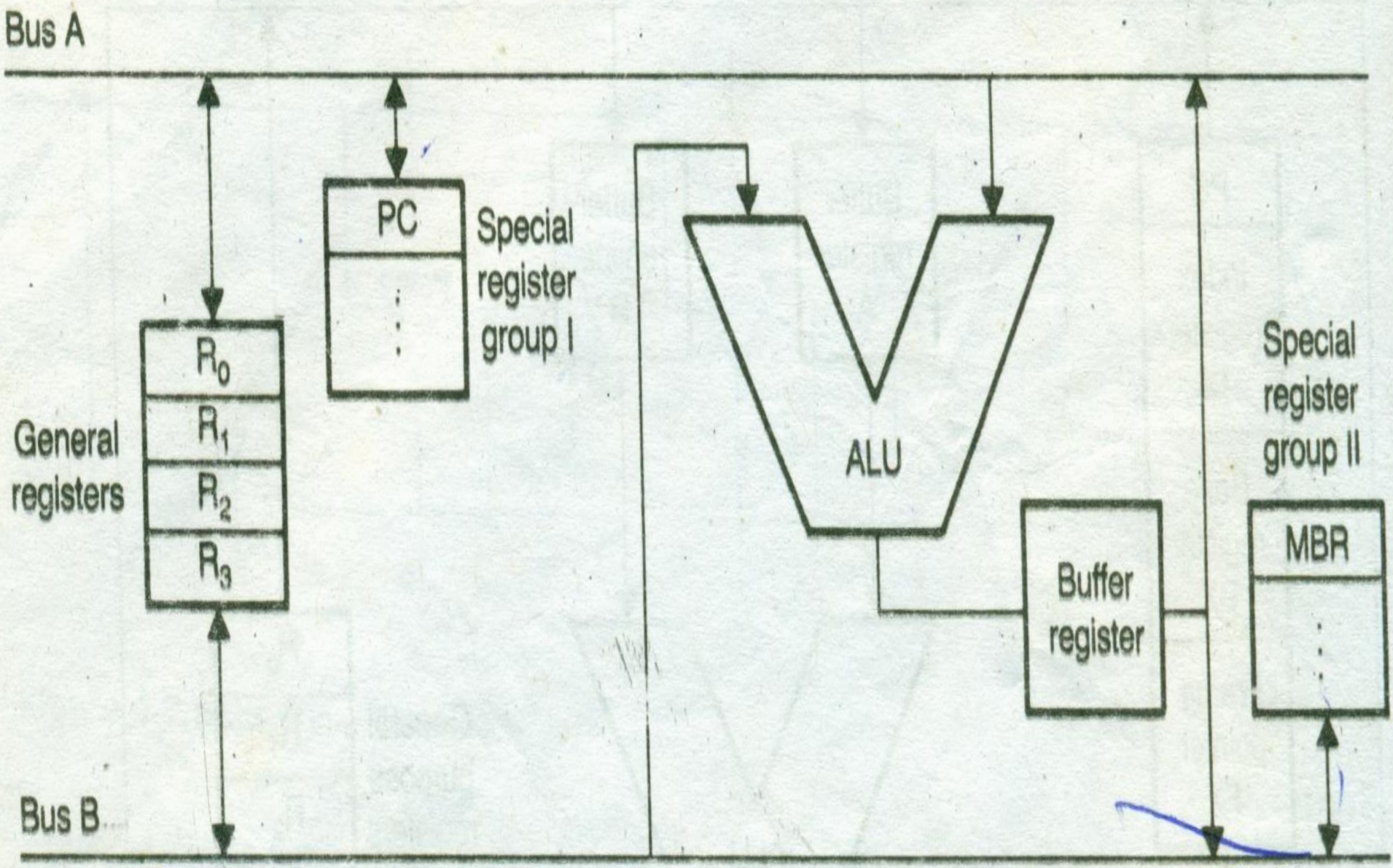


Figure 4.10 The Architecture of a Two-bus-oriented RALU

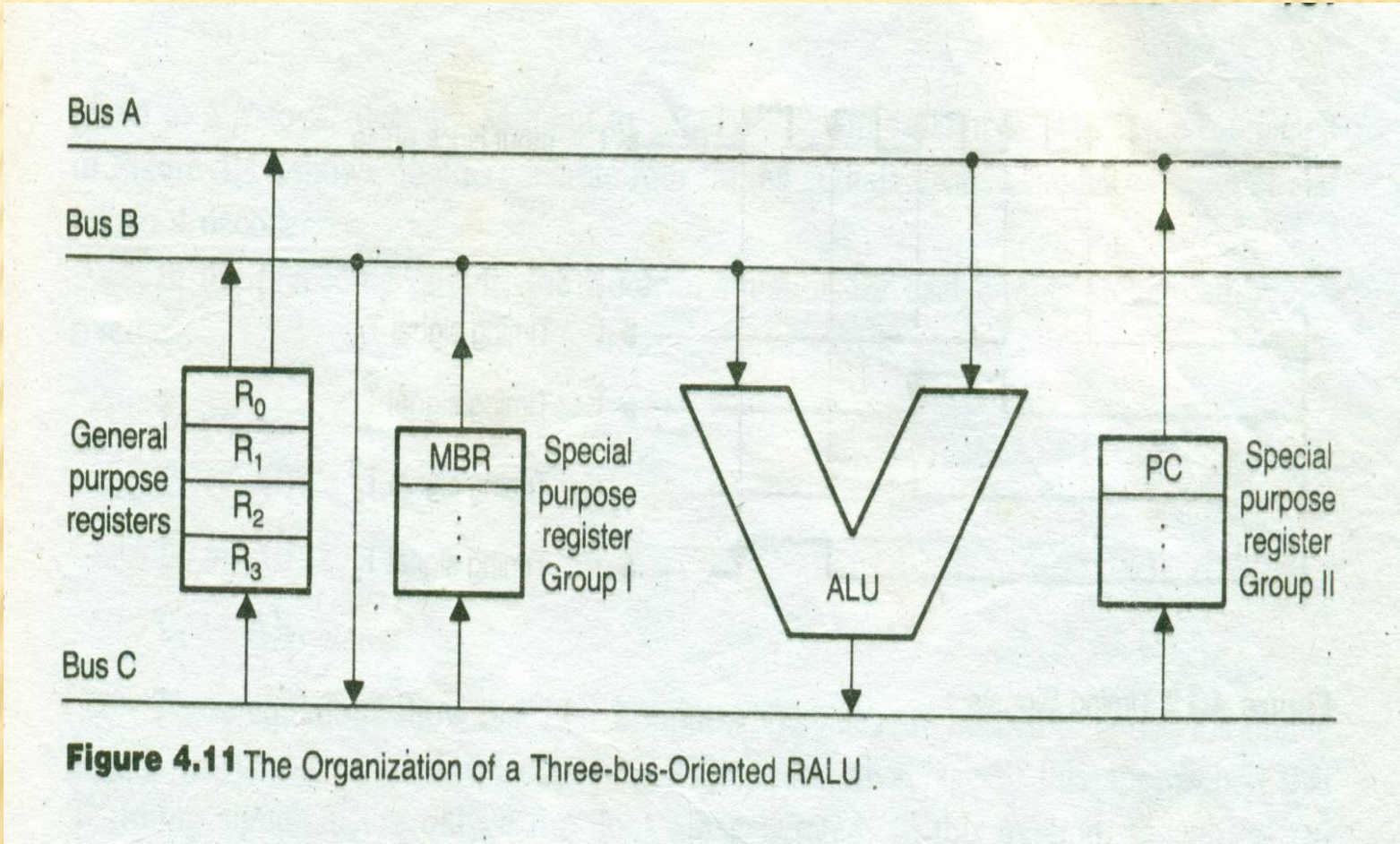


Figure 4.11 The Organization of a Three-bus-Oriented RALU

STEP 2

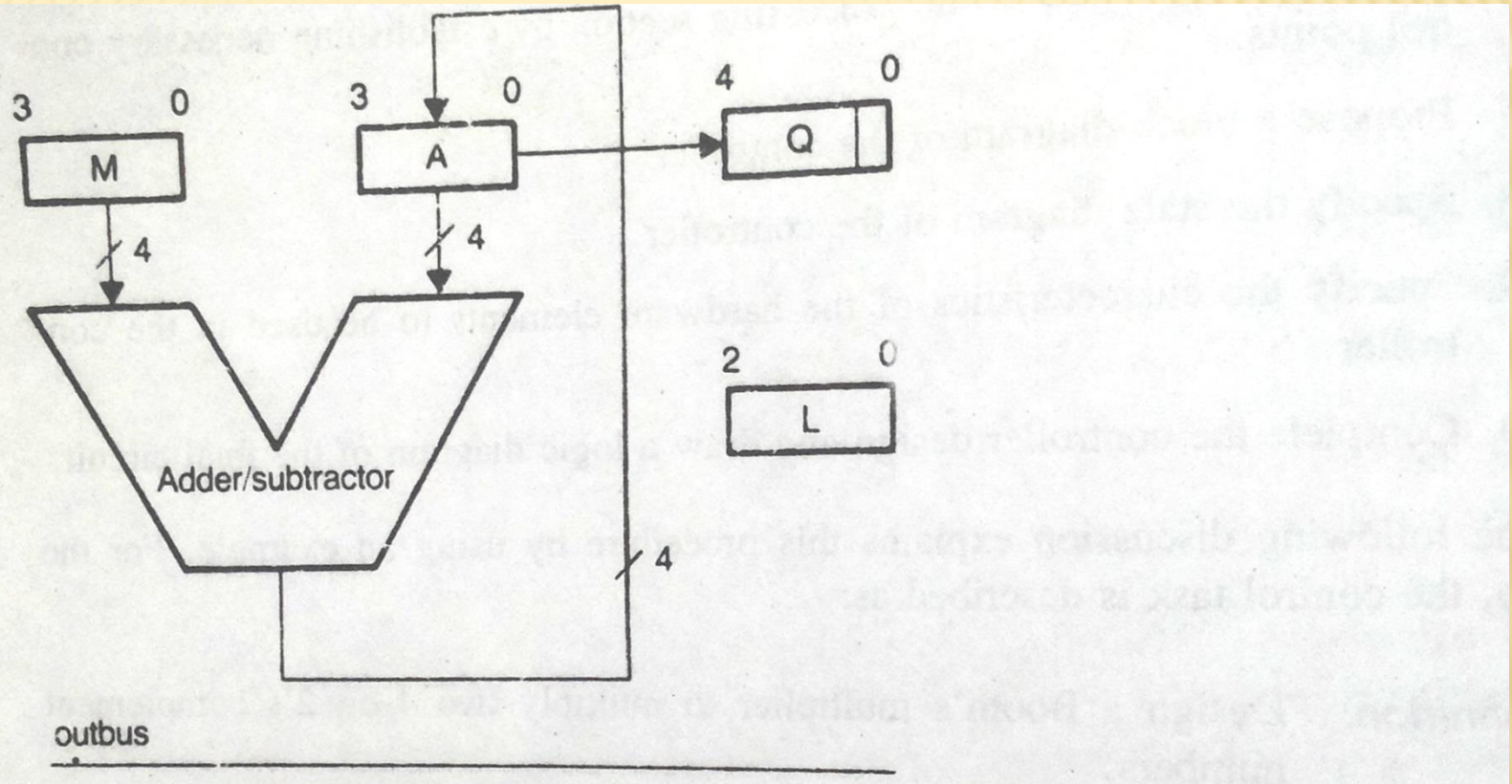


Figure 4.15 Organization of the Processing Section for Performing 4 × 4 Multiplication Using Booth's Algorithm

Step 3

Declare register A[4], M[4],Q[5], L[3];

Declare buses Inbus[4], Outbus[4];

Start: A←0, M←Inbus, L←4; C0,C1,C2

Q[4:1]←Inbus, Q[0]←0; C3

Loop: If Q[1:0]=01 then goto Add;

If Q[1:0]=10 then goto sub;

Go to Rshift

Add: A←A+M; C4,C5

Go to Rshift

Sub: A←A-M; C4',C5

Rshift: ASR(AQ), L←L-1; C6, C7

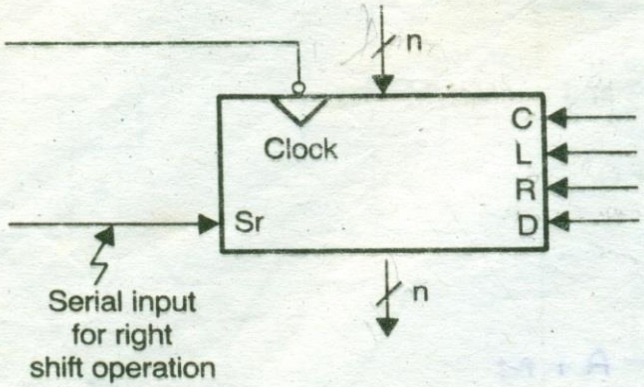
If L<>0 then goto loop

Outbus=A; C8

Outbus=Q[4:1]; C9

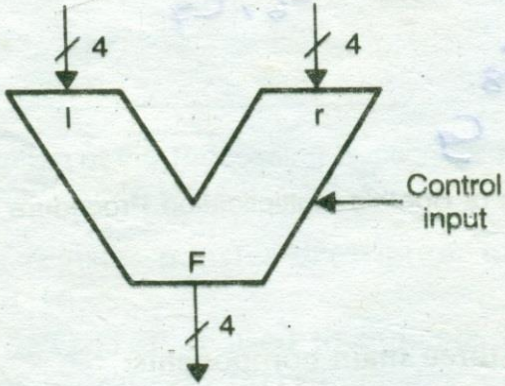
Halt: go to Halt;

STEP 5



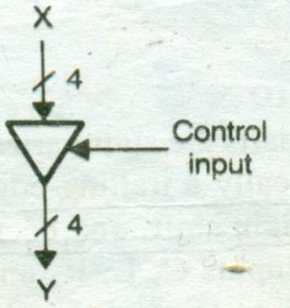
C	L	R	D	Clock	Action
1	0	0	0	↓	Clear
0	1	0	0	↓	Load external data
0	0	1	0	↓	Right shift
0	0	0	1	↓	Decrement by one
0	0	0	0	↓	No change

a. Storage Register



Control input	F
1	I + r
0	I − r

b. Adder/subtractor



Control input	Y
1	X
0	High Z

c. Tri-state Buffer

STEP 6

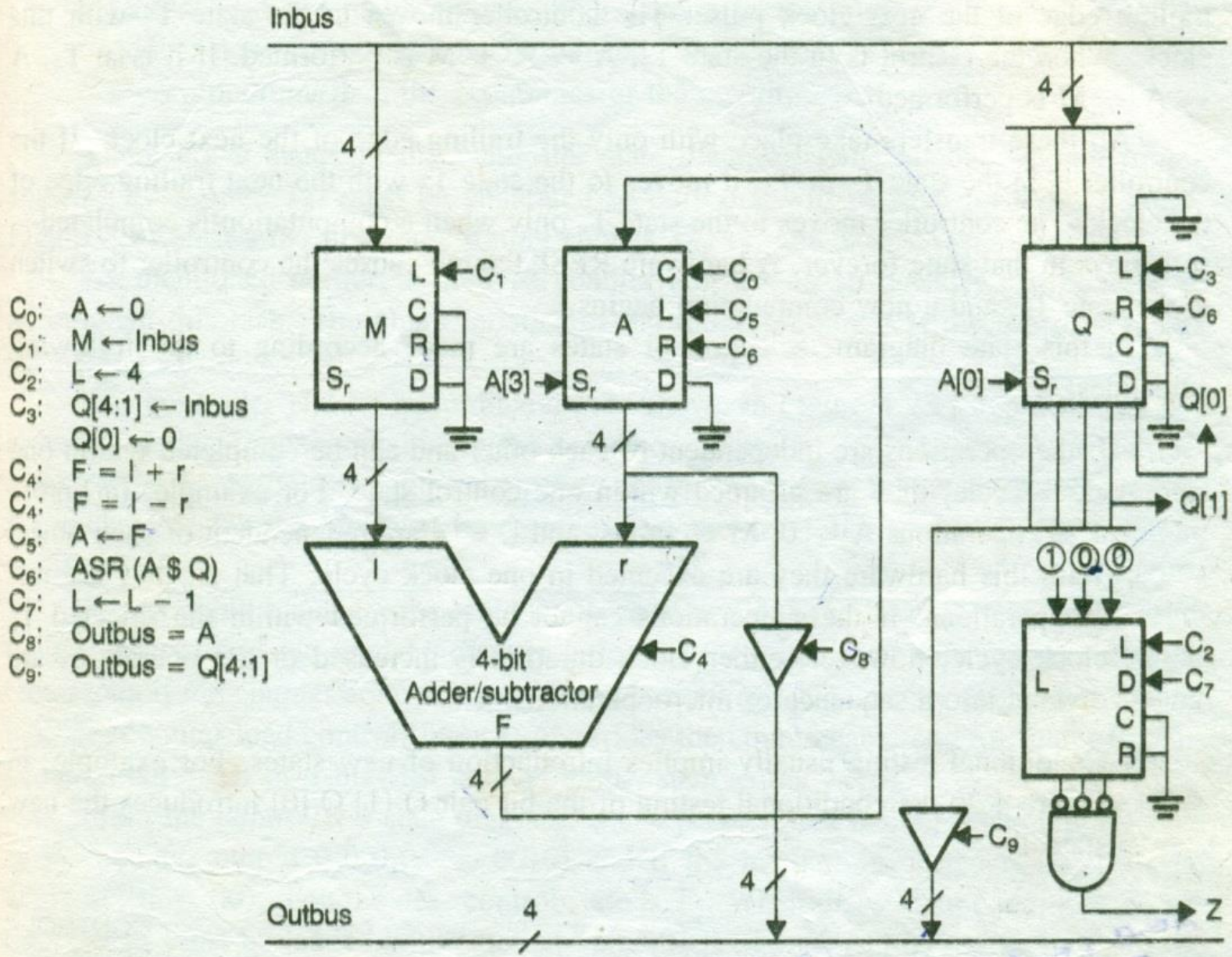


Figure 4.18 Processing Section of the Booth's Multiplier

STEP 7

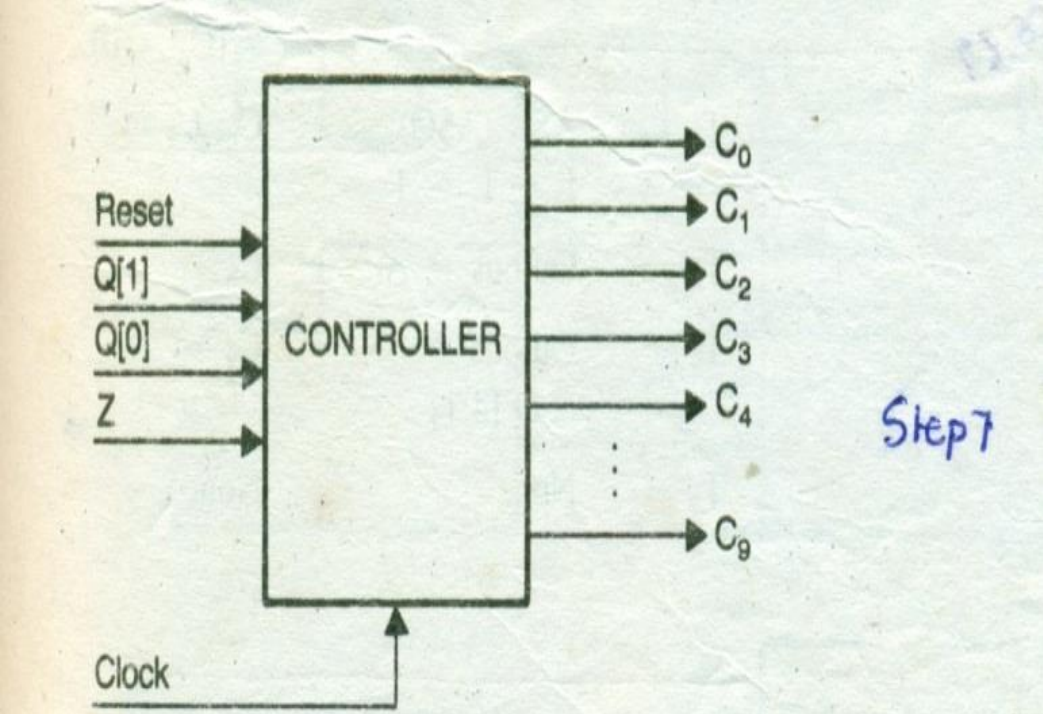
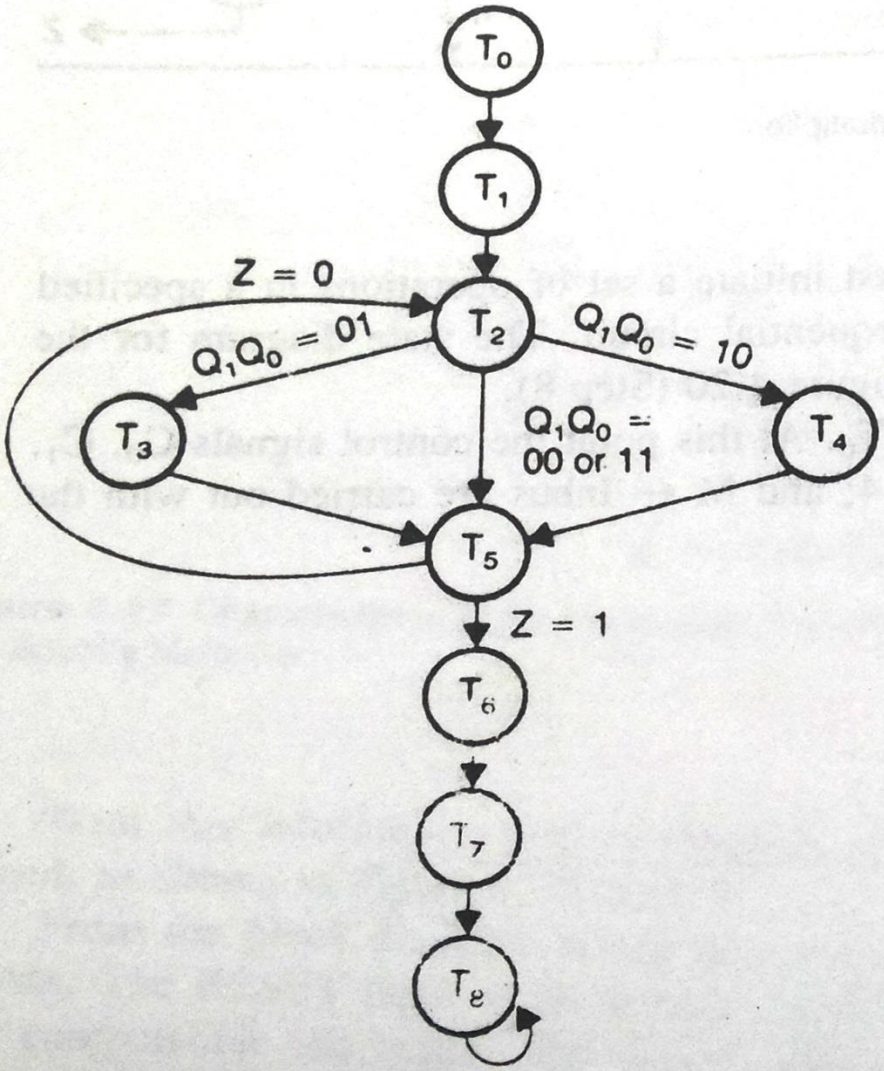


Figure 4.19 Block-diagram of the Booth's Multiplier Controller

STEP 8



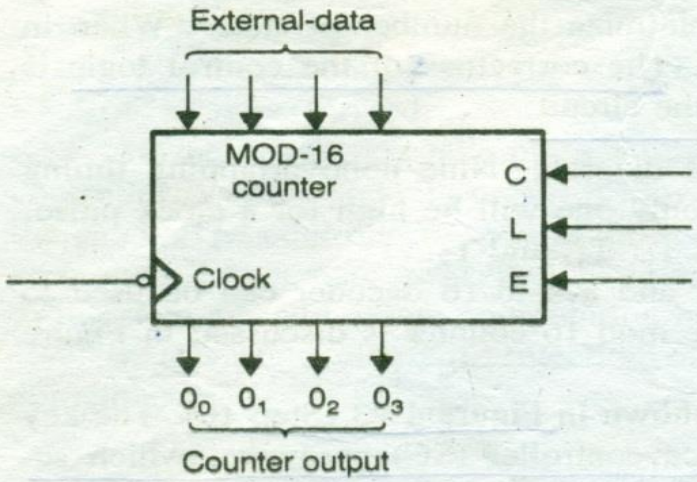
a. State Diagram

CONTROL STATE	OPERATION PERFORMED	CONTROL SIGNALS TO BE ACTIVATED
T_0	A ← 0, L ← 4, M ← Inbus	C_0, C_1, C_2
T_1	Q [4:1] ← Inbus, Q [0] ← 0	C_3
T_2	None	None
T_3	A ← A + M	C_4, C_5
T_4	A ← A − M	C_5 (C_4 = 0)
T_5	ASR (A$Q), L ← L − 1	C_6, C_7
T_6	Outbus = A	C_8
T_7	Outbus = Q [4:1]	C_9
T_8	None	None

b. Controller Action

Figure 4.20 Controller Description

STEP 9



a. Block Diagram

C	L	E	Clock	Action
1	X	X	X	Clear
0	1	X	↓	Load external data
0	0	1	↓	Count up
0	0	0	↓	No operation

b. Function Table

Figure 4.22 Characteristics of the Counter Used in the Controller Design

STEP 10

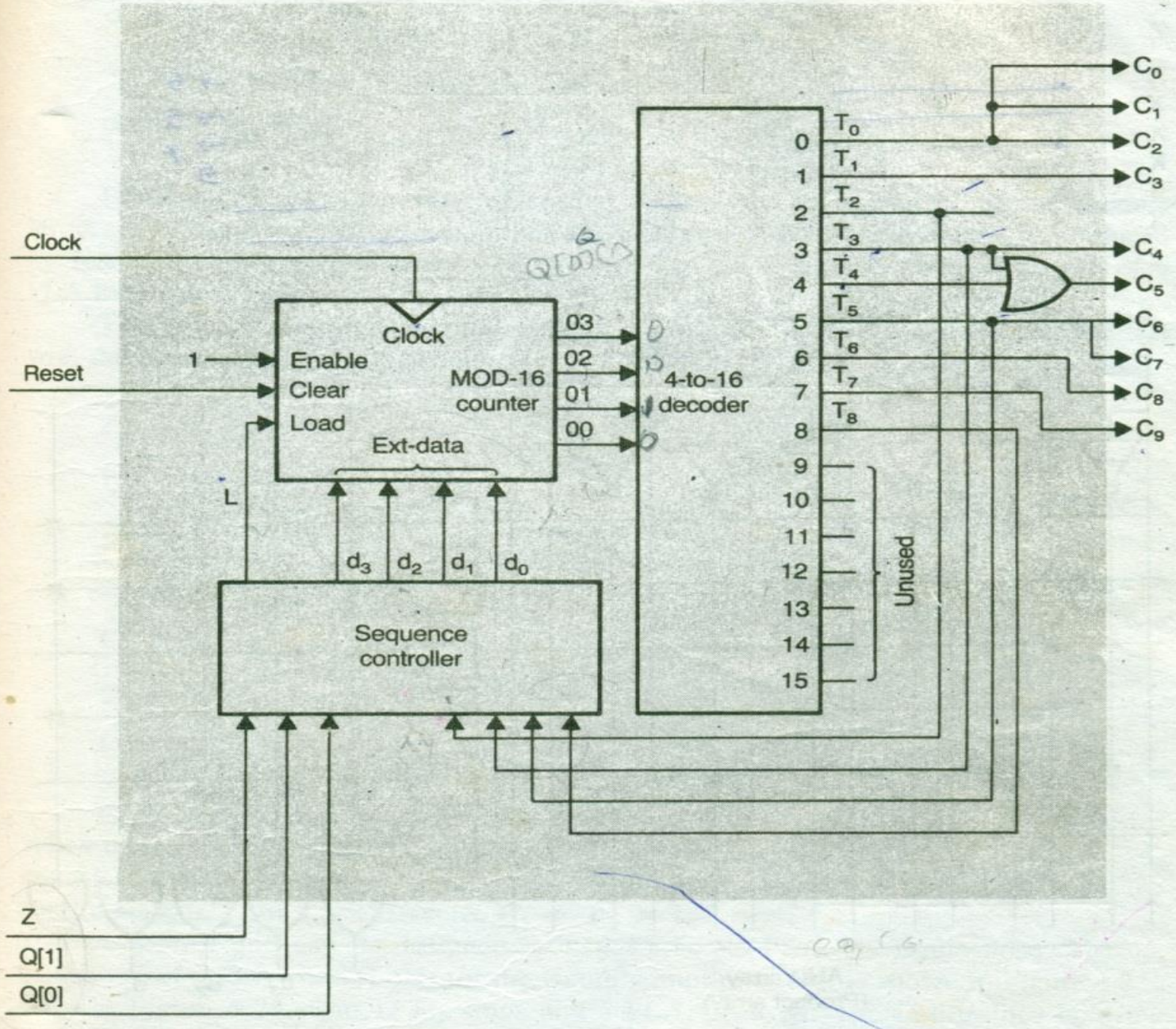
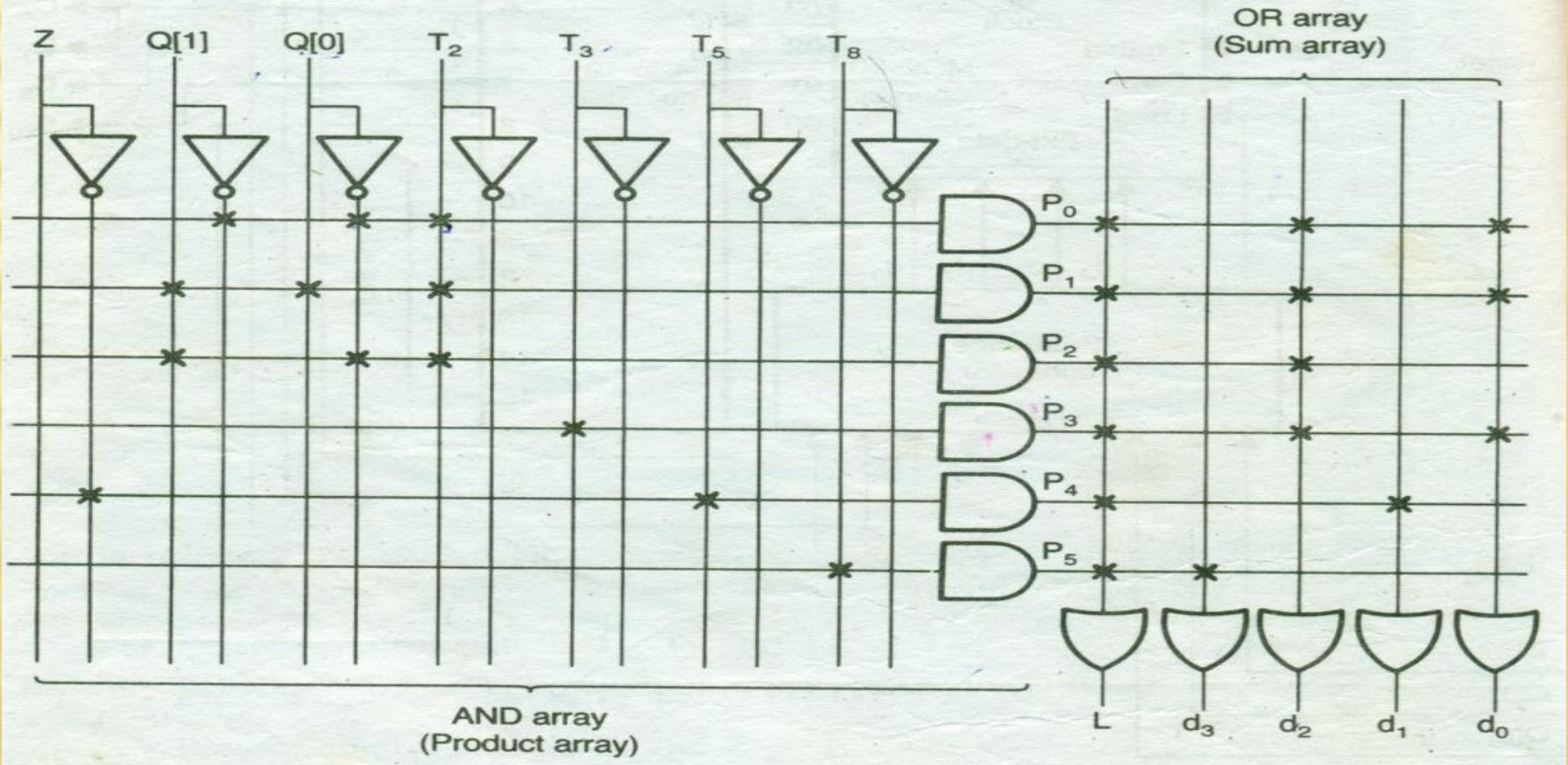


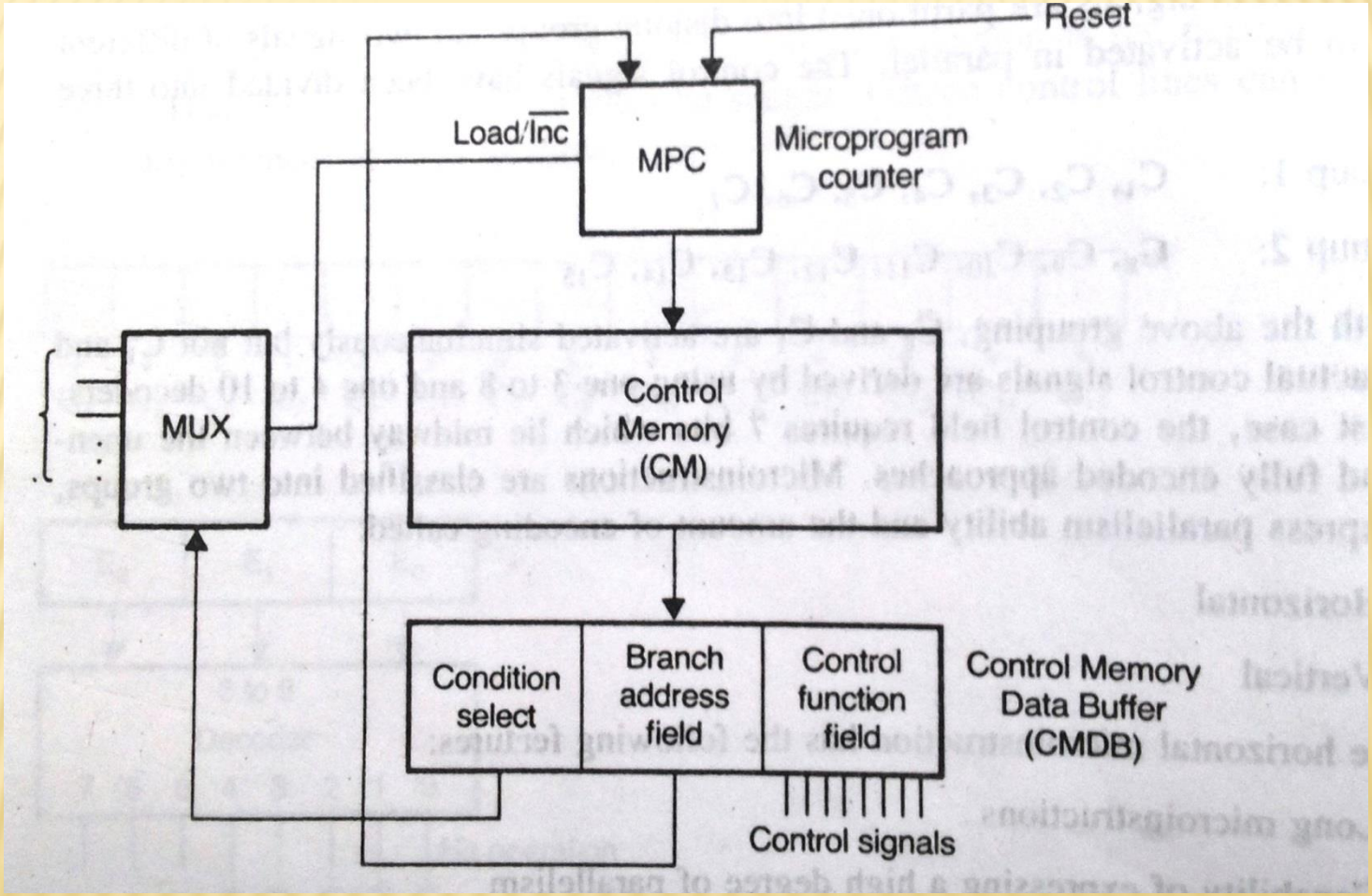
Figure 4.23 Logic Diagram of the Booth's Multiplier Controller

Z	Q [1]	Q [0]	T_2	T_3	T_5	T_8	L	External-data d3	d2	d1	d0	
X	0	0	1	X	X	X	1	0	1	0	1	→ 5
X	1	1	1	X	X	X	1	0	1	0	1	→ 5
X	1	0	1	X	X	X	1	0	1	0	0	→ 4
X	X	X	X	1	X	X	1	0	1	0	1	5
0	X	X	X	X	1	X	1	0	0	1	0	
X	X	X	X	X	X	1	1	1	0	0	0	

a. Truth Table



MICROPROGRAMMED CONTROL UNIT



Control Memory Address	Declare register A[4], M[4],Q[5], L[3]; Declare buses Inbus[4], Outbus[4];
0 0000	Start: A←0, M←Inbus, L←4; C0, C1, C2
1 0001	Q[4:1]←Inbus, Q[0]←0; C3
2 0010	Loop: If Q[1:0]=01 then Goto Add;
3 0011	If Q[1:0]=10 then Goto sub;
4 0100	Go to Rshift
5 0101	Add: A←A+M;C4,C5
6 0110	Go to Rshift
7 0111	Sub: A←A-M; C4', C5
8 1000	Rshift: ASR(AQ), L←L-1; C6,C7
9 1001	If L<>0 then Goto loop
101010	Outbus=A; C8
111011	Outbus=Q[4:1]; C9
121100	Halt: Go to Halt;

CONDITION-SELECT FIELD	INTERPRETATION
000	No branching
001	Branch if Q [1] = 0 and Q [0] = 1
010	Branch if Q [1] = 1 and Q [0] = 0
011	Branch if Z = 0
100	Unconditional branching

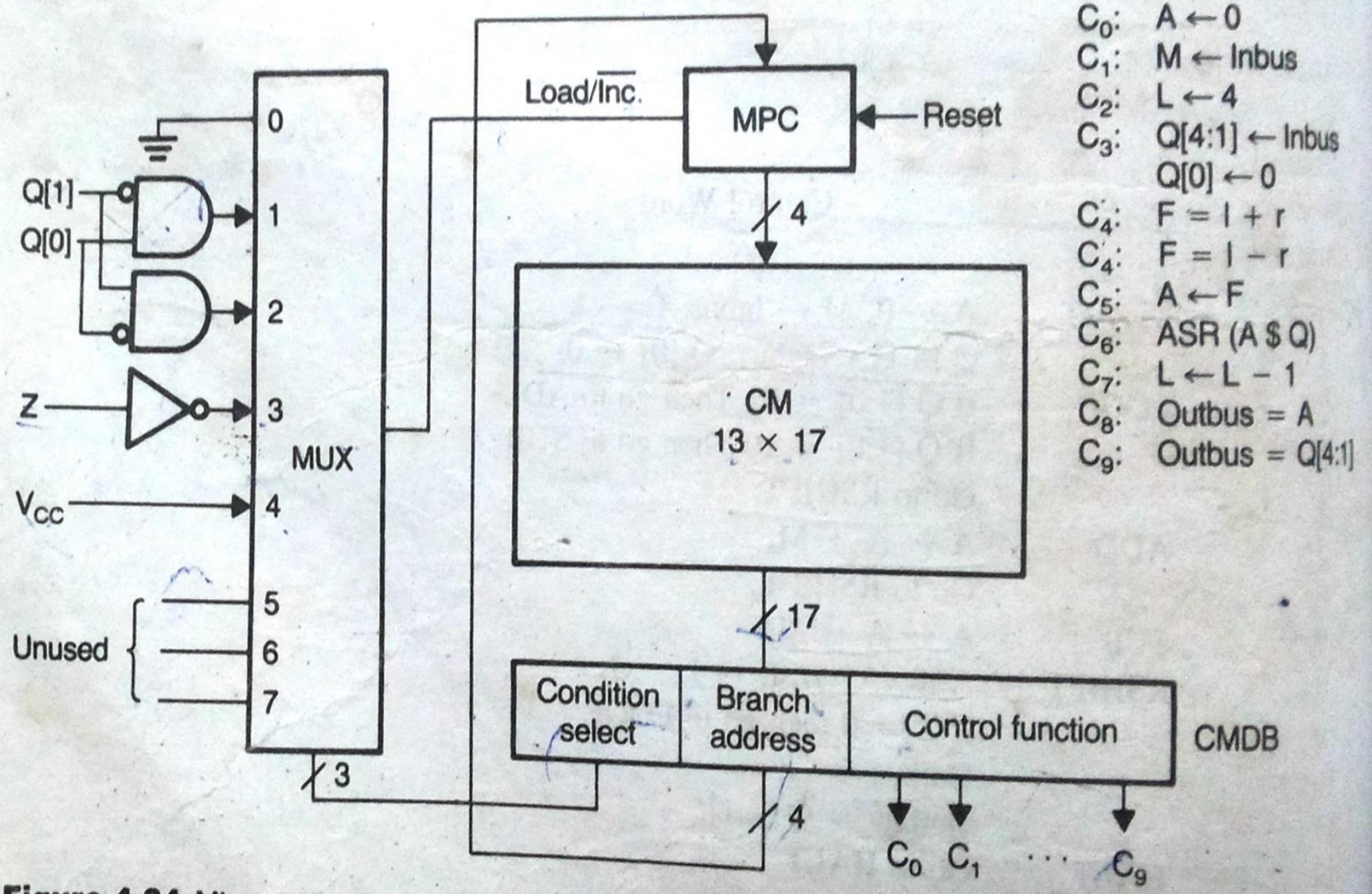


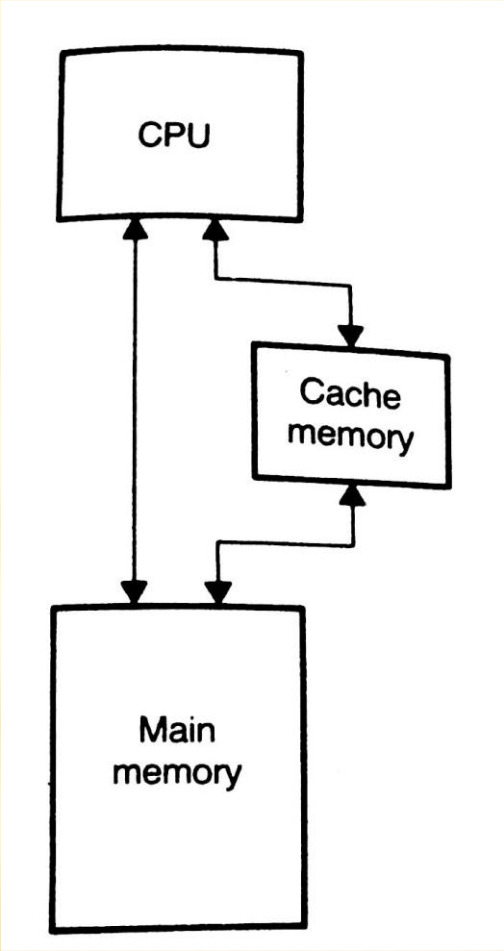
Figure 4.34 Microprogrammed Multiplier Control Unit

ROM Address																					
In decimal	In binary				Cond. Sel			Branch Addr.				Control Word									
												Control Function									
					c_{s2}	c_{s1}	c_{s0}	b_{r3}	b_{r2}	b_{r1}	b_{r0}	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	Comments
0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	A ← 0, M ← Inbus, L ← 4
1	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	Q [4:1] ← Inbus, Q [0] ← 0
2	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	If Q [1:0] = 01 Then go to 5 (ADD)
3	0	0	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	If Q [1:0] = 10 Then go to 7 (SUB)
4	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	go to 8 (RSHIFT)
5	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	A ← A + M
6	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	go to 8 (RSHIFT)
7	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	A ← A − M
8	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	ASR (ASQ), L ← L − 1
9	1	0	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	If Z = 0 Then go to 2
10	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	Outbus = A
11	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Outbus = Q [4:1]
12	1	1	0	0	1	0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	Go to 12 (HALT)

ROM Address					Control Word																	
In decimal	In binary				Control Function																	Comments
					Cond. Sel			Branch Addr.														
					c_{s2}	c_{s1}	c_{s0}	b_{r3}	b_{r2}	b_{r1}	b_{r0}	c_0	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9	
0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	A ← 0, M ← Inbus, L ← 4
1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	Q [4:1] ← Inbus, Q [0] ← 0
2	0	0	1	0	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	If Q [1:0] = 01 Then go to 5 (ADD)
3	0	0	1	1	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	If Q [1:0] = 10 Then go to 7 (SUB)
4	0	1	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	go to 8 (RSHIFT)
5	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	A ← A + M
6	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	go to 8 (RSHIFT)
7	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	A ← A − M
8	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	ASR (A$Q), L ← L − 1
9	1	0	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	If Z = 0 Then go to 2
10	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	Outbus = A
11	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	Outbus = Q [4:1]
12	1	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	Go to 12 (HALT)

Figure 4.35 Binary Listing of the Microprogram For the 4 x 4 Booth's Multiplier

CACHE MEMORY

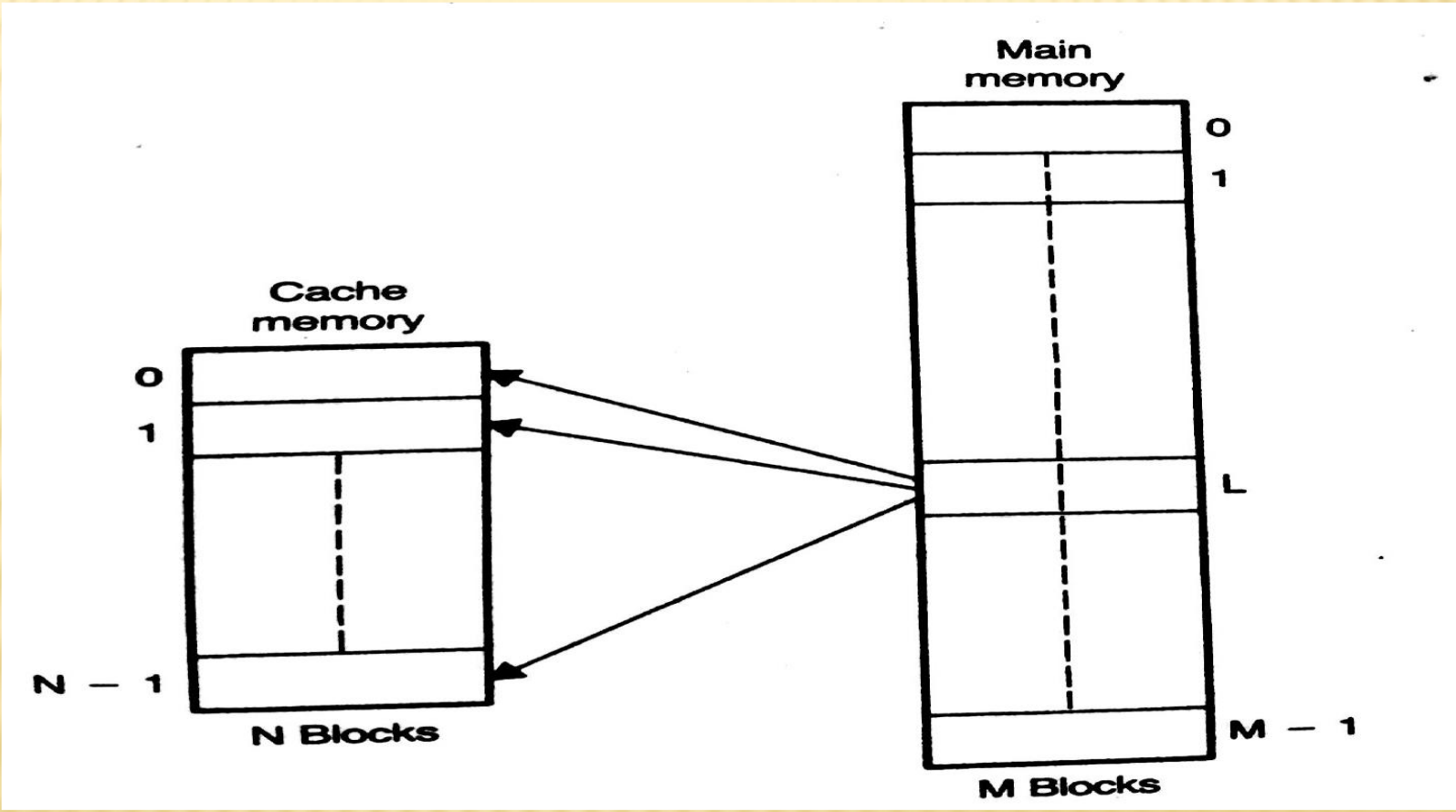


CACHE MAPPING

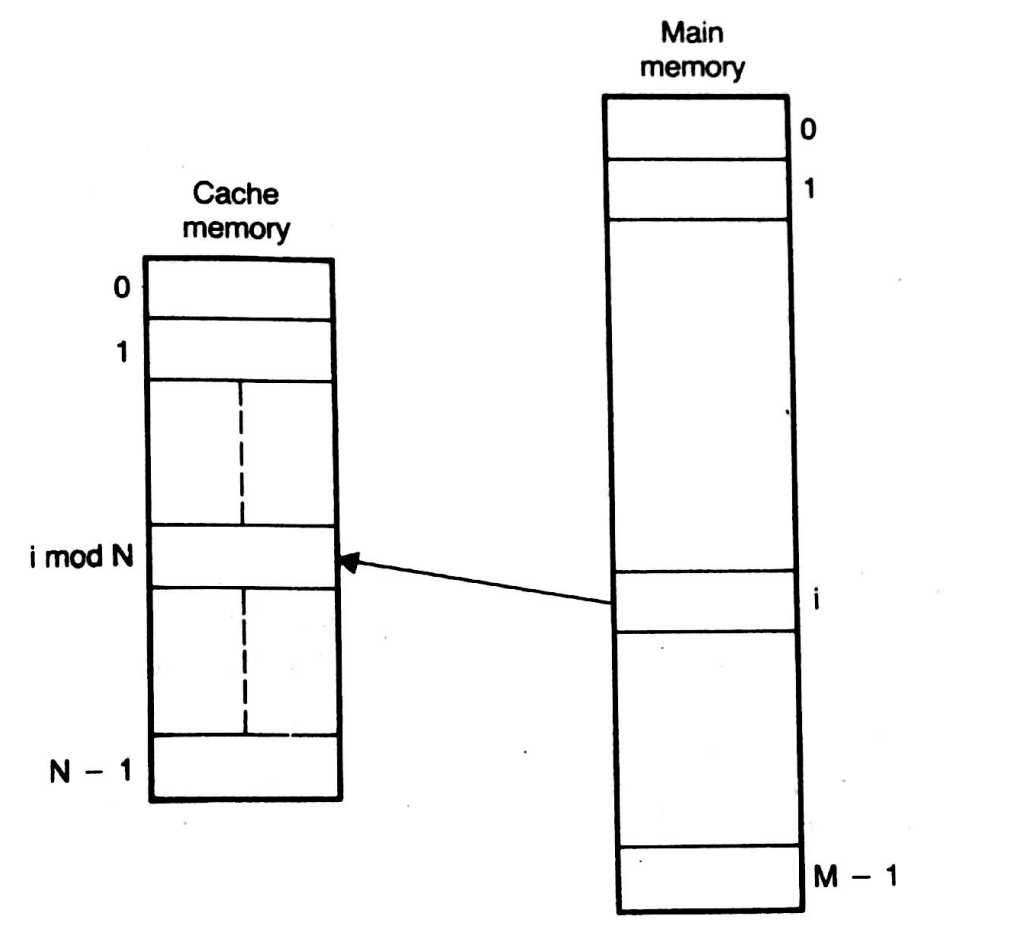
Mapping techniques:

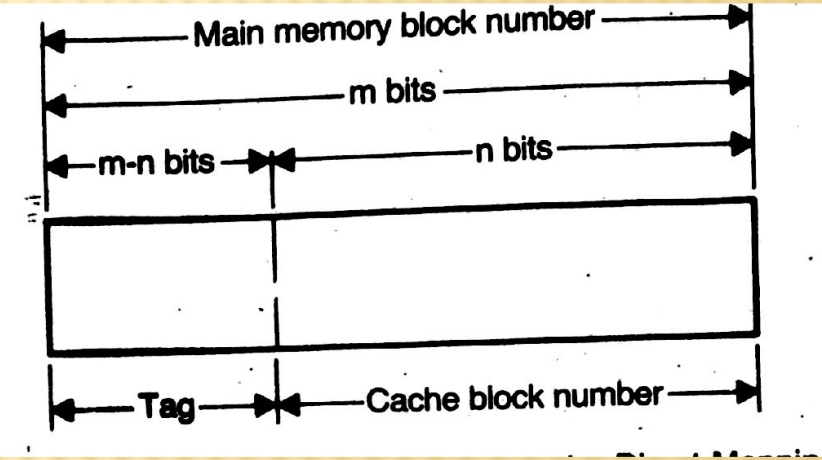
- Fully Associative Mapping
- Direct-Mapping
- Set-Associative Mapping

FULLY ASSOCIATIVE MAPPING

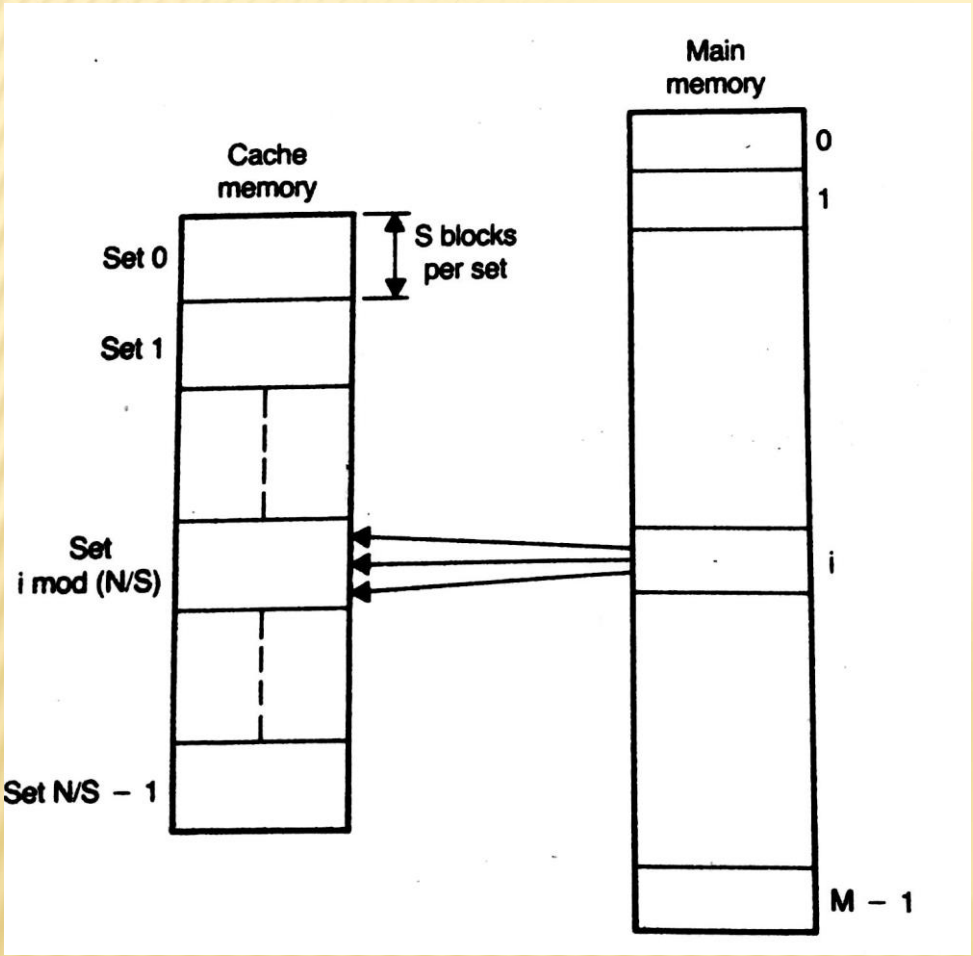


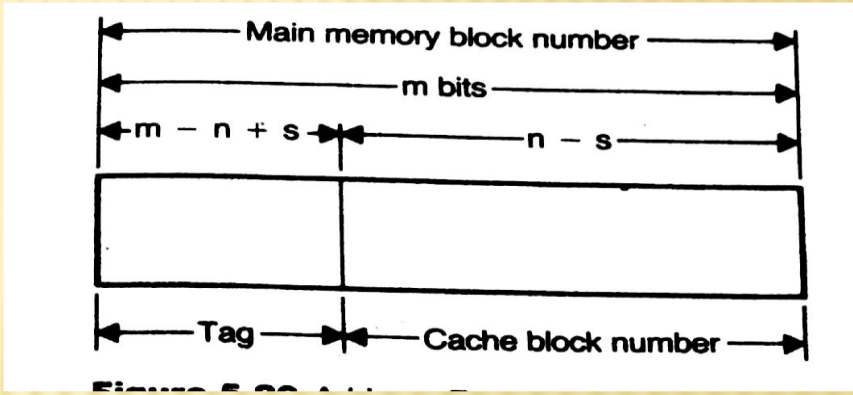
DIRECT MAPPING





SET ASSOCIATIVE MAPPING





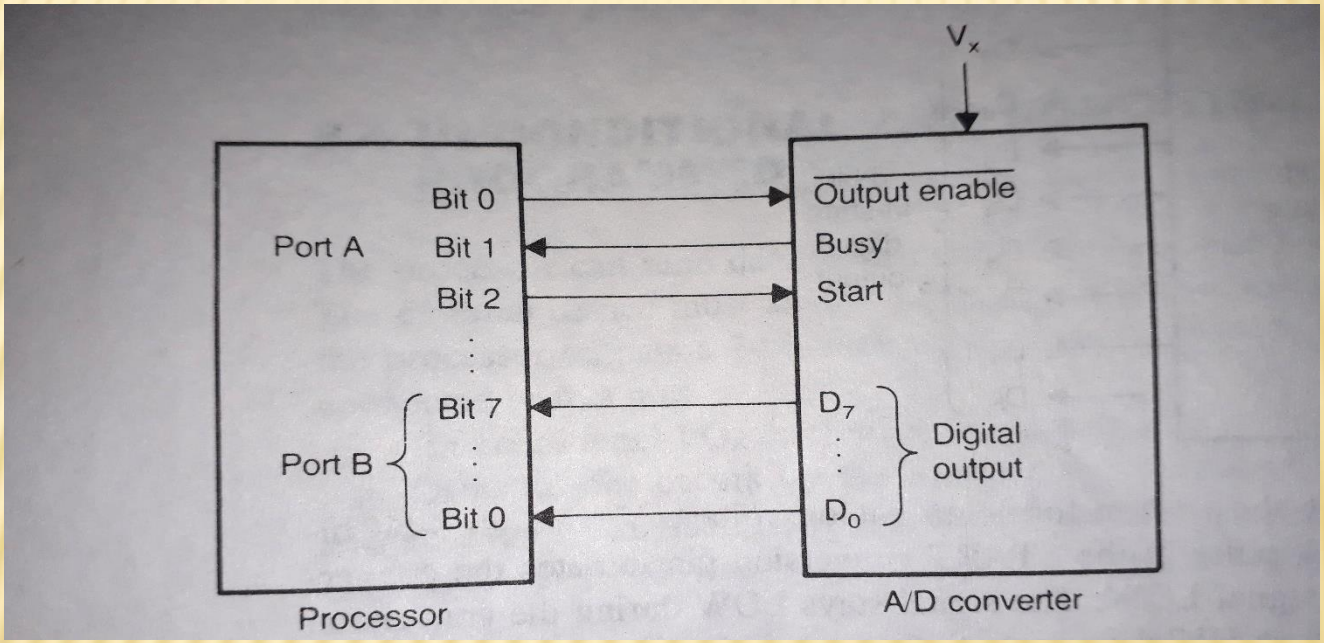
There are three ways of transferring data between the computer and a physical I/O device:

- Programmed I/O
- Interrupt I/O
- Direct memory access (DMA)

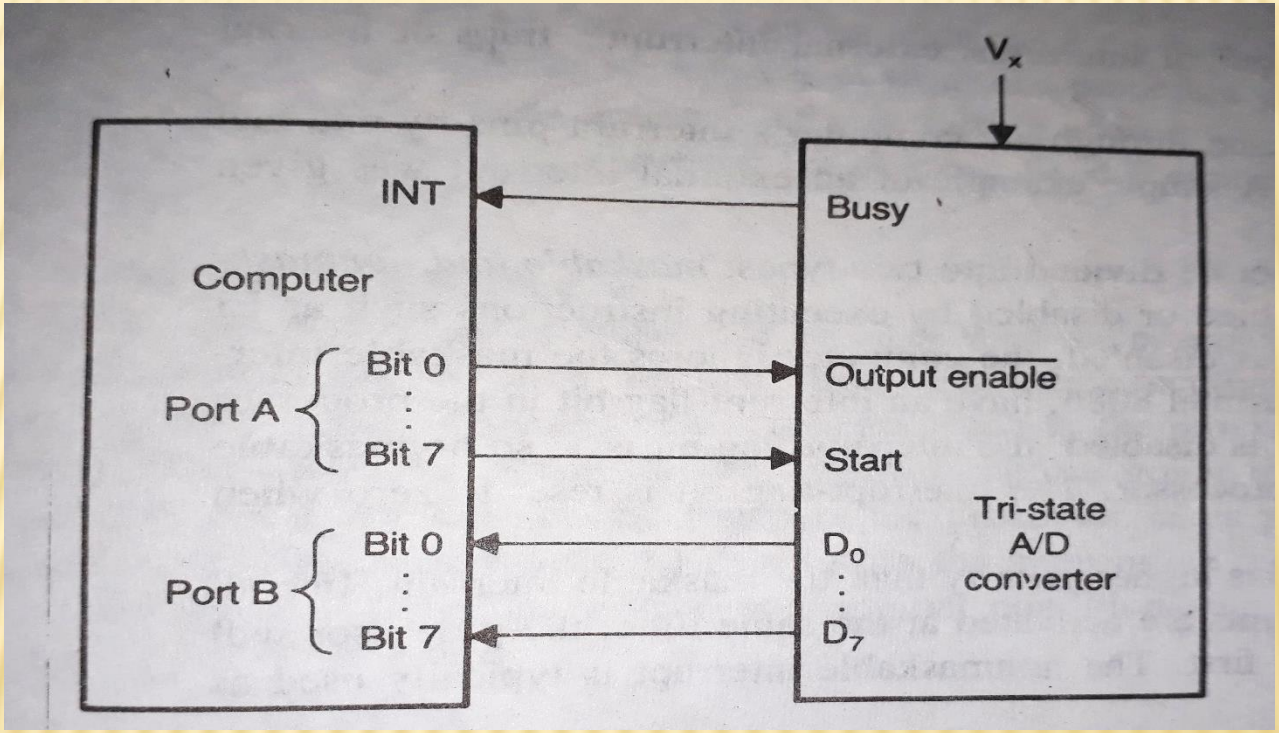
Different techniques of addressing I/O ports:

- Standard I/O (isolated I/O)
- Memory mapped I/O

PROGRAMMED I/O



INTERRUPT I/O



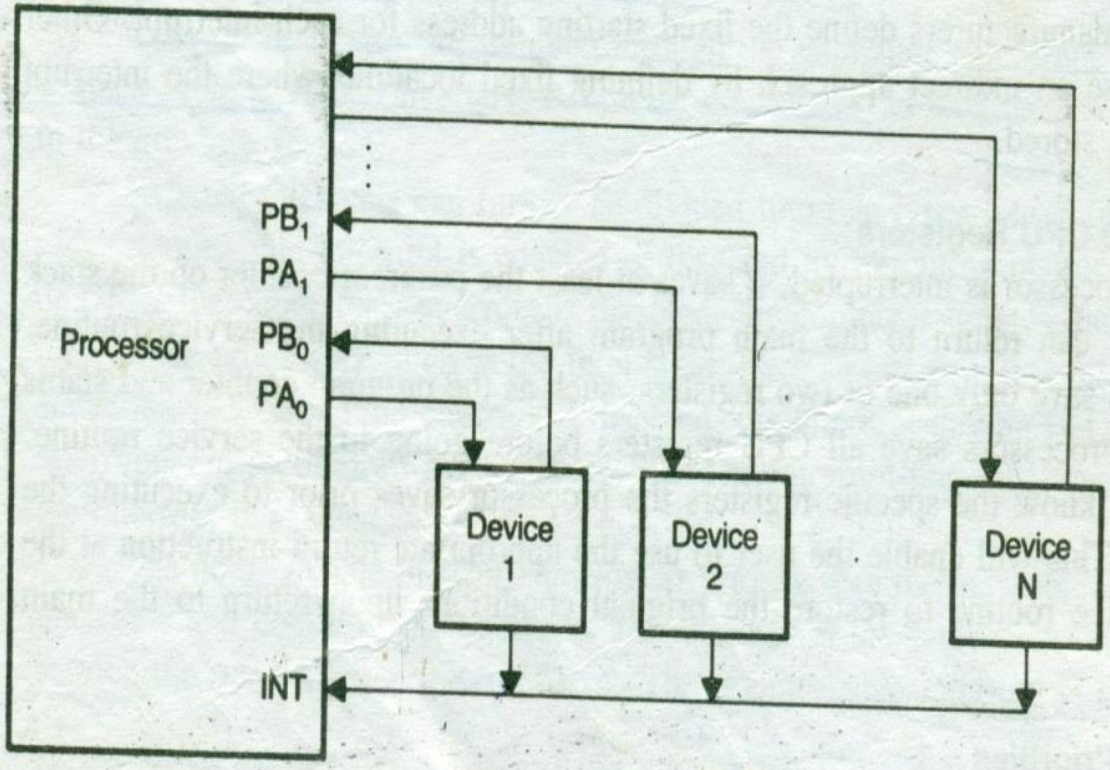


Figure 6.4 Polled Interrupt

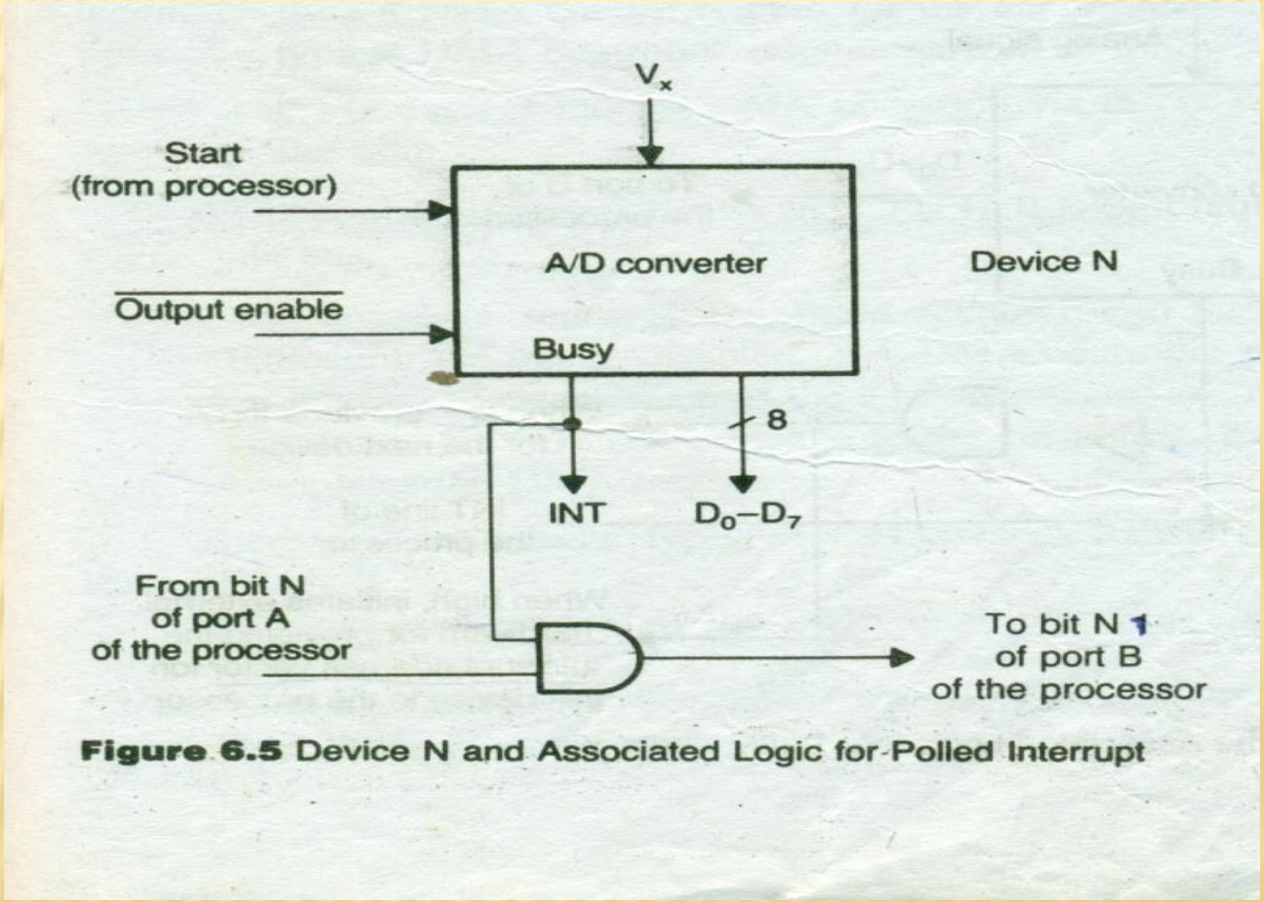


Figure 6.5 Device N and Associated Logic for Polled Interrupt

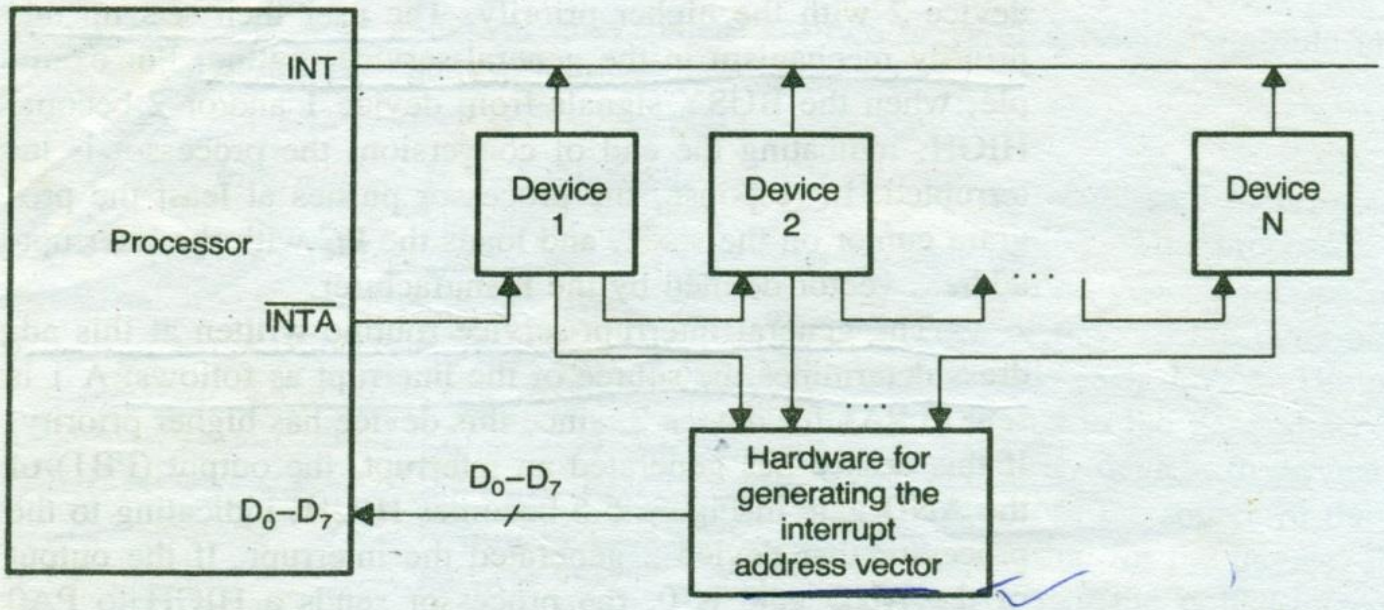


Figure 6.6 Daisy Chain Interrupt

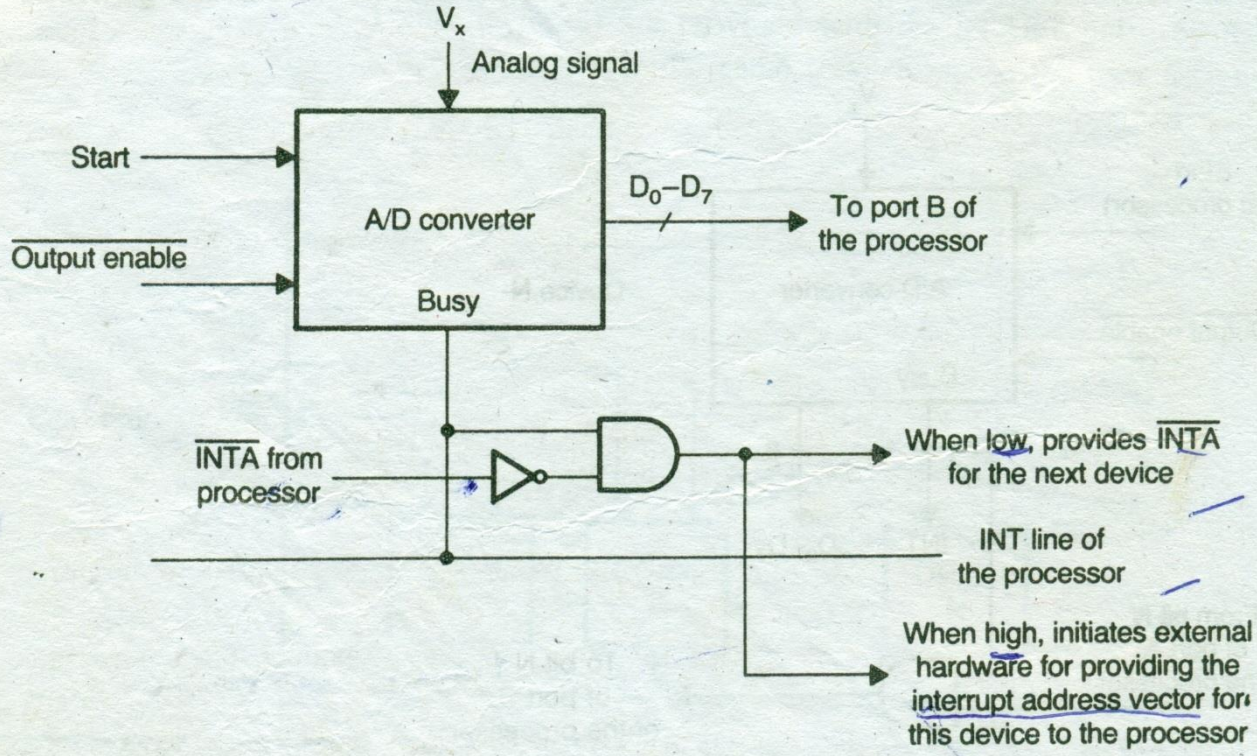
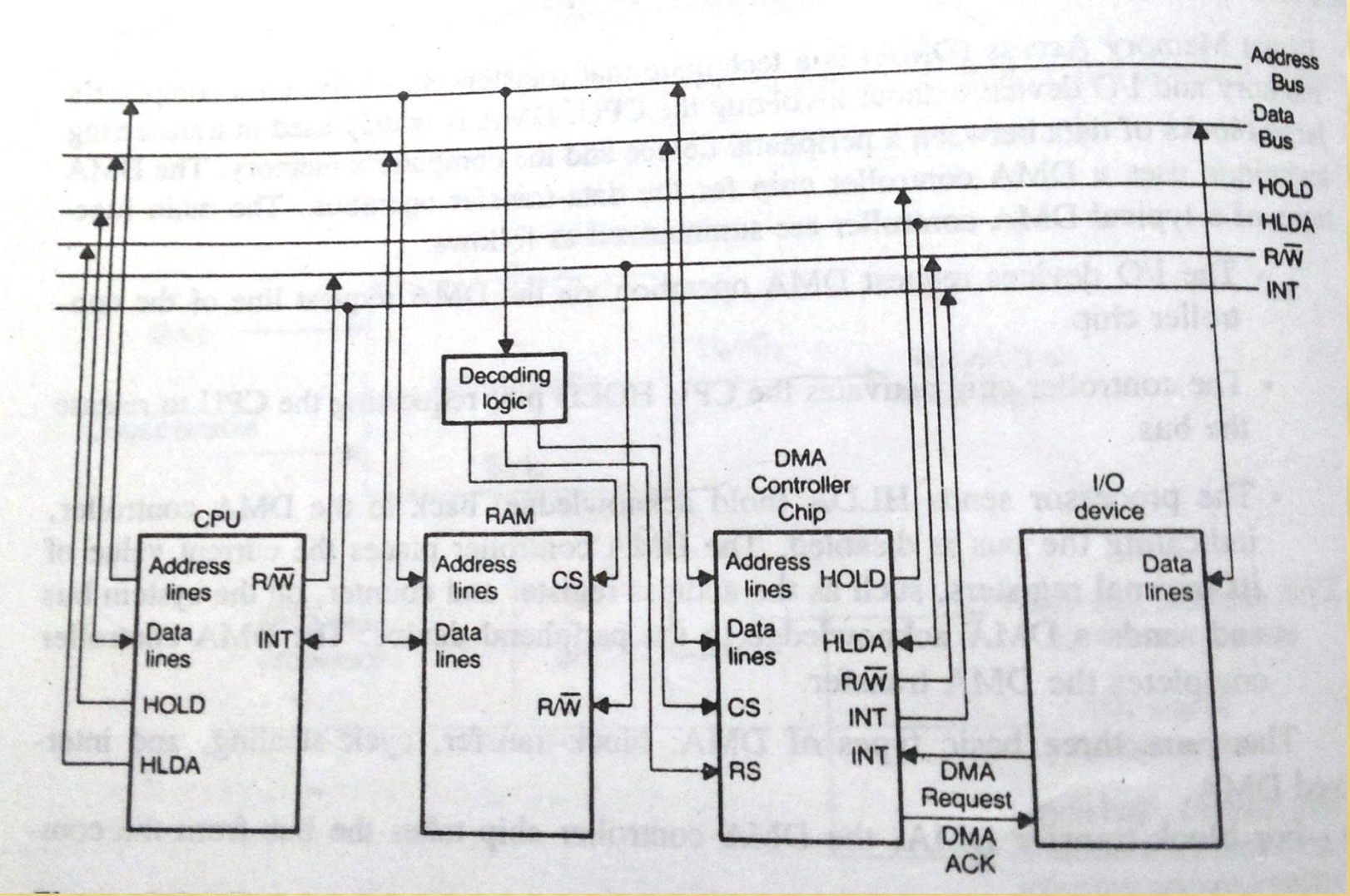


Figure 6.7 Each device and the Associated Logic in a Daisy Chain

DIRECT MEMORY ACCESS(DMA)

- Cycle Stealing
- Interleaved
- Block Transfer


Address Bus
Data Bus
HOLD
HLDA
R/W
INT
Decoding logic
CPU
Address lines
R/W
Data lines
INT
HOLD
HLDA
RAM
Address lines
CS
Data lines
R/W
DMA Controller Chip
Address lines
HOLD
Data lines
HLDA
R/W
CS
INT
RS
INT
I/O device
Data lines
DMA Request
DMA ACK