COMPUTER ORGANIZATION AND ARCHITECTURE

Course Code	: CSE 2151
Credits	: 04

CONTROL UNIT

MODULE 4

CONTROL UNIT

- **Basic concepts**
  - Fundamentals of Control Unit
  - Register transfer notations and descriptions
  - Buses

- **Design methods**
  - Hardwired approach
  - Microprogramming

INTRODUCTION

- CPU is viewed as a collection of two major components:
    - Processing section
    - Control Unit
- Control unit's responsibility is to drive the associated processing hardware by generating a set of signals that are synchronized with the master clock.
- In order to carry out a task, the CU must generate a set of control signals in a predefined sequence governed by the hardware structure of the processing section.

INTRODUCTION (CONTD.)

- Inputs to CU are:
  - Master clock
  - Status information from processing section
  - Command signals from external agent (like RESET, ABORT)
- Outputs produced by CU
  - Signals that drive the processing section and responses to an external environment.
- Control unit undertakes the following responsibilities:
  - **Instruction interpretation:** (CU read instructions, recognizes the instruction type, gets operands and route to appropriate functional units of Processing Unit (PU), necessary control signals are then issued to the PU to perform desired operation)
  - **Instruction sequencing:** CU determines the address of next instruction to be executed and loads it on to PC.

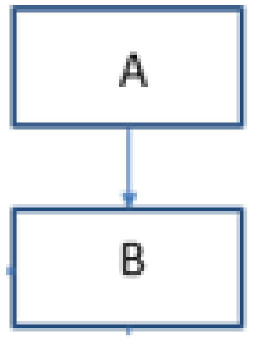
BASIC CONCEPTS

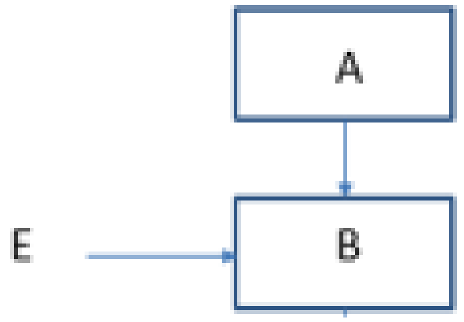
- Basis for CU design are register transfer operations
  1. 8-bit info moved from Register A to Register B.
     - Such operation is described as B←A
     - Declaring registers: Declare registers A[8], B[8], PC[16] ;
  2. Register can be defined as a portion of some other register.
     - Assigning higher order byte of 16-bit PC: Declare subregisters PCHI[8]= PC[15-8] ;
  3. Assigning individual bits
     - B[0]=A[7] means MSB of A is copied to LSB of B.
  4. Normally two inputs are associated with each register:
     1. Enable input (E) or control input controls the data flow from A to B
     2. Data input

     Register B is loaded with A only when E is held high else contents of register remain the same.

     Such conditional transfer is expressed as

     E: B←A



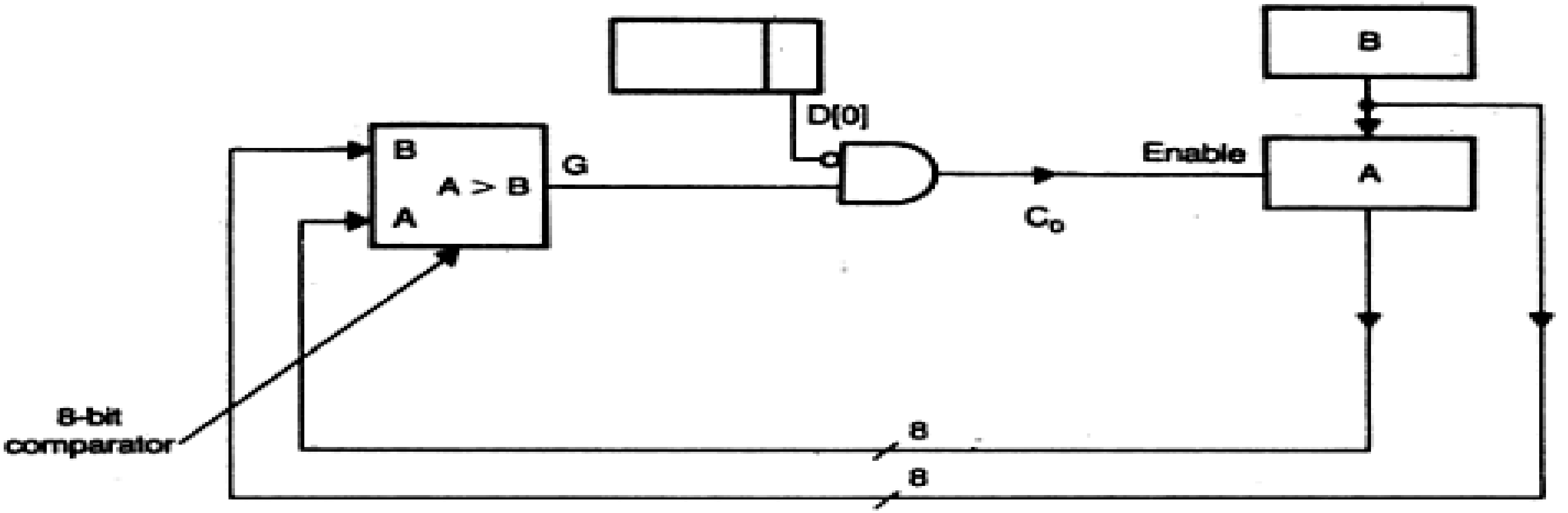


BASIC CONCEPTS (CONTD.)

5. Control input can be a function of more than one variable.

    IF A>B and D[0]=0 THEN A←B

- Comparator : if A>B, the output G from the comparator is set to high
- Conditional transfer: C_0: A←B ; where $C_0 = G \wedge D[0]'$



BASIC CONCEPTS (CONTD.)

6. To perform register transfer operation that involves selection.

   If x=0 and t=1, then

   $A \leftarrow B$

   else

   $A \leftarrow D$

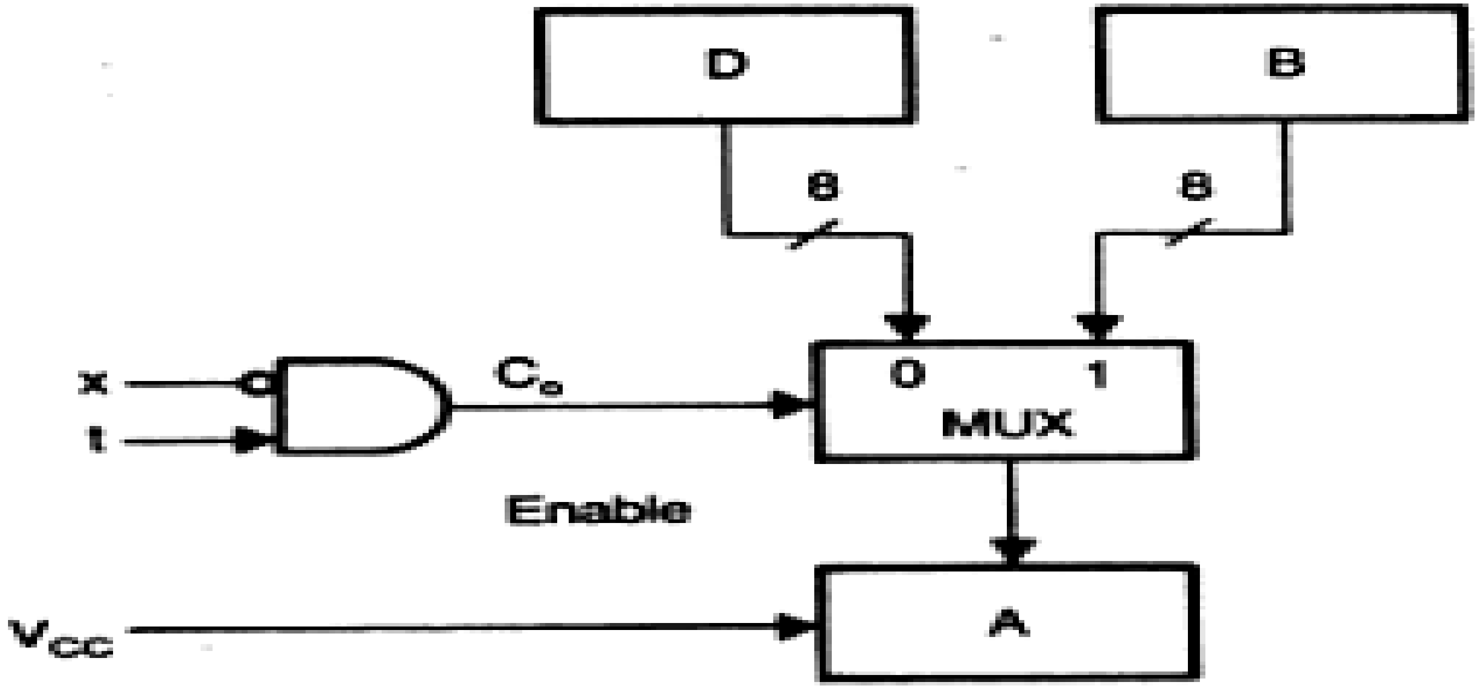
Such transfer is expressed as

C_0 : $A \leftarrow B$;

C_0' : $A \leftarrow D$;

Where $C_0 = x't$ and

$C_0' = (x't)' = x + t'$



BASIC CONCEPTS (CONTD.)

- The other register transfer operations are
- $D \leftarrow A'$; Transfer the complement of A to D.
- $A \leftarrow A+1$; Increment the content of A by 1.
- $A \leftarrow A-1$; Decrement the content of A by 1.
- $D \leftarrow A \vee B$; A OR B, store result in D
- $D \leftarrow A \wedge B$; A AND B, store result in D
- LSR(A); Logical shift right
- ASR(A); Arithmetic shift right
- LSL, ASL, ROR, ROL
- A$Q – used to concatenate A and Q
  - ASR(A$Q) ;

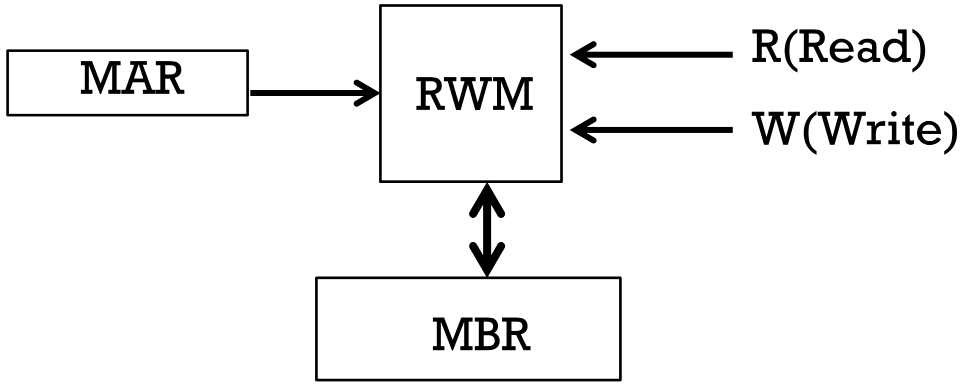
BASIC CONCEPTS (CONTD.)

- Whenever RWM is a part of processing section , MBR and MAR are associated with RWM unit.

MAR holds the address of desired memory word and MBR as buffer register in all data transfer operations.

- R: MBR← M((MAR))
- W: M((MAR))← MBR

The line b/n RWM and MBR is bidirectional bus, and it can be easily implemented using tristate buffers.



BASIC CONCEPTS (CONTD.)

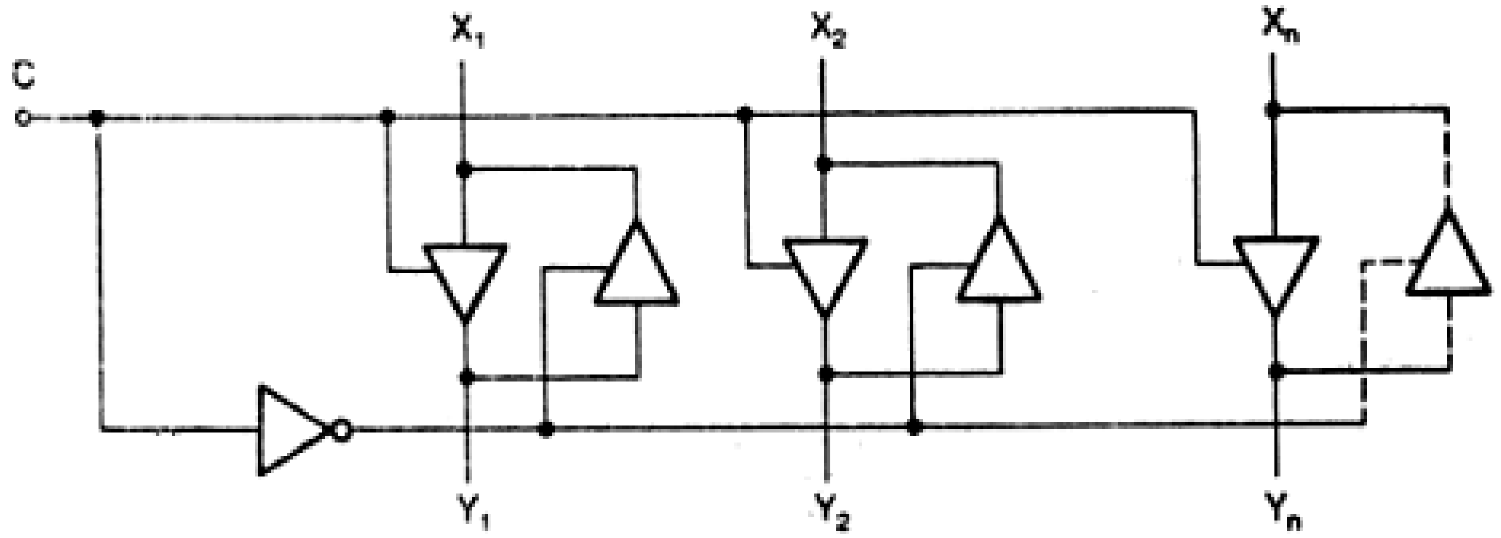


Fig: Bidirectional Data bus

- When C=1, X to Y
- When C=0, Y to X

BASIC CONCEPTS (CONTD.)

i. Declare buses Inbus[4] and outbus[4]; //4-bit buses
ii. A=inbus; // data of inbus is transferred to Reg A when next clock arrives
iii. Outbus = B[7:4] //Higher order 4 bits of 8-bit register B are made available on the outbus for one clock period.

Rate at which computer performs operations (such as $A \leftarrow A+M$, $A \leftarrow A \wedge B$) is determined by bus structure.

BUSES

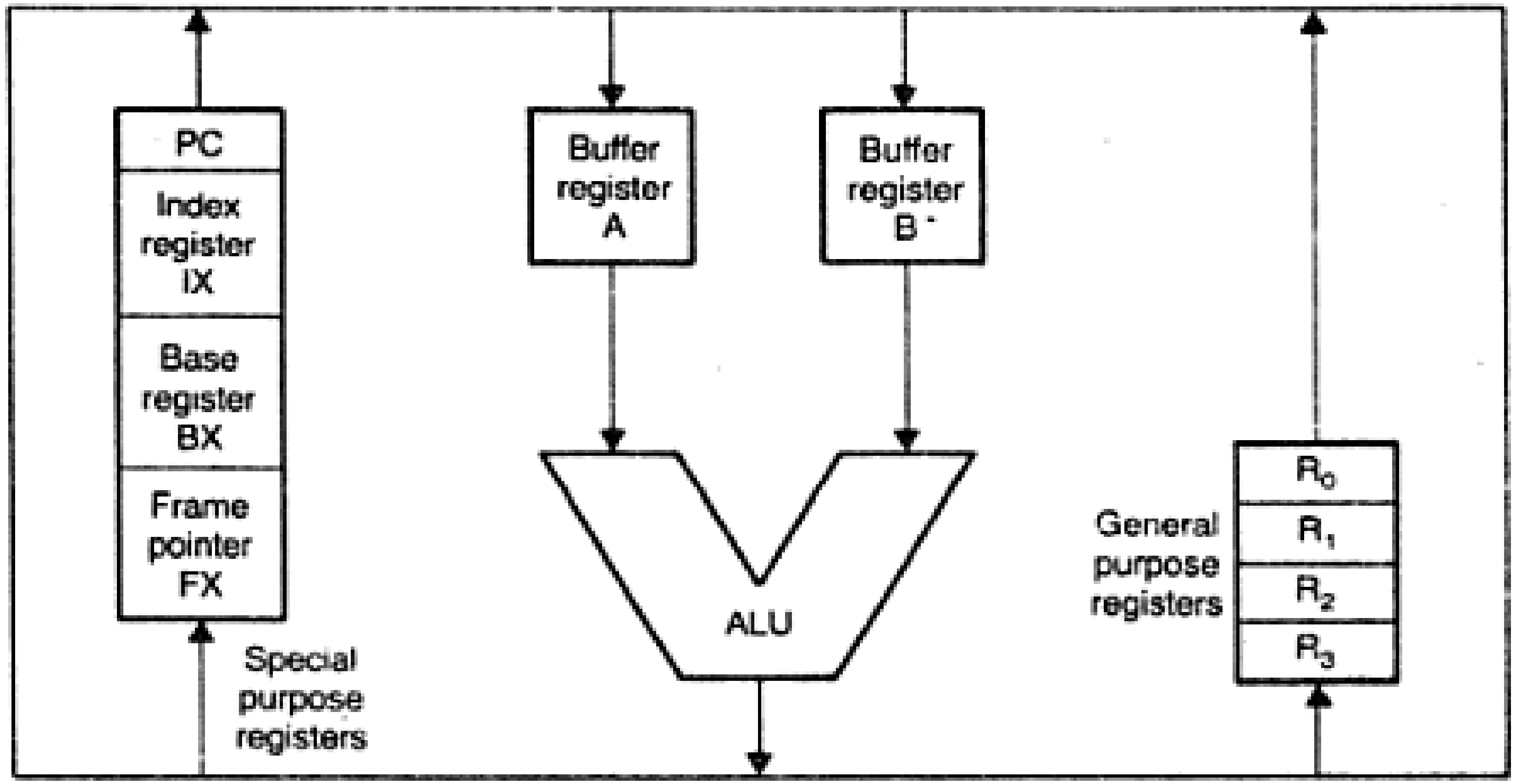
- Several types of bus structure within the CPU:
  i) Single-bus oriented ALU
  ii) Two-bus oriented ALU
  iii) Three-bus oriented ALU

BUSES

- Single-bus oriented ALU

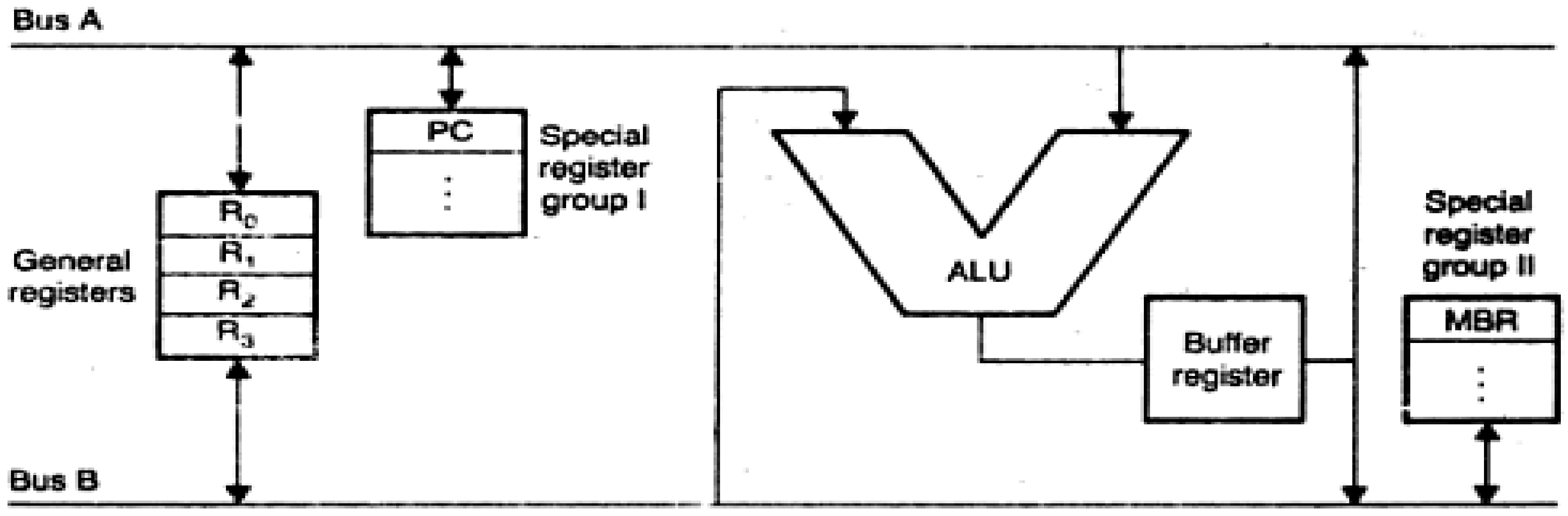
Disadvantages:

- Affects speed of execution of a typical 2 operand memory
- Increases the number of states in control logic. Hence more HW may be required to design control unit



BUSES

- Two-bus oriented ALU

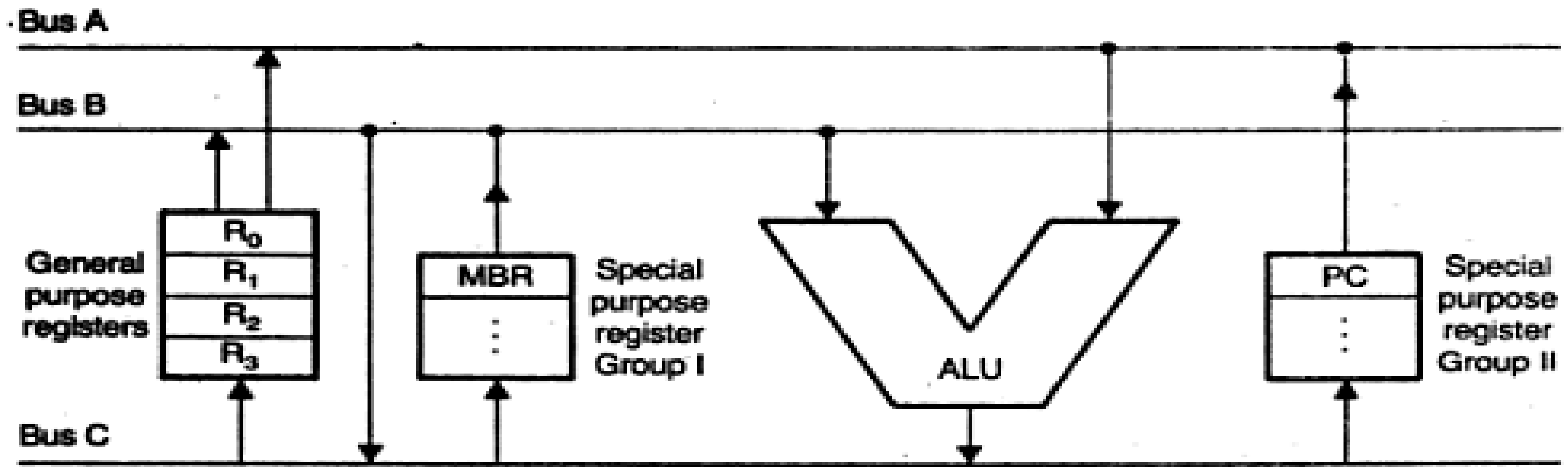


BUSES

- Two-bus oriented ALU
- Output Buffer register is used to prevent collision of the buses
- 1st cycle: loading operands and storing result in O/P buffer
- 2nd cycle: result in O/P buffer is pushed to bus(destination). The contents of buffer register can be gated to either bus A or bus B.

BUSES

- Three-bus oriented ALU


Bus A
Bus B
Bus C
General purpose registers
R_0
R_1
R_2
R_3
MBR
Special purpose register Group I
ALU
PC
Special purpose register Group II

TOPICS COVERED FROM

- Textbook 3:
  - Chapter 4: 4.1, 4.2

COMPUTER ORGANIZATION AND ARCHITECTURE

Course Code : CSE 2151

Credits : 04

HARDWIRED APPROACH

- Control logic is a clocked sequential circuit.
- So conventional sequential circuit design procedure can be applied to build CU.
- Final circuit is obtained by physically connecting gates and flip flops.
- Cost of control logic increases with system complexity.

10 STEPS FOR HARDWIRED CONTROL

1. Define task to be performed.
2. Propose a trial processing section.
3. Provide a register transfer description algorithm based on processing section outlined.
4. Validate the algorithm by using trial data.
5. Describe the basic characteristics of the hardware elements to be used in the processing section.
6. Complete the design of the processing section by establishing necessary control points.
7. Propose the block diagram of the controller.
8. Specify state diagram of controller.
9. Specify the characteristics of the hardware elements to be used in the controller.
10. Complete the controller design and draw a logic diagram of final circuit.

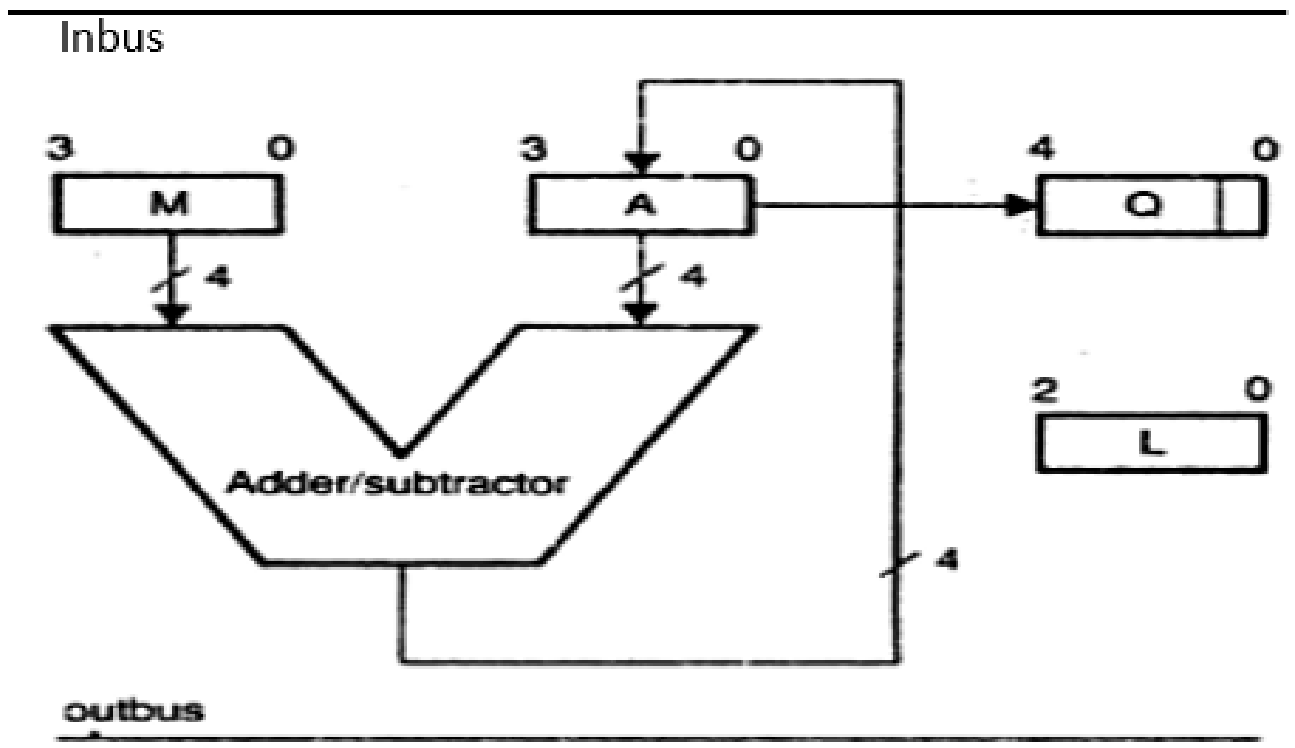
10 STEPS FOR HARDWIRED CONTROL

Step 1: Task definition.

Design a Booth's multiplier to multiply two 4-bit signed numbers.

Step 2: Trial processing section.

q_1	q_0
0	0→none
0	1→ add M
1	0→sub M
1	1→ None



10 STEPS FOR HARDWIRED CONTROL

Step 3: Register transfer description of Booth's multiplier procedure based on the processing section outlined in the previous step.

```
            Declare registers A[4], M[4], Q[5], L[3];
            Declare buses inbus[4], outbus[4];
Start:      A←0, M←inbus, L←4;                        clear A and transfer M
            Q[4:1]←inbus, Q[0]←0;                     transfer Q
Loop:       if Q[1:0]= 01, then go to ADD;
            if Q[1:0]=10, then  go to SUB;
            go to Rshift;
ADD:        A←A+M;
            goto Rshift;
SUB:        A←A-M;
Rshift:     ASR(A$Q), L←L-1;
            if L>0, then go to Loop
            outbus =A;
            outbus=Q[4:1];
 Halt:      go to Halt
```

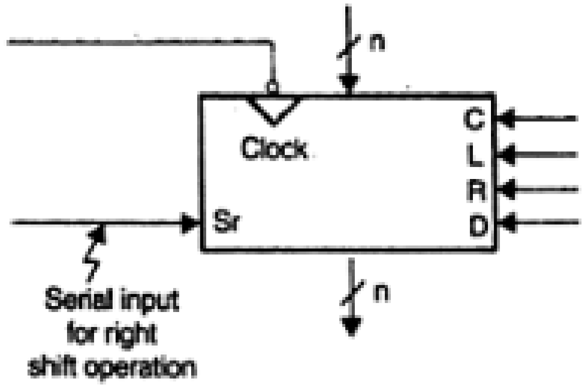
10 STEPS FOR HARDWIRED CONTROL

Step 4: Validate the algorithm by using trial data.

A	Q	Q_{-1}	M		
0000	0011	0	0111	Initial Values	
1001	0011	0	0111	A A - M	First Cycle
1100	1001	1	0111	Shift	
1110	0100	1	0111	Shift	Second Cycle
0101	0100	1	0111	A A + M	Third Cycle
0010	1010	0	0111	Shift	
0001	0101	0	0111	Shift	Fourth Cycle

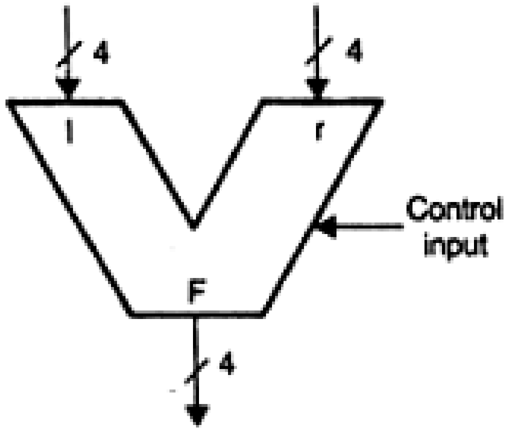
10 STEPS FOR HARDWIRED CONTROL

Step 5: Processing section includes GPRs, 4-bit adder / subtractor, Tristate buffers



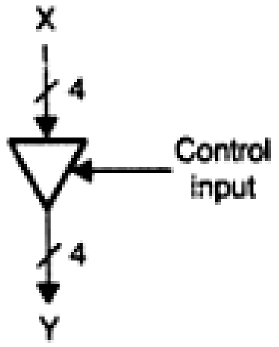
C	L	R	D	Clock	Action
1	0	0	0	↓	Clear
0	1	0	0	↓	Load external data
0	0	1	0	↓	Right shift
0	0	0	1	↓	Decrement by one
0	0	0	0	↓	No change

a. Storage Register



Control input	F
1	l + r
0	l − r

b. Adder-subtractor



Control input	Y
1	X
0	High Z

c. Tri-state Buffer

10 STEPS FOR HARDWIRED CONTROL

Step 6: The complete design of processing section establishing control points.

C_0:	$A \leftarrow 0$
C_1:	$M \leftarrow$ Inbus
C_2:	$L \leftarrow 4$
C_3:	$Q[4:1] \leftarrow$ Inbus
	$Q[0] \leftarrow 0$
C_4:	$F = l + r$
C_4:	$F = l - r$
C_5:	$A \leftarrow F$
C_6:	ASR (A \$ Q)
C_7:	$L \leftarrow L - 1$
C_8:	Outbus = A
C_9:	Outbus = Q[4:1]

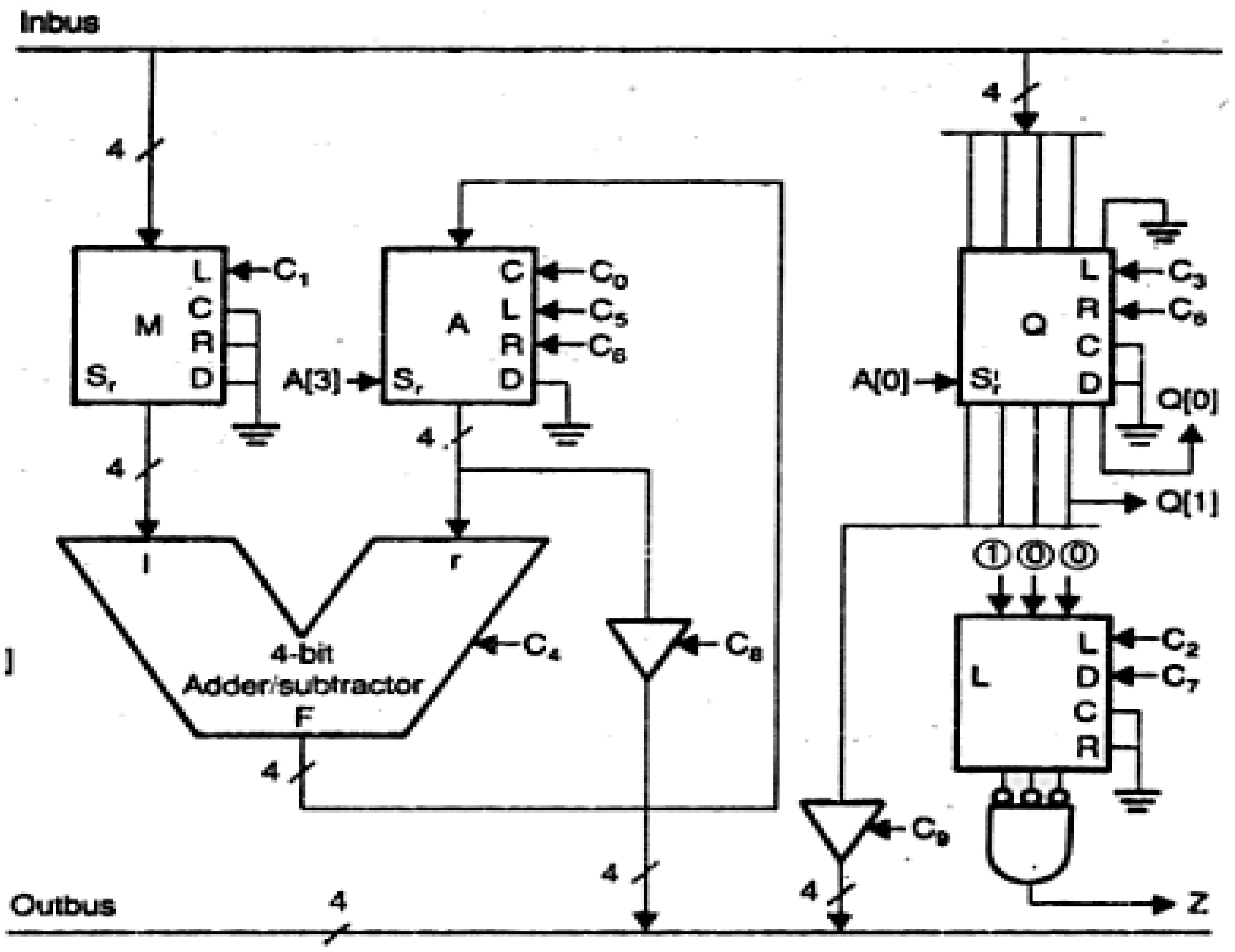


Figure 4.18 Processing Section of the Booth's Multiplier

TOPICS COVERED FROM

- Textbook 3:
  - Chapter 4: 4.2 and 4.3

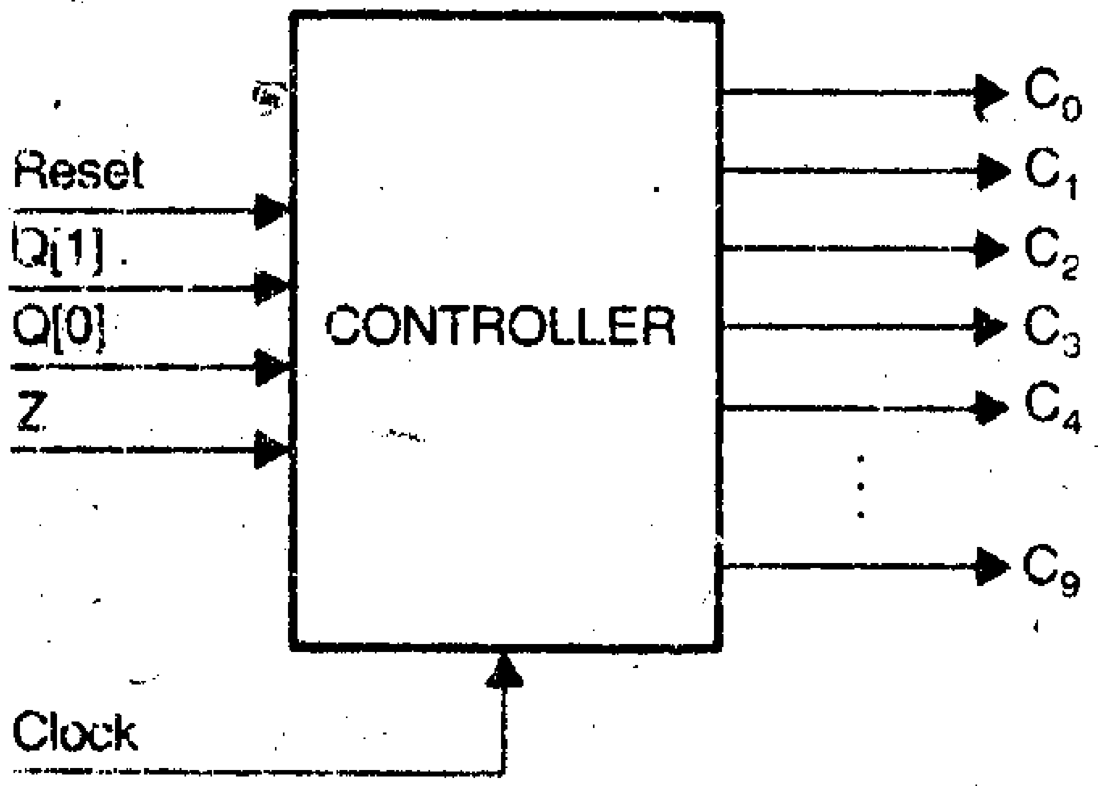
COMPUTER ORGANIZATION AND ARCHITECTURE

Course Code	: CSE 2151
Credits	: 04

10 STEPS FOR HARDWIRED CONTROL

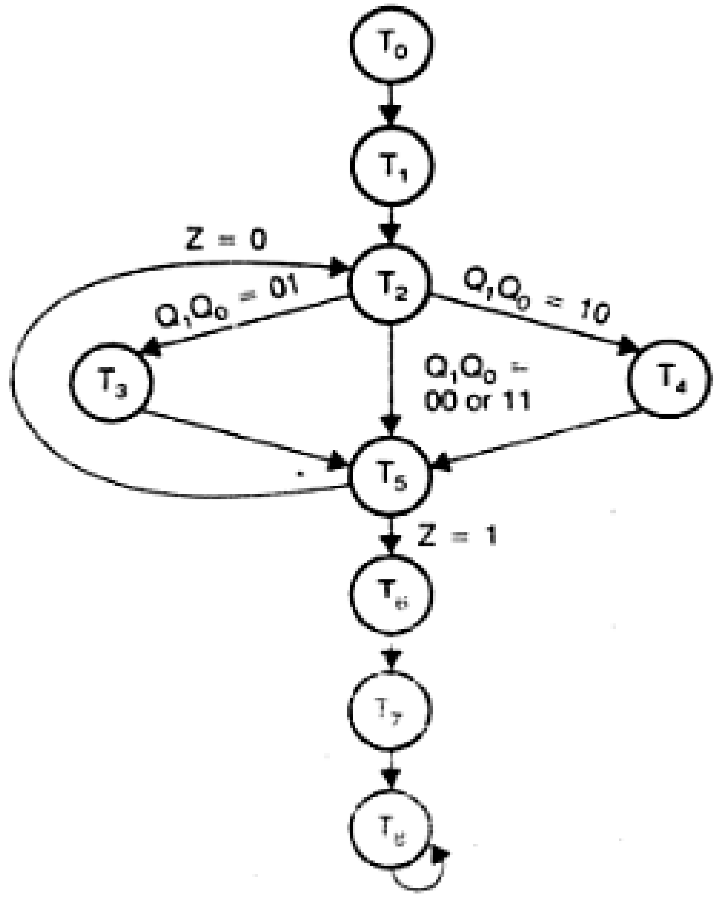
Step 7: Block diagram of controller:

- will have 5 I/Ps and 10 O/Ps.
- RESET i/p is used to reset the controller so a new computation can begin.
- CLK is used to synchronize the controller action for trailing edge of clock pulse.



10 STEPS FOR HARDWIRED CONTROL

Step 8: The state diagram of Booth's multiplier controller



a. State Diagram

CONTROL STATE	OPERATION PERFORMED	CONTROL SIGNALS TO BE ACTIVATED
T_0	$A \leftarrow 0$, $L \leftarrow 4$, $M \leftarrow$ Inbus	C_0, C_1, C_2
T_1	$Q[4:1] \leftarrow$ Inbus, $Q[0] \leftarrow 0$	C_3
T_2	None	None
T_3	$A \leftarrow A + M$	C_4, C_5
T_4	$A \leftarrow A - M$	C_5 $(C_4 = 0)$
T_5	ASR (ASQ), $L \leftarrow L - 1$	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

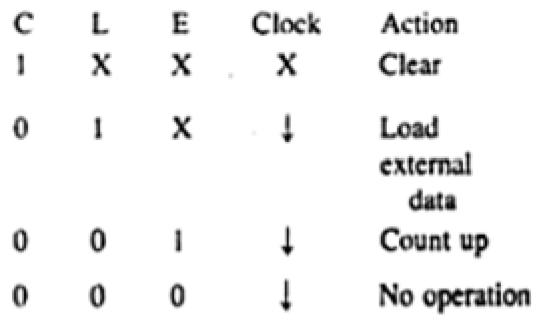
Figure 4.20 Controller Description

C_0:	$A \leftarrow 0$
C_1:	$M \leftarrow$ Inbus
C_2:	$L \leftarrow 4$
C_3:	$Q[4:1] \leftarrow$ Inbus $Q[0] \leftarrow 0$
C_4:	$F = l + r$
C_4':	$F = l - r$
C_5:	$A \leftarrow F$
C_6:	ASR (A \$ Q)
C_7:	$L \leftarrow L - 1$
C_8:	Outbus = A
C_9:	Outbus = $Q[4:1]$

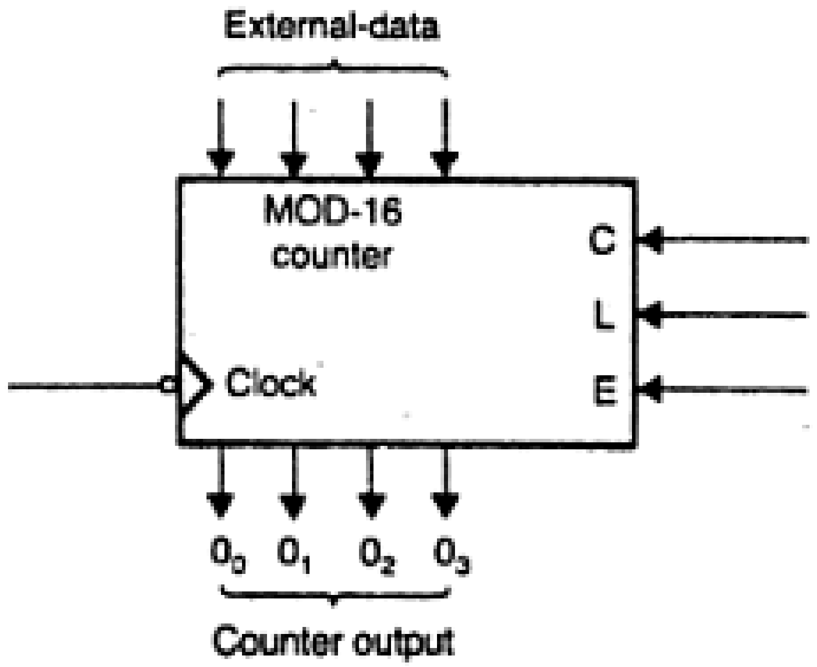
10 STEPS FOR HARDWIRED CONTROL

Step 9: The controller includes a mod -16 counter, a 4: 16 decoder, a sequence controller (SC).

- SC HW, which sequences the controller according to state diagram.
- Hence Truth Table for SC must be derived from the controller's state



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



a. Block Diagram

10 STEPS FOR HARDWIRED CONTROL

Step 10: Logic Diagram of the Booth's multiplier controller

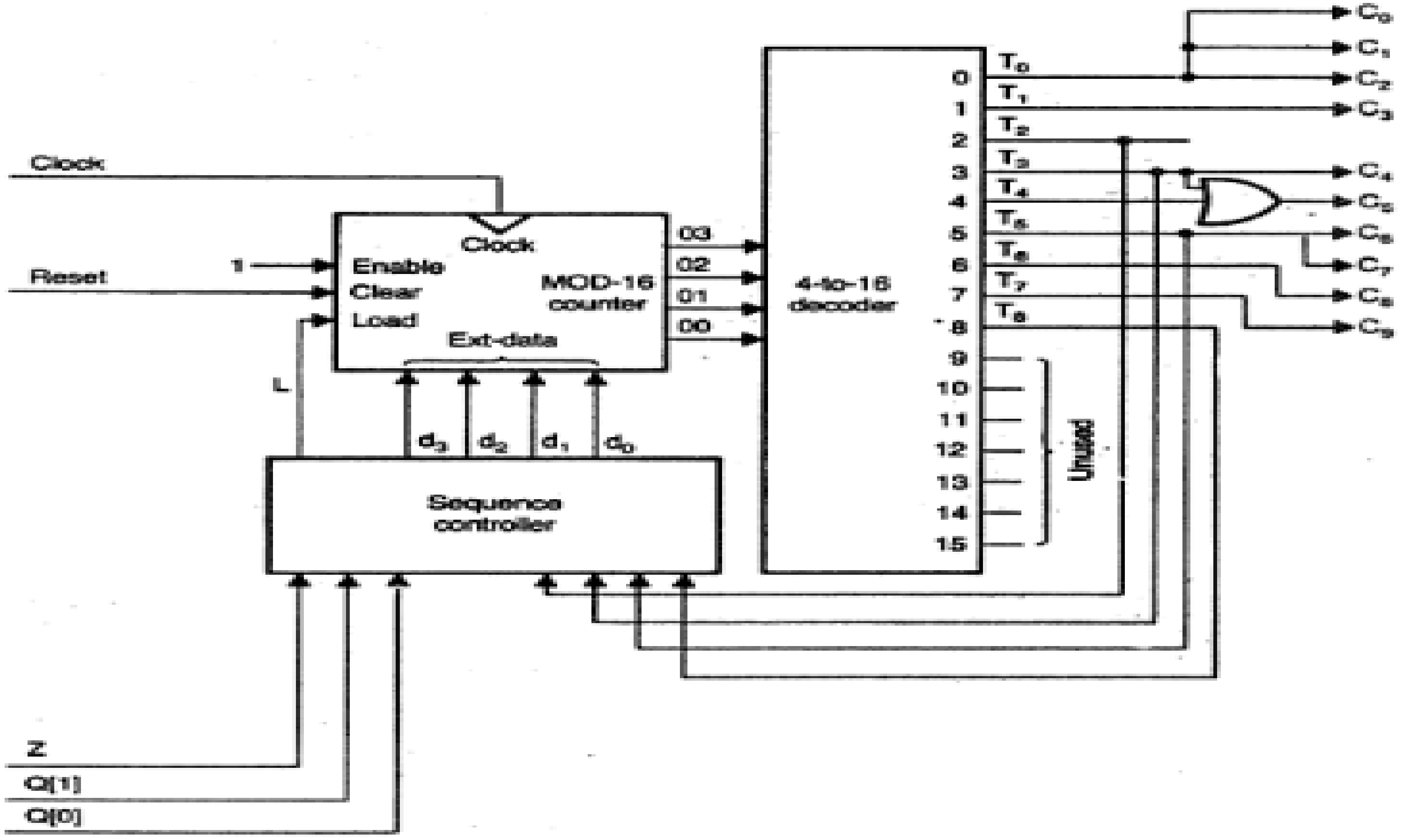
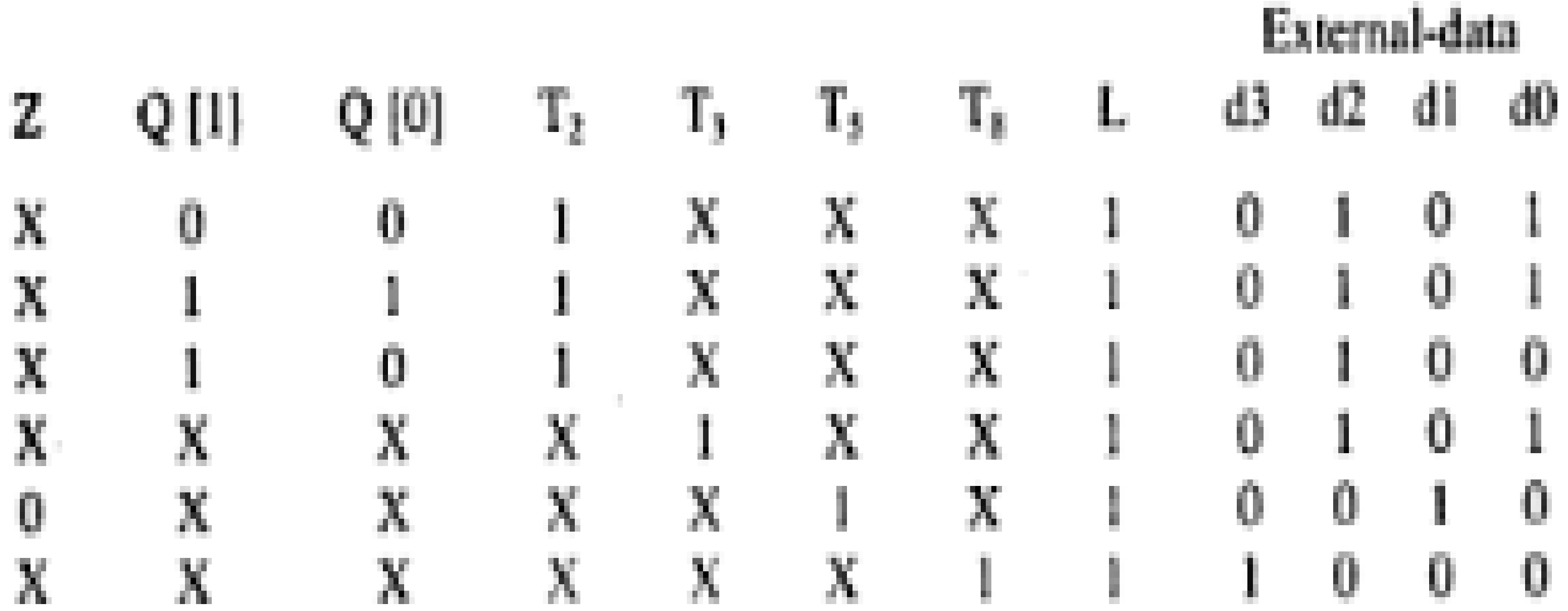


Figure 4.23 Logic Diagram of the Booth's Multiplier Controller

10 STEPS FOR HARDWIRED CONTROL

Step 10: Truth Table for SC Design

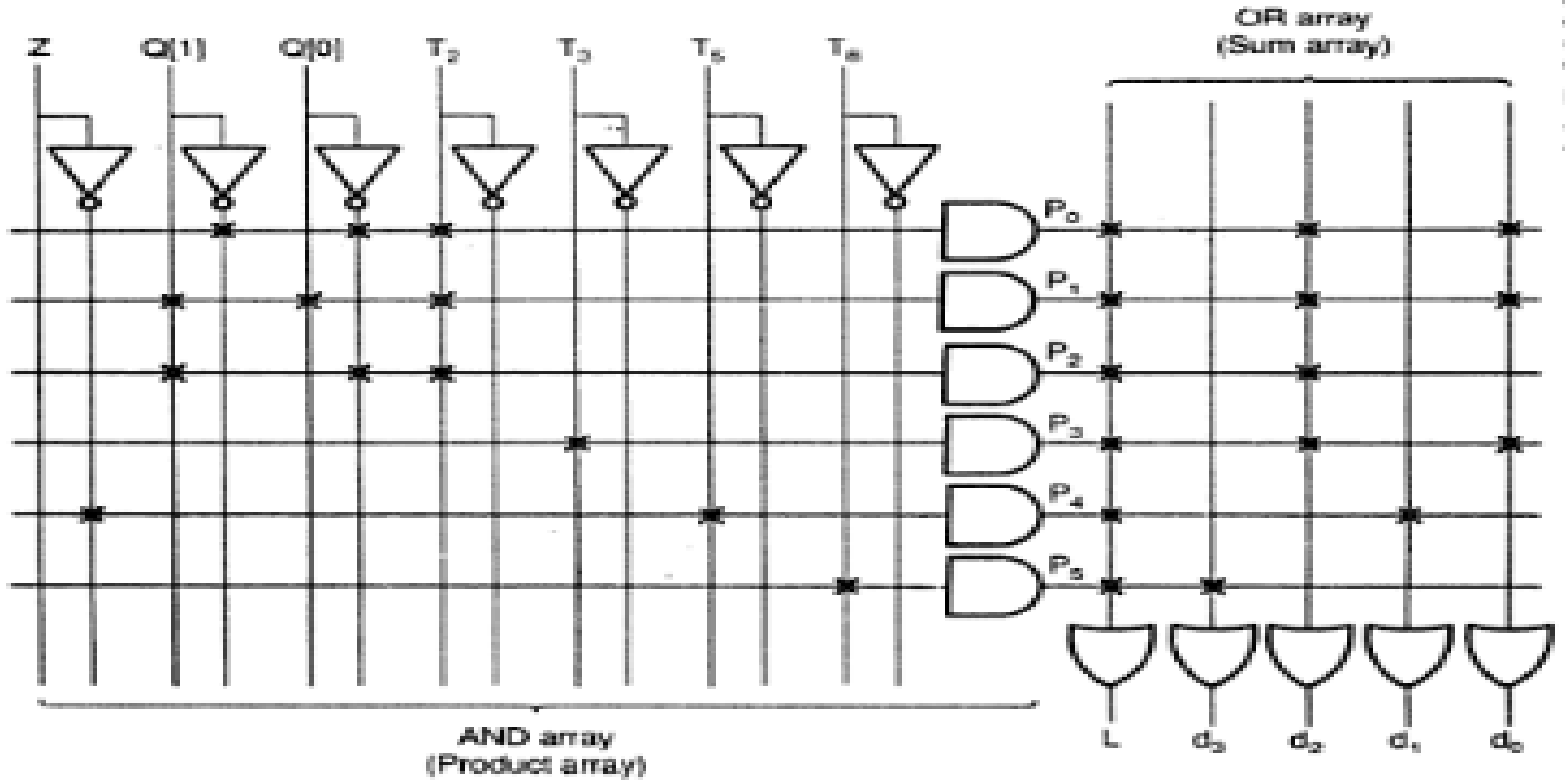


								External-data			
Z	Q [1]	Q [0]	T_2	T_1	T_3	T_0	L	d3	d2	d1	d0
X	0	0	1	X	X	X	1	0	1	0	1
X	1	1	1	X	X	X	1	0	1	0	1
X	1	0	1	X	X	X	1	0	1	0	0
X	X	X	X	1	X	X	1	0	1	0	1
0	X	X	X	X	1	X	1	0	0	1	0
X	X	X	X	X	X	1	1	1	0	0	0

10 STEPS FOR HARDWIRED CONTROL

Step 10: PLA Design

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



b. PLA Implementation

Figure 4.24 Sequence Controller Design

10 STEPS FOR HARDWIRED CONTROL

Step 10: PLA Design

Implementing SC using PLA:

$P_0 = Q[1]' Q[0]' T_2$

$P_1 = Q[1] Q[0] T_2$

$P_2 = Q[1] Q[0]' T_2$

$P_3 = T_3$

$P_4 = Z' T_5$

$P_5 = T_8$

- The PLA o/ps are summarized as
- L = P0 + P1 + P2 + P3 + P4 + P5
- d3 = P5
- d2 = P0 + P1 + P2 + P3
- d1 = P4
- d0 = P0 + P1 + P3

- The controller design is completed by relating the control (T0-T8) with control i/ps C0-C9 as below:
- C0 = C1 = C2 = T0
- C3 = T1
- C4 = T3
- C5 = T3 + T4
- C6 = C7 = T5
- C8 =T6
- C9 = T7

TOPICS COVERED FROM

- Textbook 3:
  - Chapter 4: 4.3

COMPUTER ORGANIZATION AND ARCHITECTURE

Course Code : CSE 2151

Credits : 04

MICROPROGRAMMED CONTROL UNIT

- Control word: all control signals that can be simultaneously activated are grouped to form the CW.
- Micro operation: A←0, outbus=A, etc..
- Each CW contains signals to activate one or more microoperations.
- Control memory:
  - Control words are held in separate memory called control memory (CM).
  - Control words are fetched from CM and individual control fields are routed to various functional units to achieve desired task.
- All microinstructions have 2 important fields:
  - Control field
  - Next address field
- Purpose of control field is to indicate which control lines are to be activated.
- Purpose of Next address field is to specify the address of the next microinstruction to be executed.

WILKE'S DESIGN OF MICROPROGRAMMED CONTROL UNIT

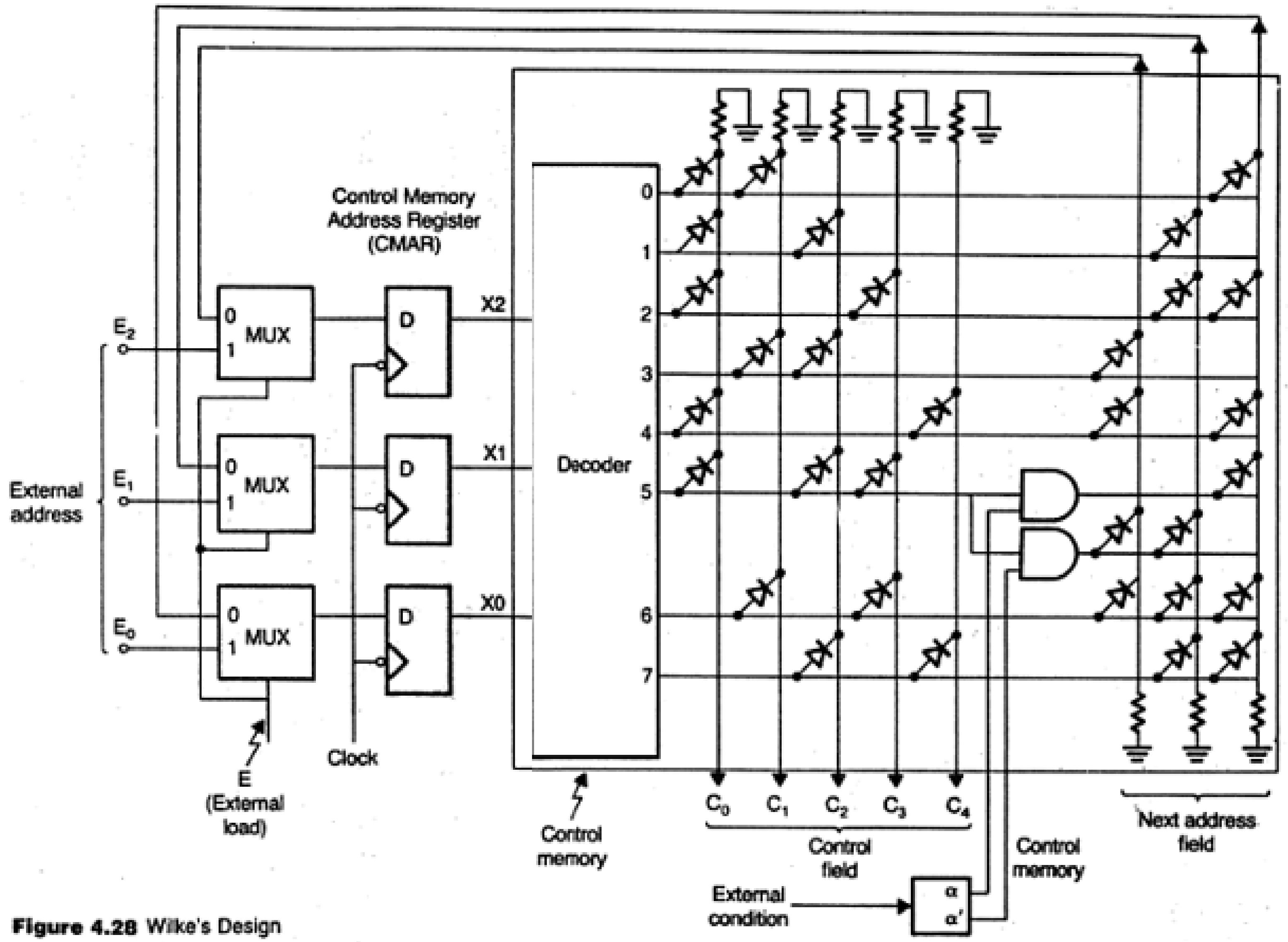


Figure 4.28 Wilke's Design

ENCODING AND DECODING

- The length of the μinstruction is dependent on factors:
  - The degree of parallelism
  - The control field organization
  - The method by which the address of next μinstruction is specified.
- All μoperations executed in parallel can be specified in a single μinstruction.
- This allows short μprograms to be written.
- If the degree of parallelism increases, then the length of μinstruction increases.
- Similarly short μinstructions have limited capability in expressing parallelism. The overall length of μprogram will increase.
- Various ways of organizing the control information:
- Consider A, B, C, D each communicates with the outbus
  - C0: outbus=A;
  - C1: outbus=B;
  - C2: outbus=C;
  - C3: outbus=D;

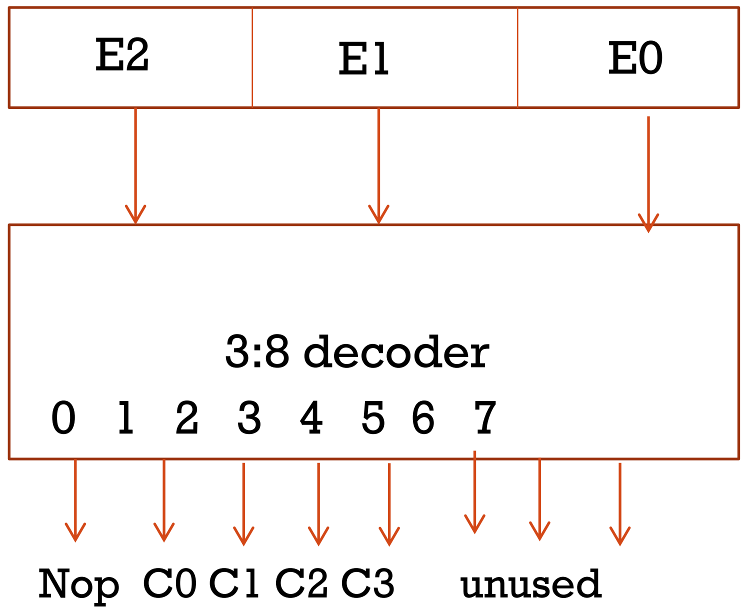
ENCODING AND DECODING

- Here there is no need of decoding the control field.
- This method is known as unencoded format.

C0	C1	C2	C3	
1	0	0	0	Outbus=A
0	1	0	0	Outbus=B
0	0	1	0	Outbus=C
0	0	0	1	Outbus=D
0	0	0	0	NO operation

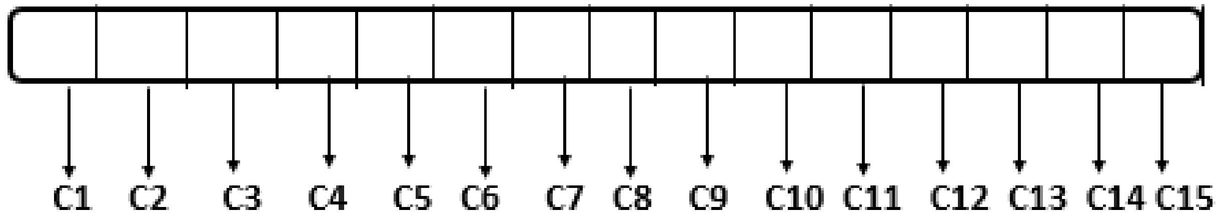
ENCODING AND DECODING

- The above valid 5 binary patterns can be represented as

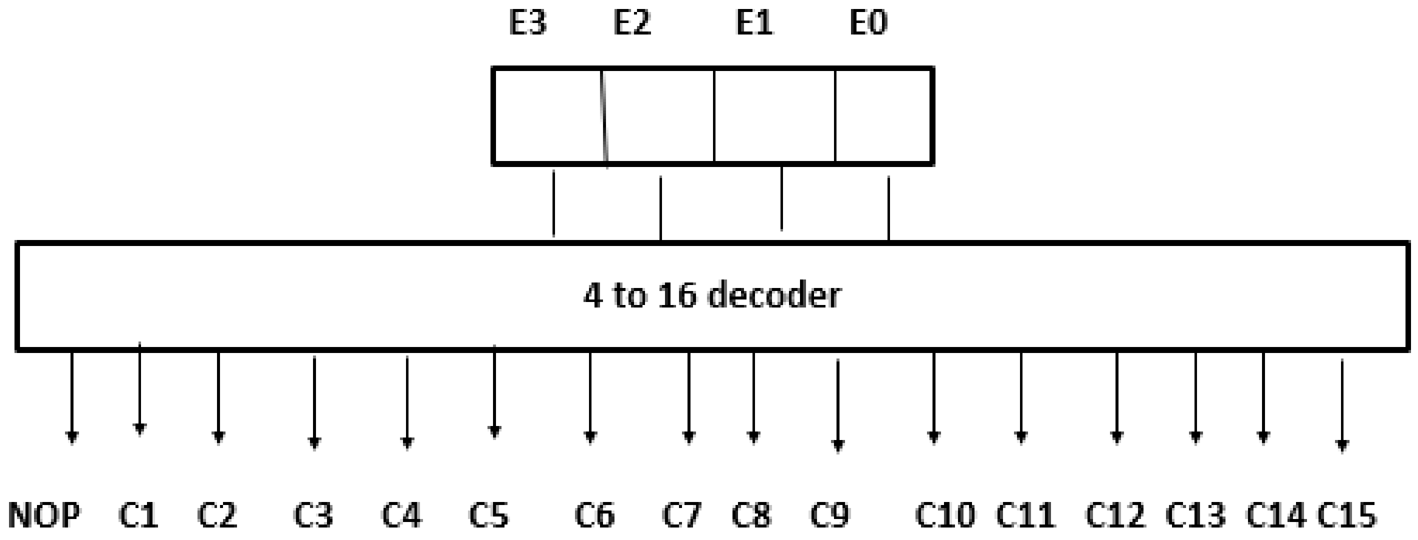


E2	E1	E0	
0	0	0	No operation
0	0	1	Outbus=A
0	1	0	Outbus=B
0	1	1	Outbus=C
1	0	0	Outbus=D

FULLY UNENCODED FORM

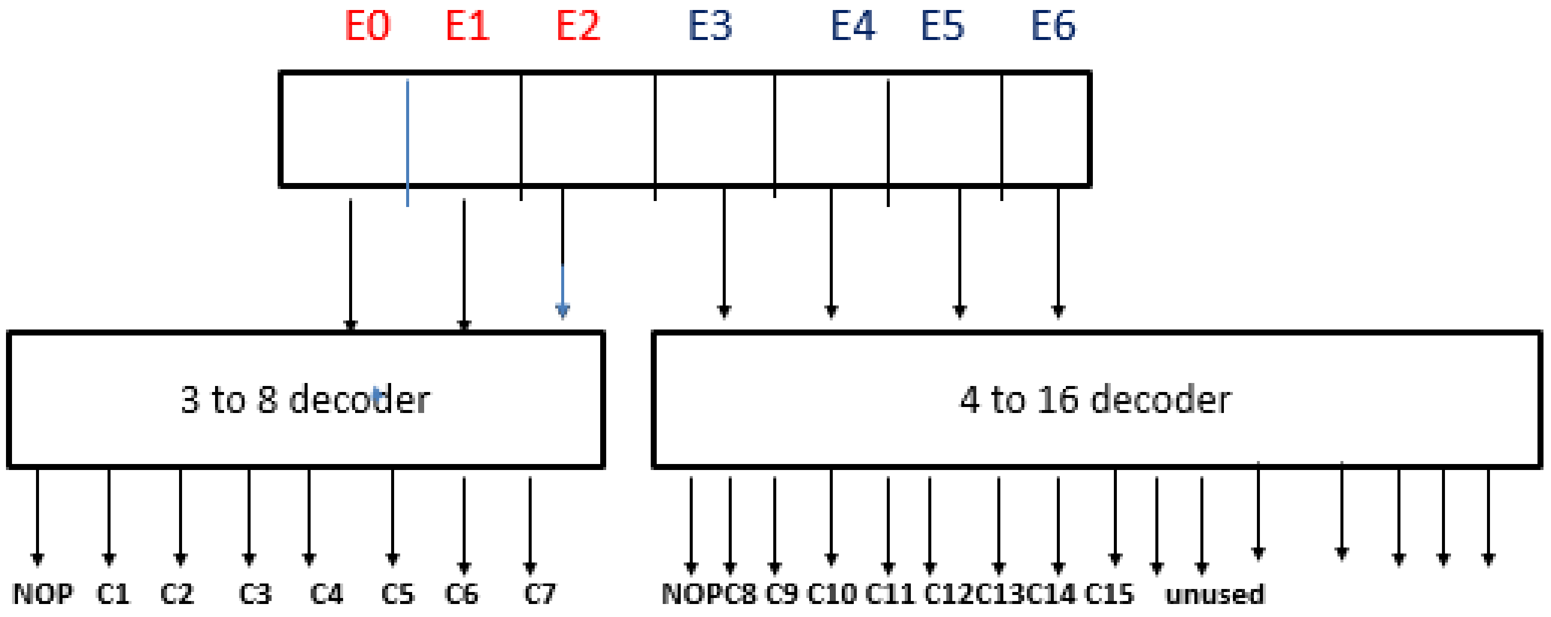


- Encoded form--control field size :4 bits



PARTIALLY ENCODED FORM

- E0 E1 E2 Group1→ C1, C2,C3,C4,C5.C6,C7
  E3 E4 E5 E6 Group2→ C8, C9, C10, C11, C12, C13, C14, C15



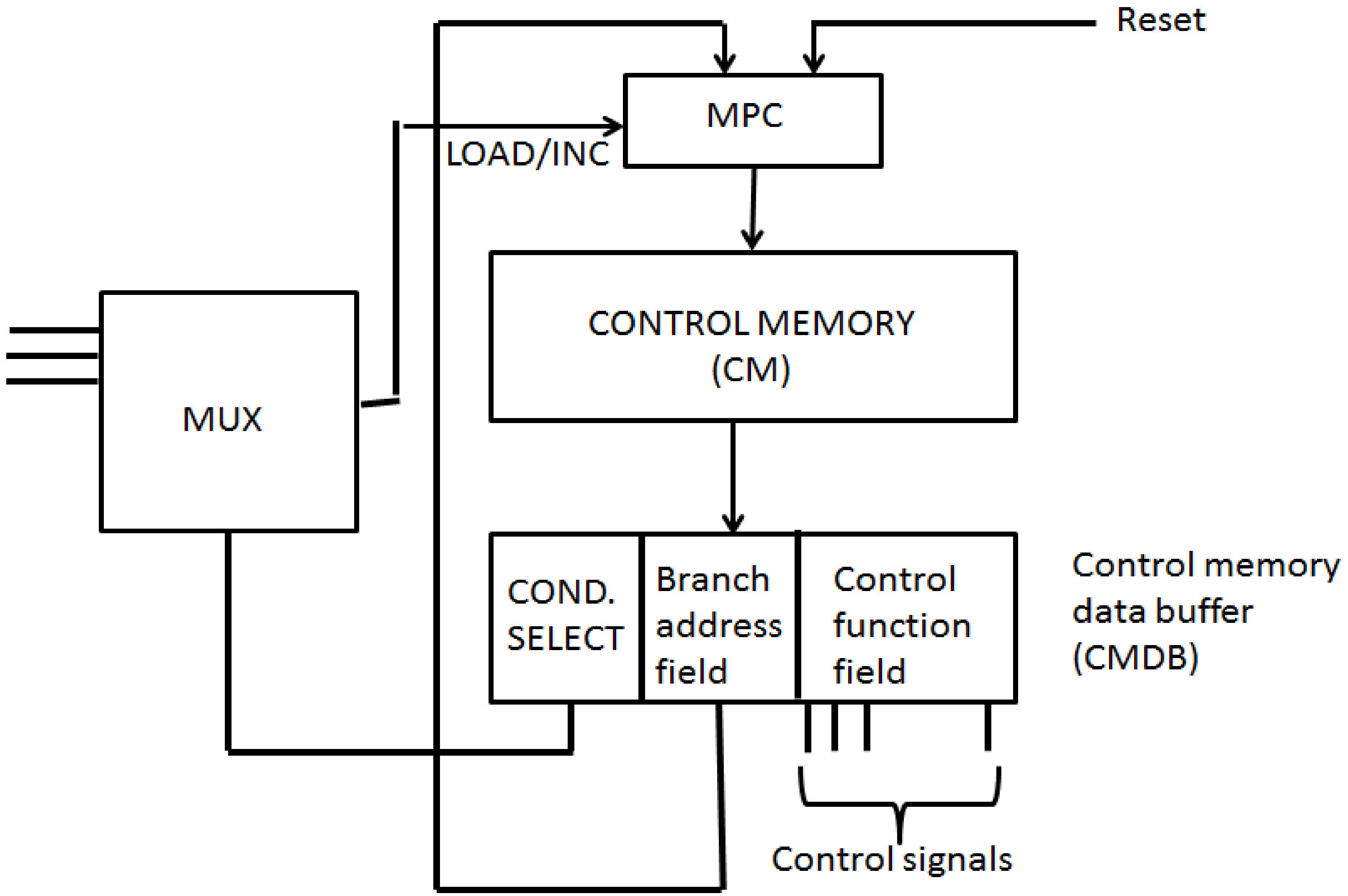
MICROINSTRUCTIONS CLASSIFICATION

- Two groups based which express parallelism and the amount of encoding called
  - Vertical microinstruction
    - Need considerable amount of decoding
    - Short microinstruction
    - Limited scope for expressing parallelism
  - Horizontal microinstruction
    - No decoding required
    - Long microinstruction
    - Capability of expressing a high degree of parallelism

ARCHITECTURE OF MODERN MICROPROGRAMMED CONTROL UNIT

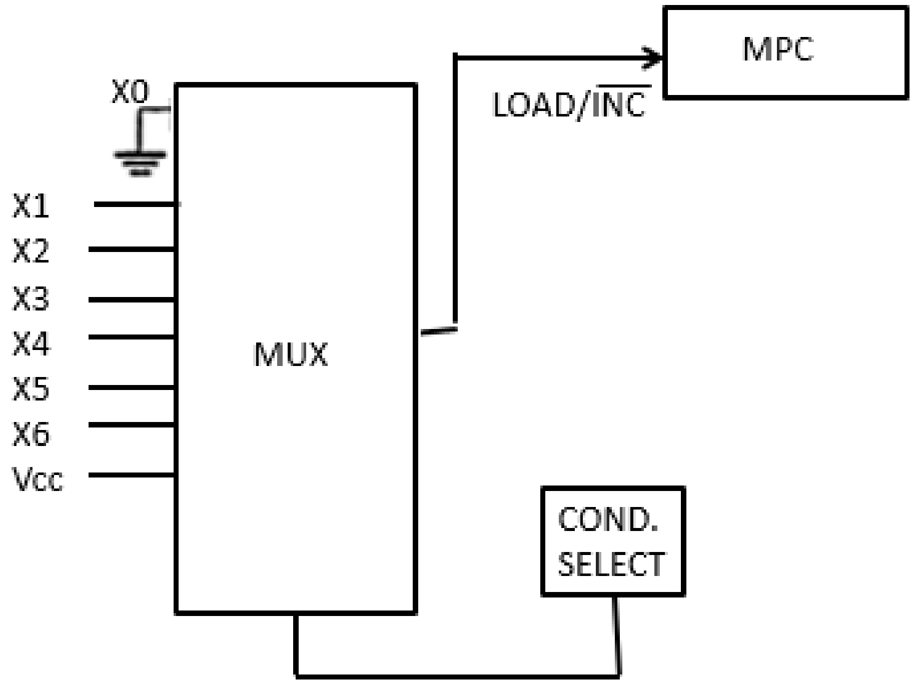
- Next address field from the μinstruction can be eliminated. This pointer is referred as microprogram counter (MPC).
- MPC is functionally identical to PC.
- It points to the μinstruction to be executed next and incremented after each μinstruction fetch.

GENERAL PURPOSE MICROPROGRAMMED CONTROL ORGANIZATION



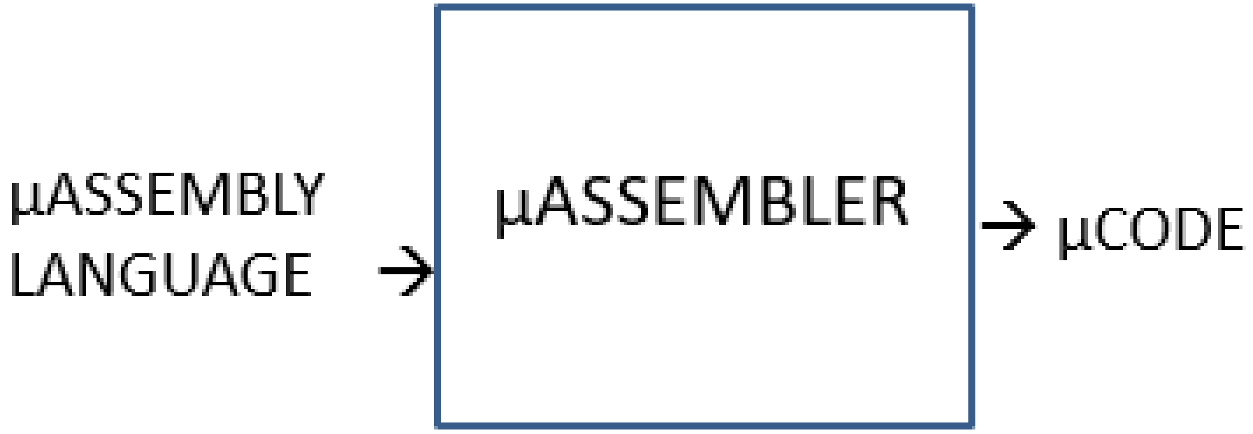
GP MICROPROGRAMMED CONTROL ORGANIZATION

- Suppose 6 external conditions X1, X2, X3, ..X6 are to be tested.
- Cond select field and MUX can be organized as:
  - If cond sel 000 then MUX o/p 0. MPC incremented. No branch.
  - If cond sel 111 MUX o/p 1. Unconditional branching.
  - If cond sel 001 MUX o/p is same as value of X1. Now MPC will be loaded with branch addr when X1=1 else it is incremented.



GP MICROPROGRAMMED CONTROL ORGANIZATION

- Writing μprogram is like writing ALP.
- μprogrammer must have more thorough knowledge about system architecture
- To speedup the development of μcode, →μassembly language.
- These μcodes are held in CM.



DESIGN OF MICROPROGRAMMED CU FOR 4X4 BOOTH'S MULTIPLIER

- STEP1: Write μprogram in a symbolic form.

Control Mem Addr	**Control Word**	
0	START:	A←0, M ← Inbus, L ← 4;
1		Q[4:1] ← Inbus, Q[0] ← 0;
2	LOOP:	If Q[1:0]=01 then goto ADD;
3		If Q[1:0]=10 then goto SUB;
4		goto RSHIFT;
5	ADD:	A ← A+M;
6		Goto RSHIFT;
7	SUB:	A ← A-M;
8	RSHIFT:	ASR(A$Q), L ← L-1;
9		If Z=0 then goto LOOP
10		outbus=A;
11		outbus=Q[4:1];
12	HALT:	goto HALT

- CM holds 13 words, requiring a 4-bit branch address field.

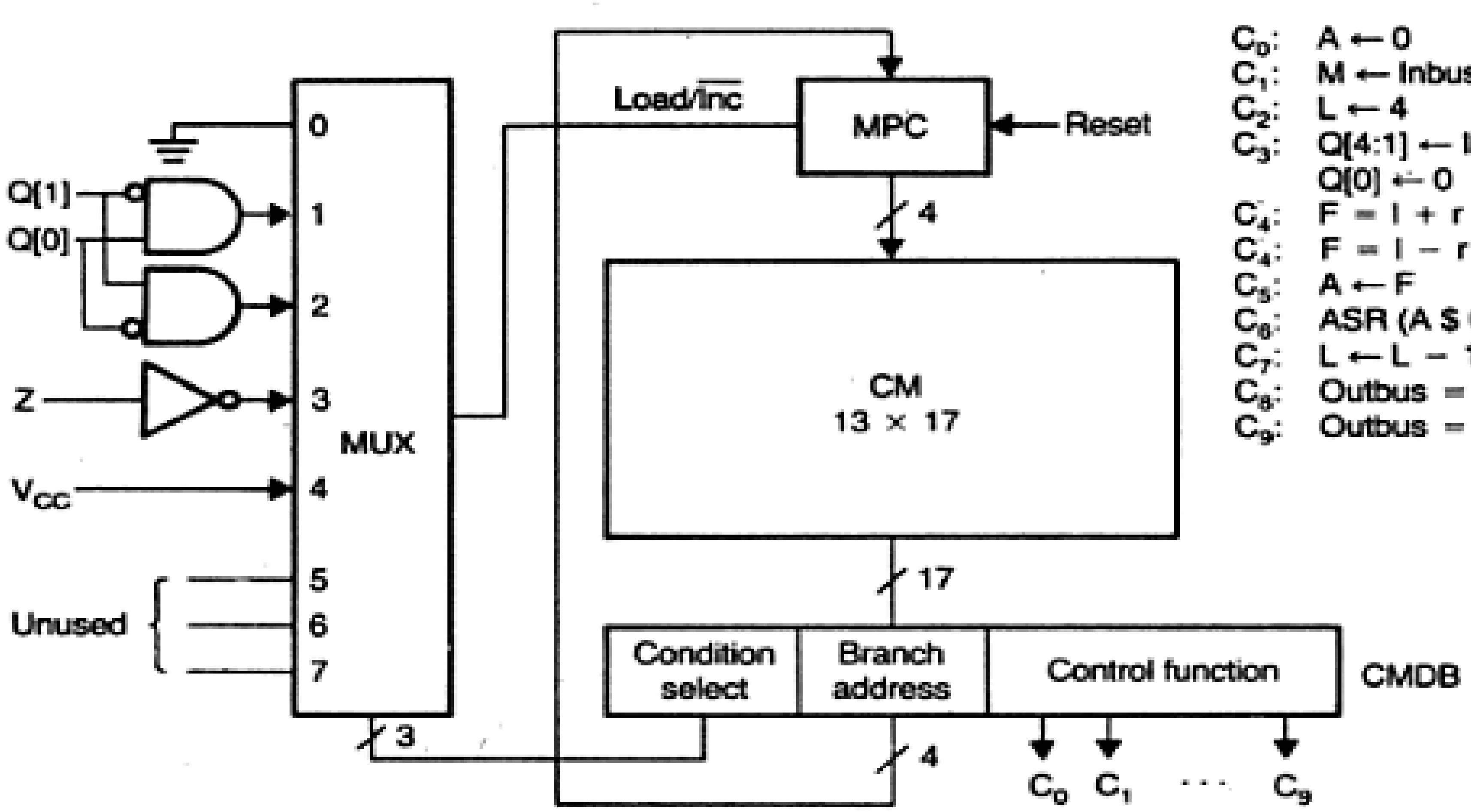
DESIGN OF MICROPROGRAMMED CU FOR 4X4 BOOTH'S MULTIPLIER

- STEP2: Q[1]Q[0]=01, Q[1]Q[0]=10 and Z=0 are checked.
- These conditions are applied as i/ps to condition select MUX.
- MUX must have atleast 5 data i/ps and 8:1 data selector.
- 3-bit cond sel field is used to encode 5 diff cond:

Cond Select Field	**Action Taken**
0 0 0	No branching
0 0 1	Branch if Q[1]Q[0]=01
0 1 0	Branch if Q[1]Q[0]=10
0 1 1	Branch if Z=0
1 0 0	Unconditional branch

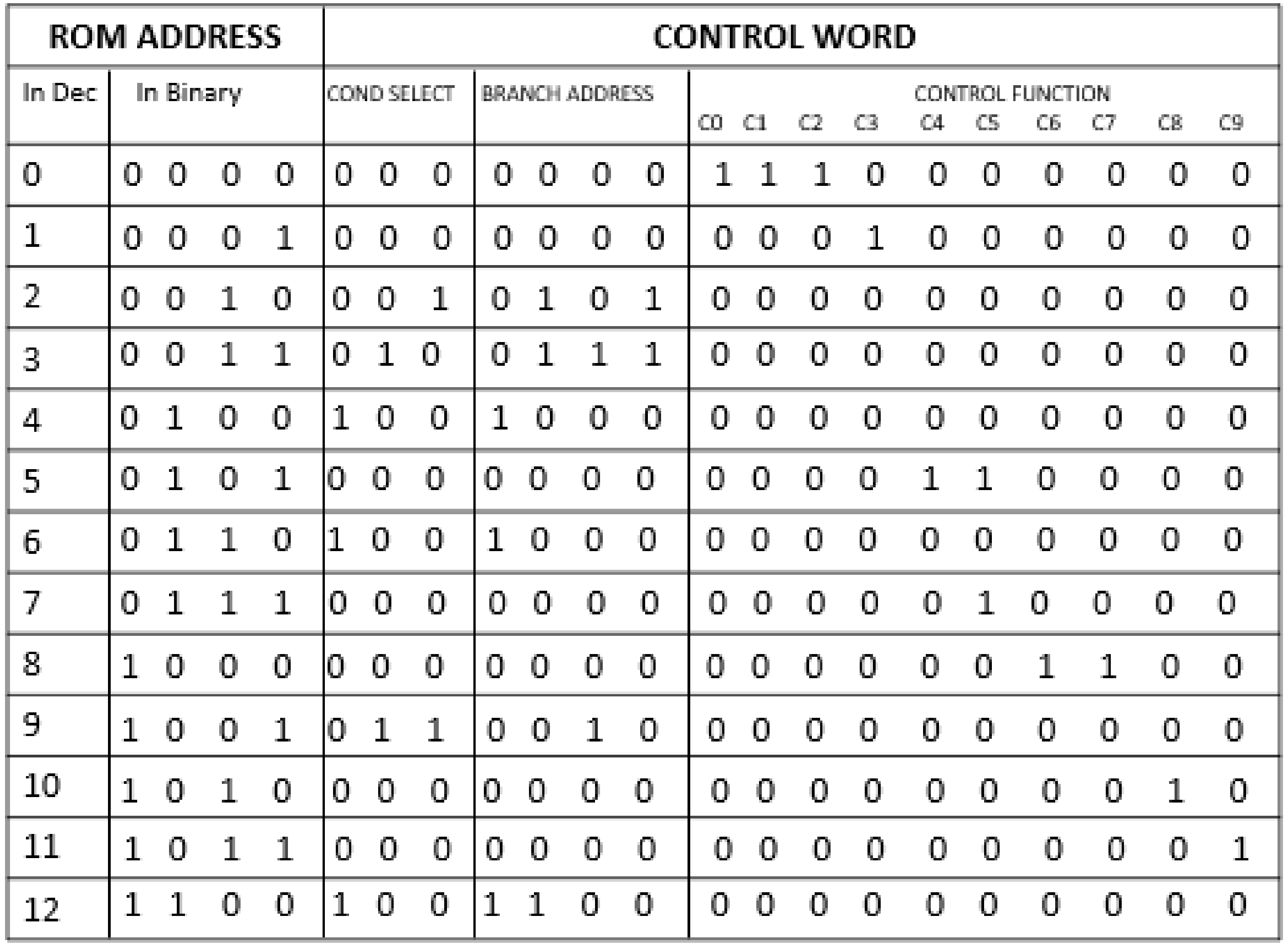
- Size of CW = Size of Cond sel field + Size of branch(size of address bits) + No of functions field
  = 3 + 4 + 10
  = 17 bits
- Size of CMDB is 17 bits and CM is 13 x 17 =221bits

DESIGN OF MICROPROGRAMMED CU FOR 4X4 BOOTH'S MULTIPLIER



C_0:	$A \leftarrow 0$
C_1:	$M \leftarrow$ Inbus
C_2:	$L \leftarrow 4$
C_3:	$Q[4:1] \leftarrow$ Inbus
	$Q[0] \leftarrow 0$
C_4':	$F = l + r$
C_4'':	$F = l - r$
C_5:	$A \leftarrow F$
C_6:	ASR (A \$ Q)
C_7:	$L \leftarrow L - 1$
C_8:	Outbus = A
C_9:	Outbus = Q[4:1]

DESIGN OF MICROPROGRAMMED CU FOR 4X4 BOOTH'S MULTIPLIER



ROM ADDRESS		CONTROL WORD											
In Dec	In Binary	COND SELECT	BRANCH ADDRESS	CONTROL FUNCTION									
				C0	C1	C2	C3	C4	C5	C6	C7	C8	C9
0	0 0 0 0	0 0 0	0 0 0 0	1	1	1	0	0	0	0	0	0	0
1	0 0 0 1	0 0 0	0 0 0 0	0	0	0	1	0	0	0	0	0	0
2	0 0 1 0	0 0 1	0 1 0 1	0	0	0	0	0	0	0	0	0	0
3	0 0 1 1	0 1 0	0 1 1 1	0	0	0	0	0	0	0	0	0	0
4	0 1 0 0	1 0 0	1 0 0 0	0	0	0	0	0	0	0	0	0	0
5	0 1 0 1	0 0 0	0 0 0 0	0	0	0	0	1	1	0	0	0	0
6	0 1 1 0	1 0 0	1 0 0 0	0	0	0	0	0	0	0	0	0	0
7	0 1 1 1	0 0 0	0 0 0 0	0	0	0	0	0	1	0	0	0	0
8	1 0 0 0	0 0 0	0 0 0 0	0	0	0	0	0	0	1	1	0	0
9	1 0 0 1	0 1 1	0 0 1 0	0	0	0	0	0	0	0	0	0	0
10	1 0 1 0	0 0 0	0 0 0 0	0	0	0	0	0	0	0	0	1	0
11	1 0 1 1	0 0 0	0 0 0 0	0	0	0	0	0	0	0	0	0	1
12	1 1 0 0	1 0 0	1 1 0 0	0	0	0	0	0	0	0	0	0	0

Binary listing of μprogram for 4 X 4 Booth's Multiplier

HARDWIRED APPROACH V/S MICROPROGRAMMED CU:

- Microprogrammed approach is more expensive.
- Control memory may reduce the overall speed of the machine, since microinstructions retrieval process takes significant amount of time.
- Microprogramming provides a well-structured control organization.
- With Microprogramming, many additions and changes are made by simply changing the microprogram in Control memory, whereas a small change in hardwired approach may lead to redesign the entire system.
- Cost of the control logic increases with system complexity though hardwired logic is economical for simple control algorithm. In microprogrammed implementation cost of the simplest system is higher though adding new features requires additional control memory.

EXERCISE

Consider the following register transfer description algorithm

Declare Registers: A[8], B[8], C[8];

```
START:  A ← 0;
        B ← 00001010;
LOOP:   A ← A + B;
        B ← B - 1;
        If B< > 0 then go to LOOP;
        C ← A;
HALT:   Go to HALT
```

A. Design the processing section for implementing the above algorithm identifying all the control points.

B. Draw a neat state diagram. Write the operations performed and the control signals to be activated in each state

C. Design controller using decoder, counter and sequence controller and draw the diagram

D. Draw the PLA diagram for implementing the sequence controller

D. Design a Modern Microprogrammed control unit and give the binary listing of the microprogram

TOPICS COVERED FROM

- Textbook 3:
  - Chapter 4: 4.3.2