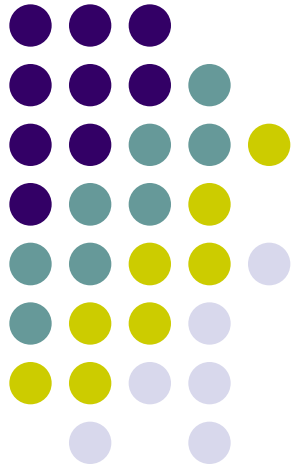
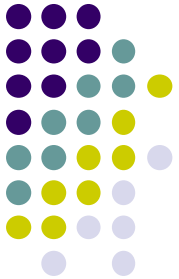
Chapter 4. control unit

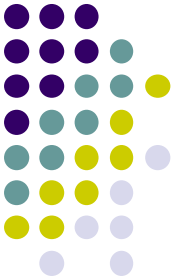


Agenda



- Basic concepts
  - Fundamentals of CU
  - Register transfer notations and descriptions
  - Buses
- Design methods
  - Hardwired approach
  - Microprogramming

Major Components of CPU

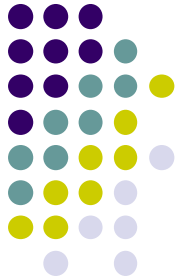


- Processing Section
  - Includes hardware elements
  - ALU, Shift registers, Comparators, Multipliers
  - To Operate on data(character codes, integers , real numbers)
- Control Unit
  - To control system operations by routing the selected data items to the selected processing hardware at right time

Cont...

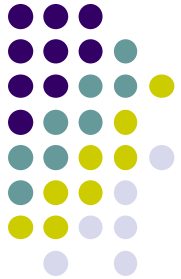
- Responsibility of control unit: ‘
  - To drive associated processing hardware by generating a set of signals that are synchronized with a Master clock
- Input to control unit:
  - The Master Clock (a clock that provides the primary source of internal timings for a processor or stand alone control unit)
  - Status information from the processing section
  - Command signals from the external agent

Cont..



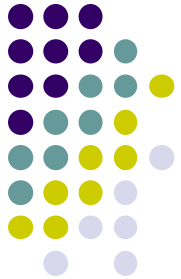
- Output of control unit:
  - The signals that drives the processing section
  - Response to an external environment
- Responsibilities undertaken
  - Instruction Interpretation
  - Instruction Sequencing

Cont..



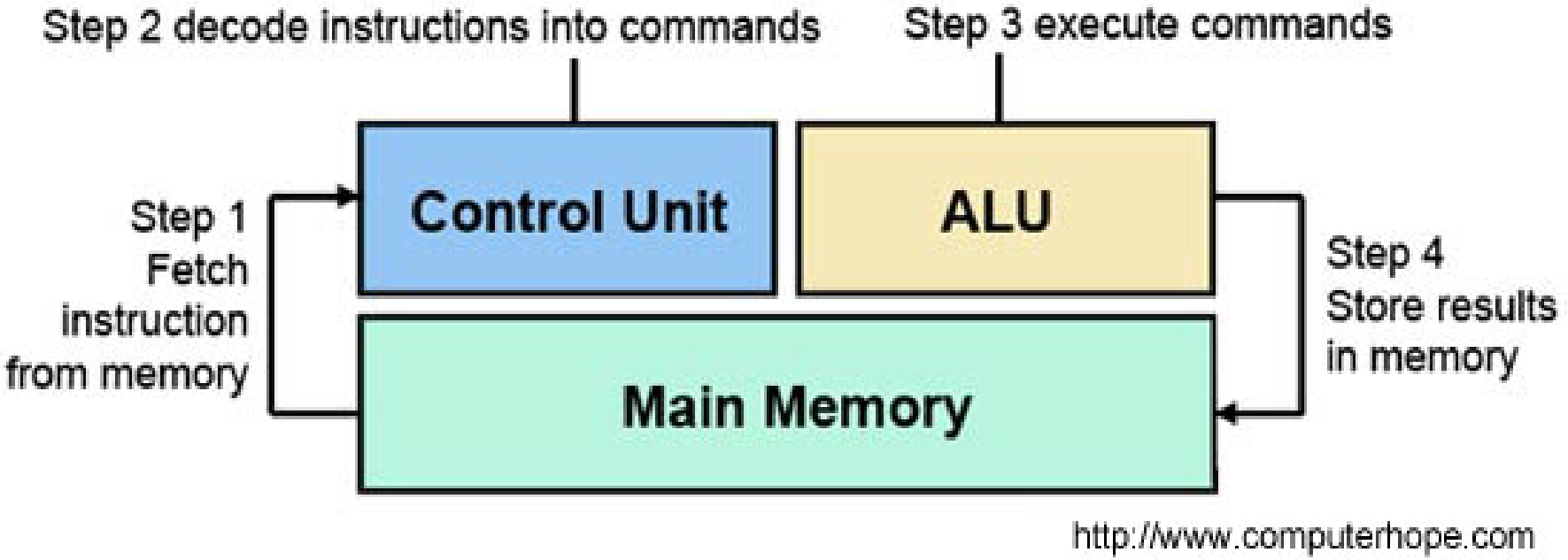
- Instruction Interpretation:
  - Control unit reads instruction from the memory unit using PC as pointer
  - Recognizes the instruction type
  - Gets necessary operands and route them to functional unit of execution unit for desired operation
  - Results routed to the specific destination

Cont..



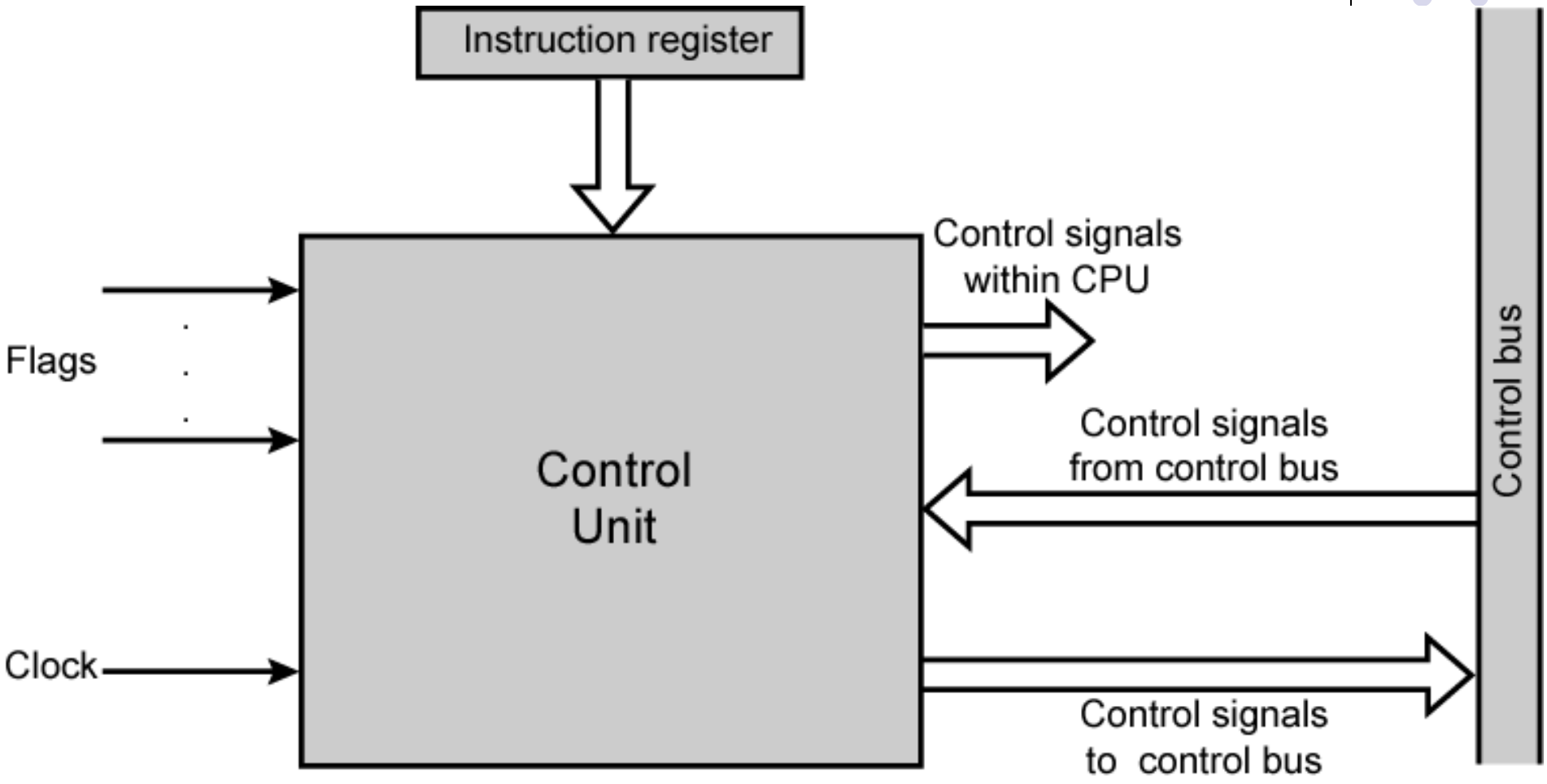
- Instruction Sequencing:
  - Control unit determines the address of the next instruction to be executed and loads it into the PC

Machine Cycle

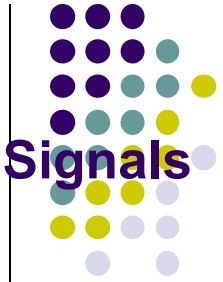


http://www.computerhope.com

Model of Control Unit

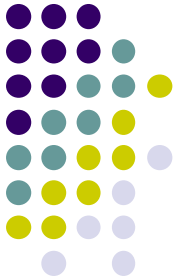


Functions of Control Unit using Control Signals



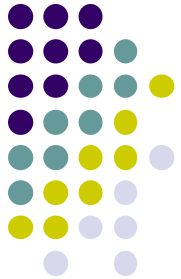
- Sequencing
  - **CU causes the CPU to step through a series of micro-operations in proper sequence based on the program being executed**
- Execution
  - **CU causes each micro-operation to be performed**
- Control Signals
  - **External: inputs indicating the state of the system**
  - **Internal: logic required to perform the sequencing and execution functions**

Fundamental Concepts



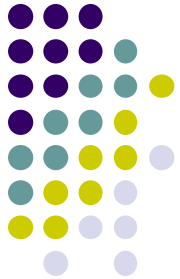
- Processor fetches one instruction at a time and perform the operation specified.
- Instructions are fetched from successive memory locations until a branch or a jump instruction is encountered.
- Processor keeps track of the address of the memory location containing the next instruction to be fetched using Program Counter (PC).
- Instruction Register (IR)

Fundamental Concepts contd..



- Purpose of control unit is to control the system operations by routing the selected data items to the selected processing HW at right time
- Control unit's responsibility is to drive the associated processing HW by generating a set of signals that are synchronized with the master clock
- In order to carry out a task such as ADD, the control unit must generate a set of control signals in a predefined sequence governed by the HW structure of the processing section.

Fundamental Concepts contd..



- Inputs to control unit are:
  - Master clock
  - Status info from processing section
  - Command signals from external agent
- Outputs produced by control unit
  - Signals that drive the processing section and responses to an external envt (operation complete or abort) due to exceptions (overflow and underflow)
- Control unit undertakes the following responsibilities
  - **Instruction interpretation:** (read instr. , recognize, get operands and route to appropriate functional units, necessary control signals issued)
  - **Instruction sequencing:** control unit determines the address of next instruction to be executed and loads to PC

Register transfer notations

- Basis for CU design are register transfer operations

Declaring registers:

A[8], B[8], PC[16]

Assigning registers

B←A

Assigning higher order byte of 16 bit PC

PCHI[8]= PC[15-8]

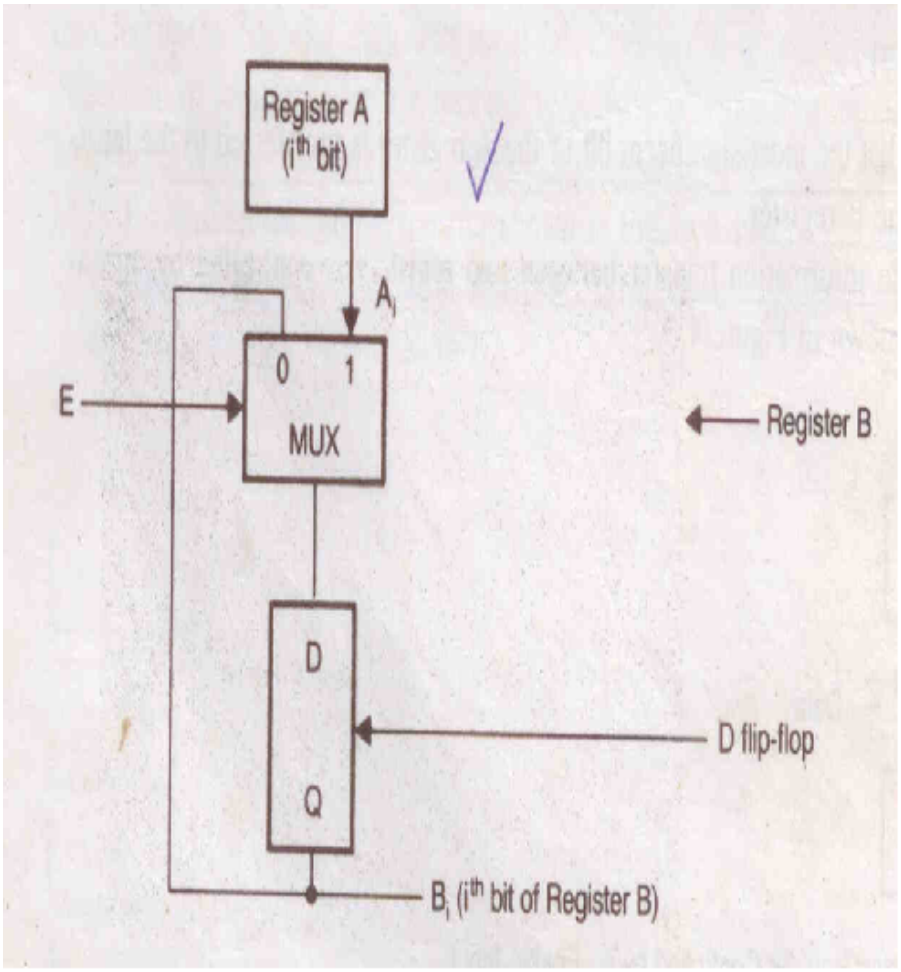
Assigning individual bits

A[4]=B[5]

Info transfer b/w 2 registers is controlled by enable signal E(control input), driven by control unit

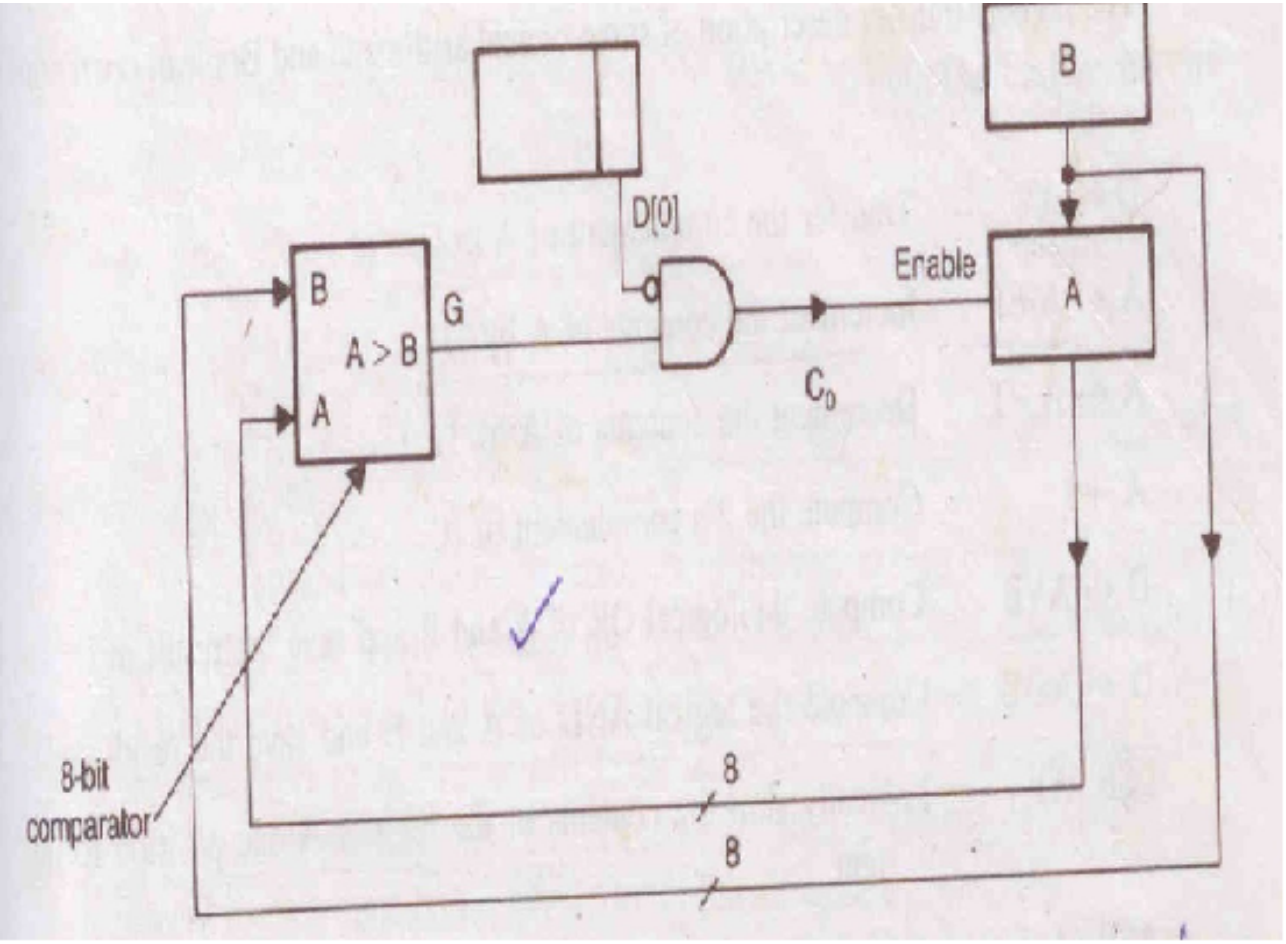
HW implementation of a register with enable I/P

E: B←A



HW implementation of C0:A←B

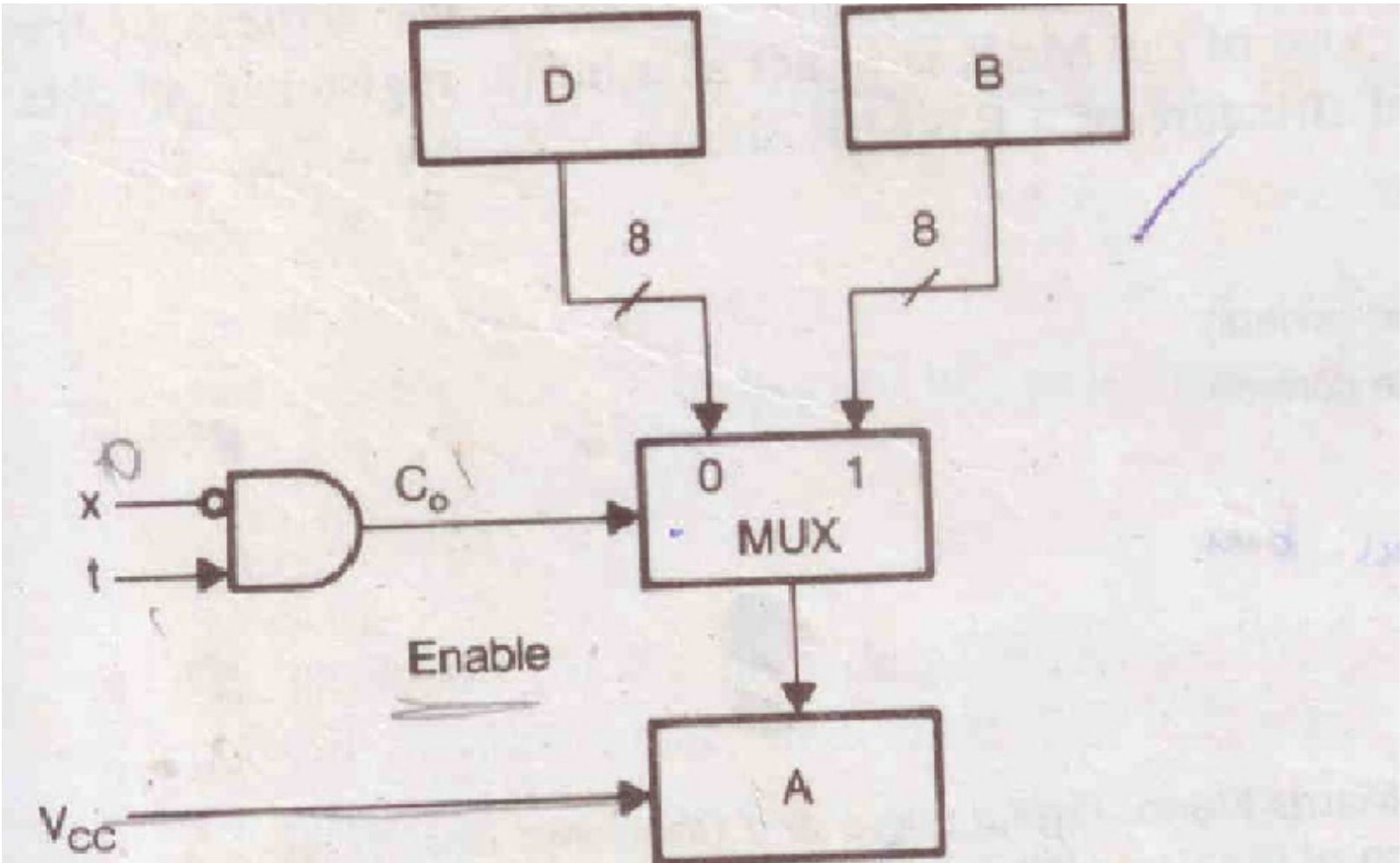
IF A>B and D[0]=0 then A←B



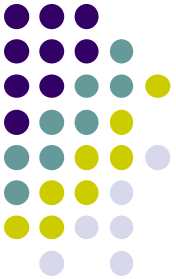
HW implementation

If x=0 and t=1, then A←B

else A←D



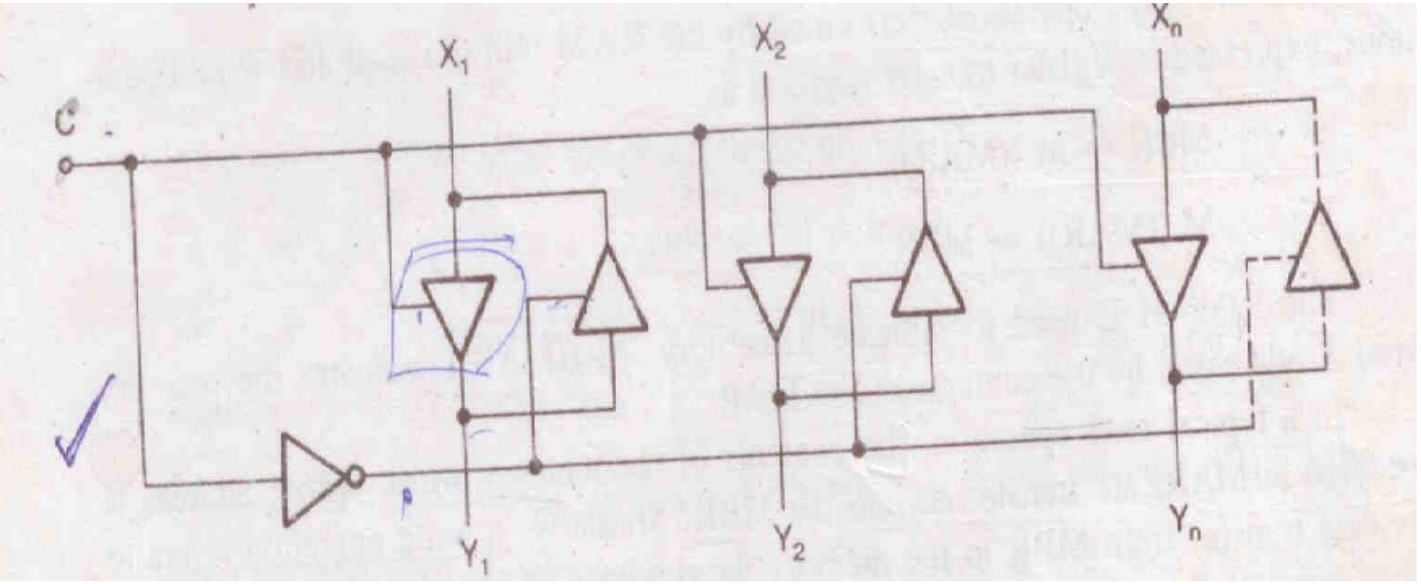
Register transfer description



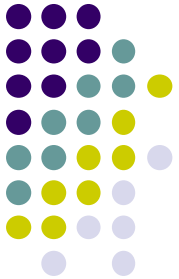
- D←A’
- D← AVB (A OR B, store result in D)
- D← AΛB (A AND B, store result in D)
- LSR(A)
- ASR(A)
- LSL, ASL, ROR, ROL
- A$Q – used to concatenate A and Q
  - ASR(A$Q)

RWM(Read Write Memory Unit)

- MBR and MAR are associated with RWM
- R: MBR← M((MAR))
- W: M((MAR))← MBR
- The line b/w RWM and MBR is bidirectional bus and it can be easily implemented using tristate buffers

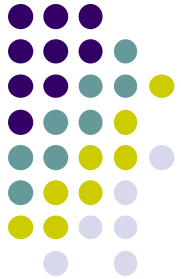


Tristate buffer



- **3-state** logic logic allows an output port to assume a high impedance logic allows an output port to assume a high impedance state in addition to the 0 and 1 logic levels, effectively removing the output from the circuit.

Buses



- Route data in and out of a digital system
- Normally 2 buses: inbus and outbus
- Notations
  - Inbus[4] and outbus[4] -- 4 bit buses
  - A=inbus (data of inbus is transferred into A register when next clock arrives
  - Outbus = B[7:4] (higher order 4 bits of an 8 bit register is made available on the outbus for one clock period)

Register Transfer instructions for multiplying(single bus)

Declare Registers A[8], M[8],Q[8];

Declare buses inbus[8] and outbus[8];

start: A←0, M←inbus;

Q← inbus;

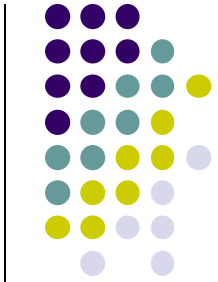
Loop: A←A+M, Q← Q-1;

if Q>0 then goto loop;

Outbus = A;

goto halt (stop)

Buses contd…



HW required

- 8 bit inbus, outbus, 8 bit parallel adder, 3 8 bit registers

Concurrent operations

- A← 0, M← inbus
- A← A+M, Q← Q-1

M← inbus and Q← inbus must be done serially

Microoperations: operations (such as A←0, A←A+M) that can be done in one cycle

Rate at which computer performs operations (such as A← A+M, A←AΛB) is determined by bus structure

Single bus structure is the simplest and cheapest

Single bus

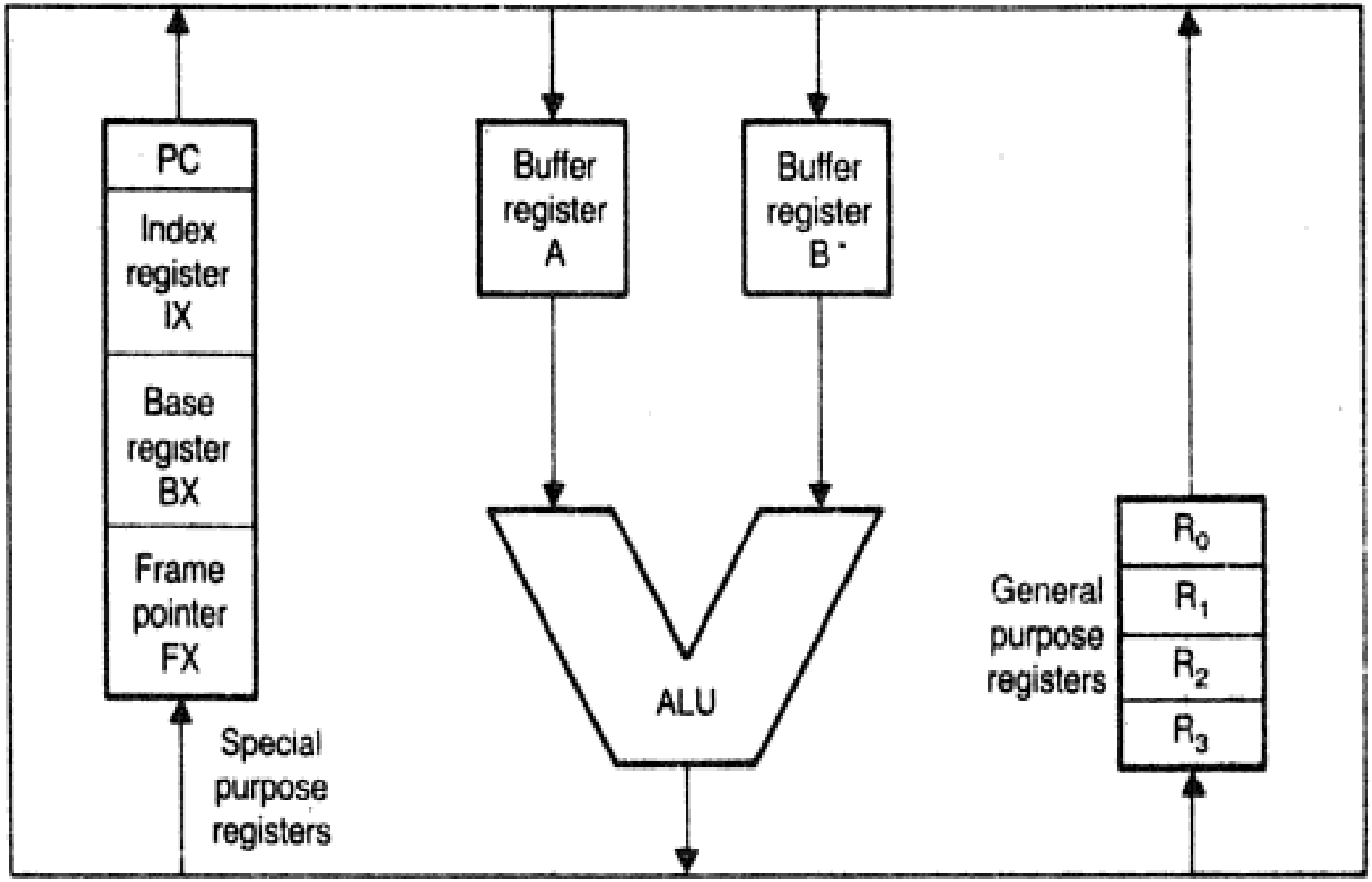


Figure 4.9 Organization of a Single Bus-oriented RALU

Single bus features

- At any given time, data may be transferred b/w 2 CPU registers or b/w a register and ALU
- Single bus architecture should have following features:
  - Bus must be multiplexed across various operands
  - ALU must have buffer registers to hold transferred operand
- Ex: R2← R1+R0 is completed in 3 clock cycles(3 control states)
  - 1st clock cycle: contents of R0 transferred to buffer register A of ALU
  - 2nd clock cycle: R1→ B
  - Sum produced by ALU is loaded into R2 , when 3rd clock pulse arrives

Disadvantages:

- Affects speed of execution of a typical 2 operand memory
- Increases no. of states in control logic. Hence more HW may be required to design control unit

Double bus

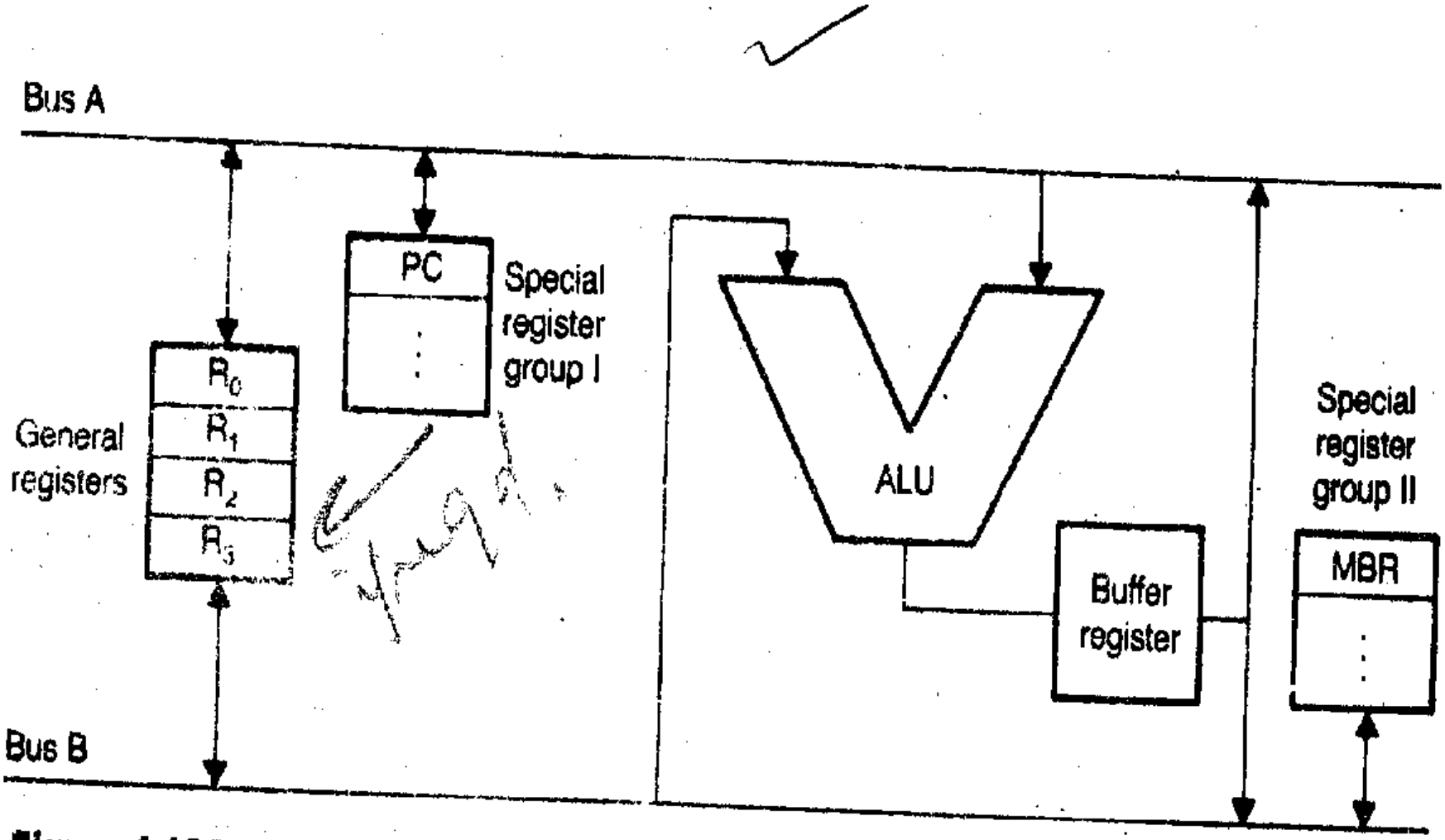


Figure 4.10 The Architecture of a Two-bus-oriented RALU

double bus contd…

Features:

- All general purpose registers are connected to both buses
- Both operands are routed in one cycle
- Special purpose registers divided into 2 groups and each group is connected to one of the buses
- Data from 2 special purpose registers of the same group cannot be transferred to the ALU at the same time
- Whenever there is a need to process simultaneously the contents of 2 special purpose registers of the same group, contents of one of the registers must be transferred to GPR(general purpose register) prior to processing.

double bus contd…

Features contd:

- 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)
- There can be dedicated paths b/w PC and MAR

3 Bus Structure

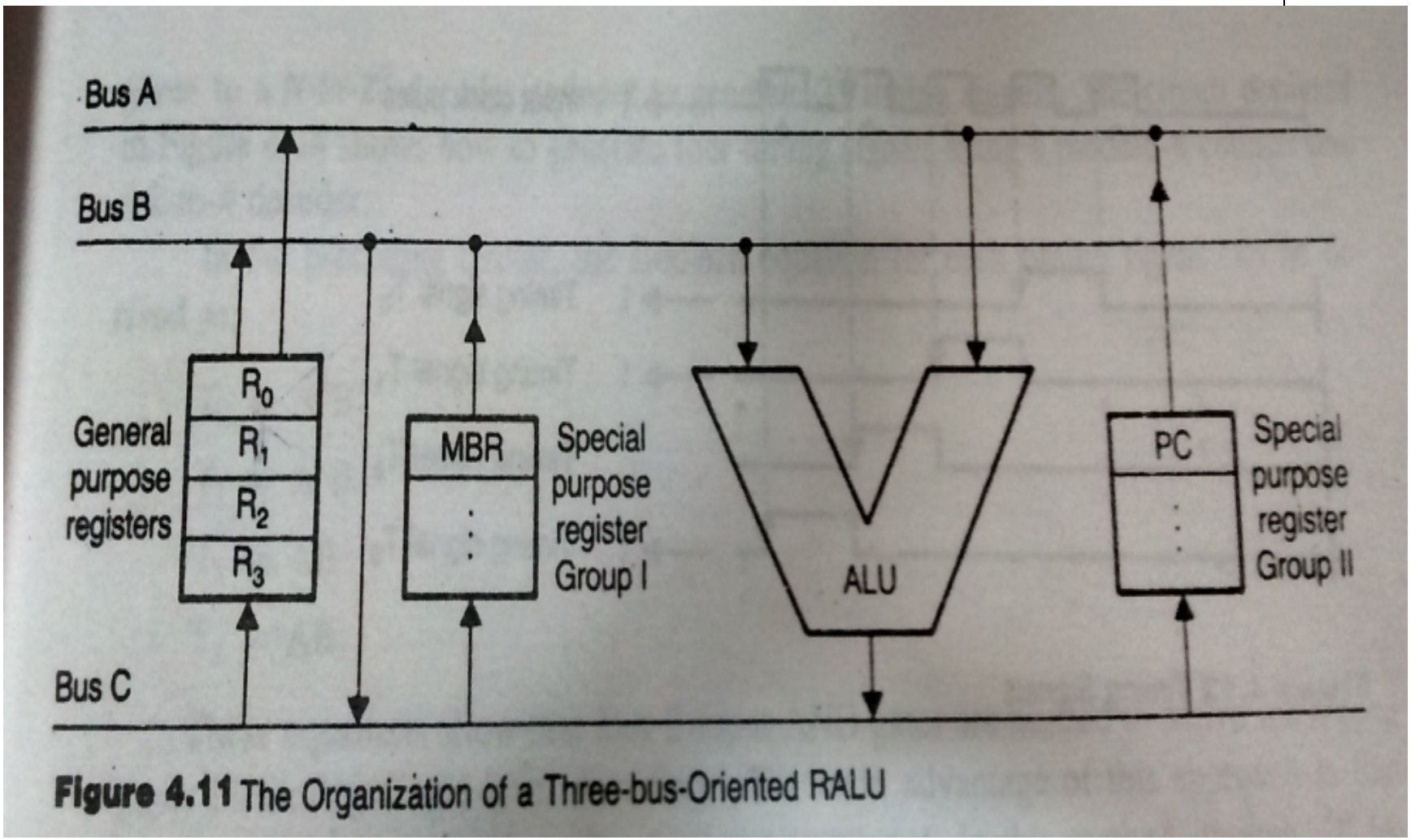
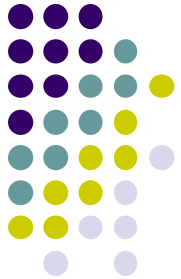


Figure 4.11 The Organization of a Three-bus-Oriented RALU

3 bus features



- Addition of bus C allows to perform ALU operations such as R2← R0+R1 in one cycle
- Increase system cost and control logic is complicated
- There can be small delays

Timing signals

- Control unit has to properly sequence a set of operations
- Sequence of N consecutive operations will occur in response to N consecutive clock pulses
- To carry out an operation Pi in ith clock pulse, CU must count the clock pulses and produce a timing signal Ti.
- Ti will assume value of 1 during the duration of the ith clock pulse

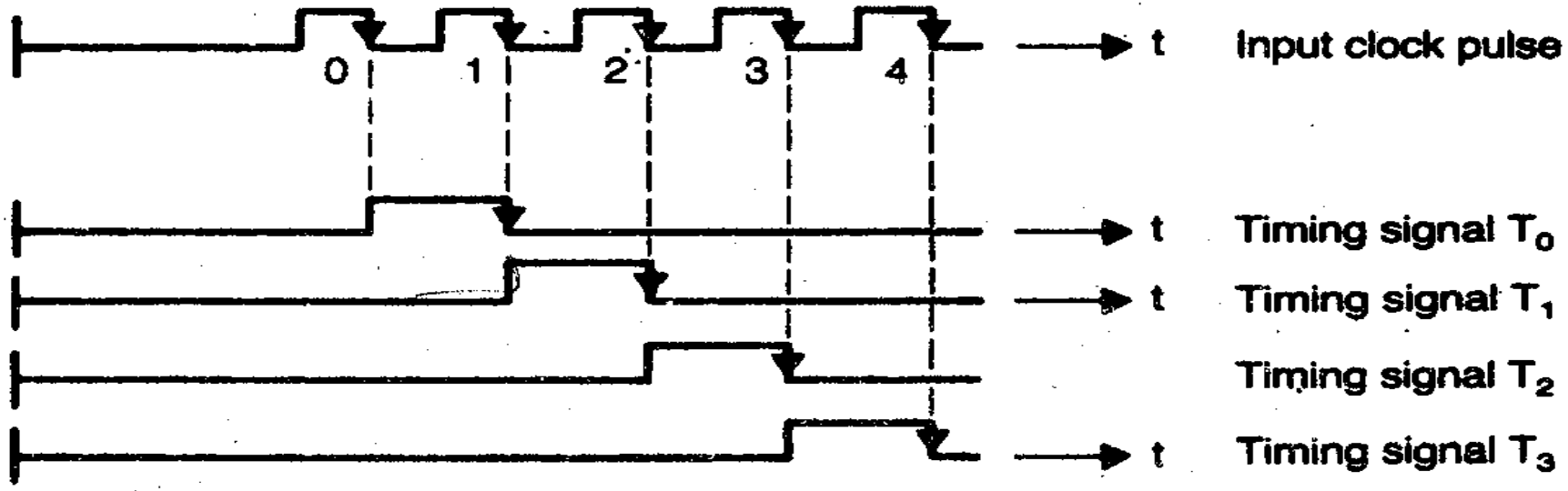


Figure 4.12 Timing Signals

Timing signals contd…

- Timing signals are generated using a ring counter

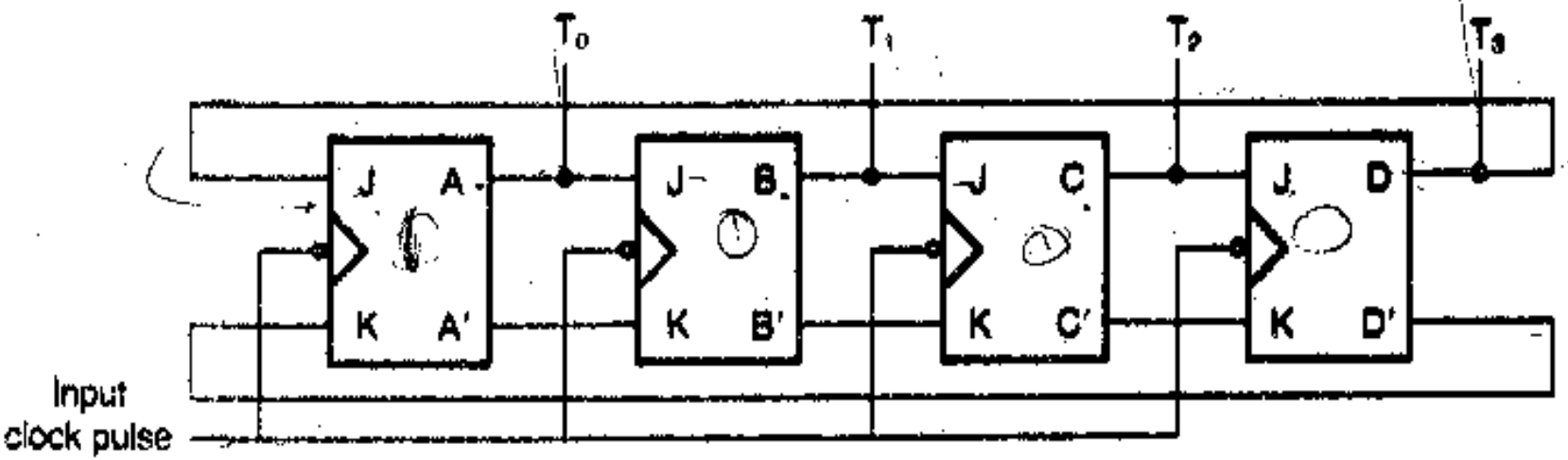
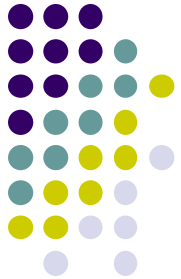


Figure 4.13 Ring Counter

- This system will sequence the following manner

PRESENT STATE				NEXT STATE			
A	B	C	D	A^+	B^+	C^+	D^+
1	0	0	0	0	1	0	0
0	1	0	0	0	0	1	0
0	0	1	0	0	0	0	1
0	0	0	1	1	0	0	0

Timing signals contd…

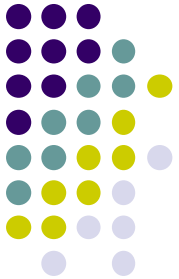


- Boolean equations for each timing variable are T0=A, T1=B, T2=C, T3=D

Disadvantage:

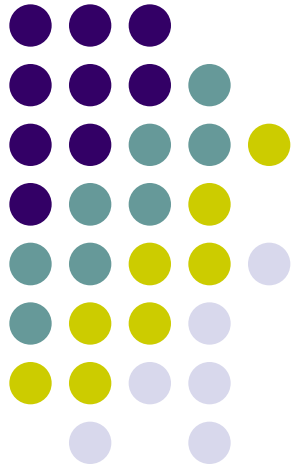
- N flip flops are required to generate N timing signals
- Not economically feasible for large values of N

Modulo-4 Counter with decoder

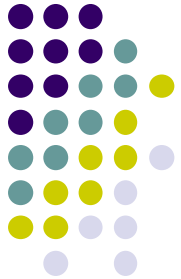


- To generate timing signals economically
- Modulo 2^N counter designed using N flip flops
- N outputs from this counter are given to a N to 2^N decoder as input to generate 2^N timing signals

Design methods

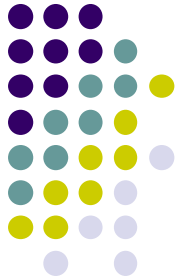


Overview



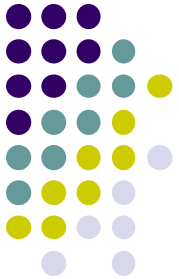
- To execute instructions, the processor must have some means of generating the control signals needed in the proper sequence.
- Two categories: hardwired control and microprogrammed control
- Hardwired system can operate at high speed; but with little flexibility.

Hardwired vs Microprogrammed



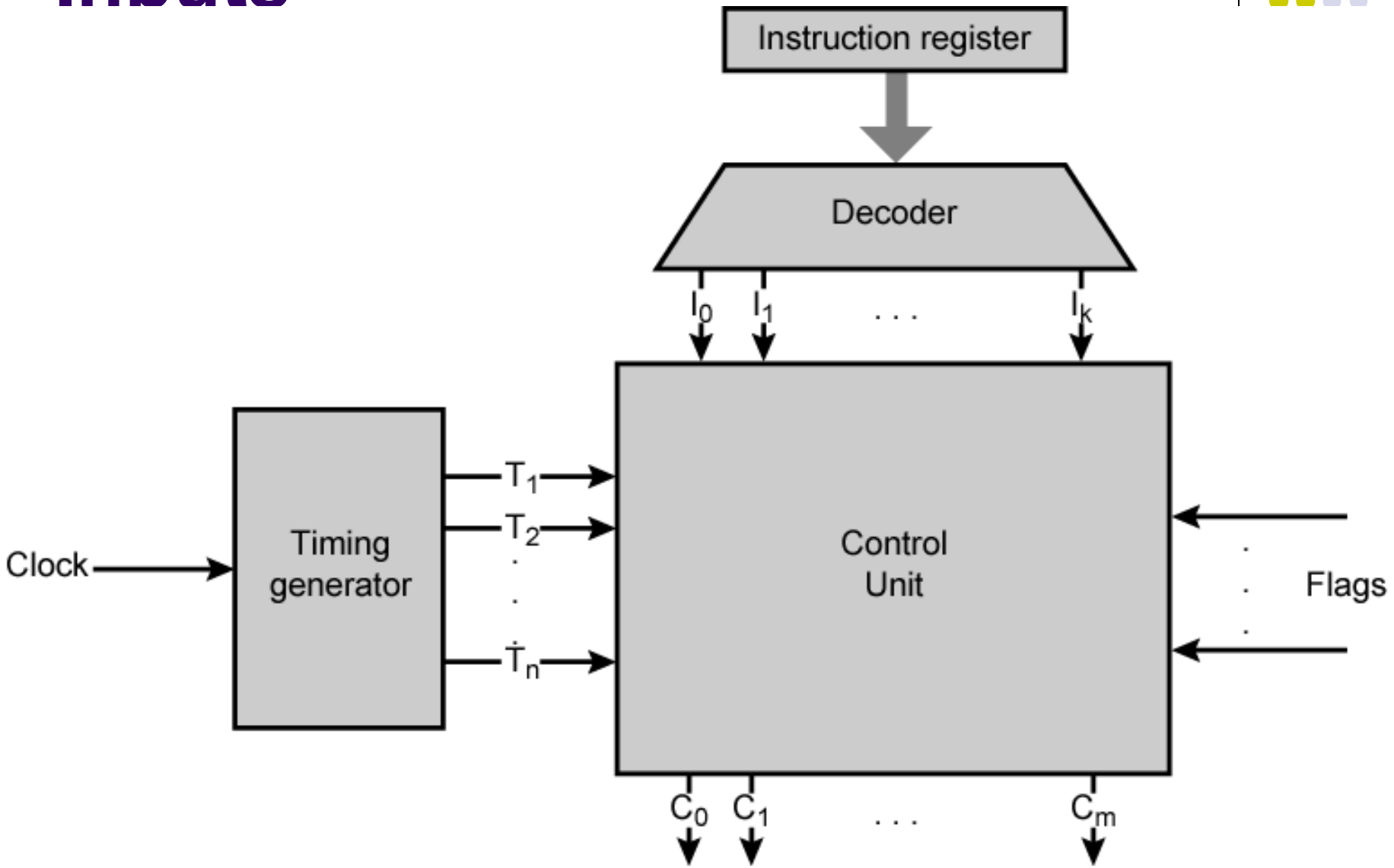
- Hardwired
  - Use gates to generate signals
  - Squeeze out the juice for performance(not flexible)
  - Different logic styles possible
  - Economical initially
  - Small change→redesign
- Microprogrammed
  - Store the control signals in the sequence
  - Just read from the memory every clock cycle
  - Expensive initially
  - Additions done by simply changing the microprogram in control memory
  - Diagnostics routine can be made available in memory

HARDWIRED APPROACH

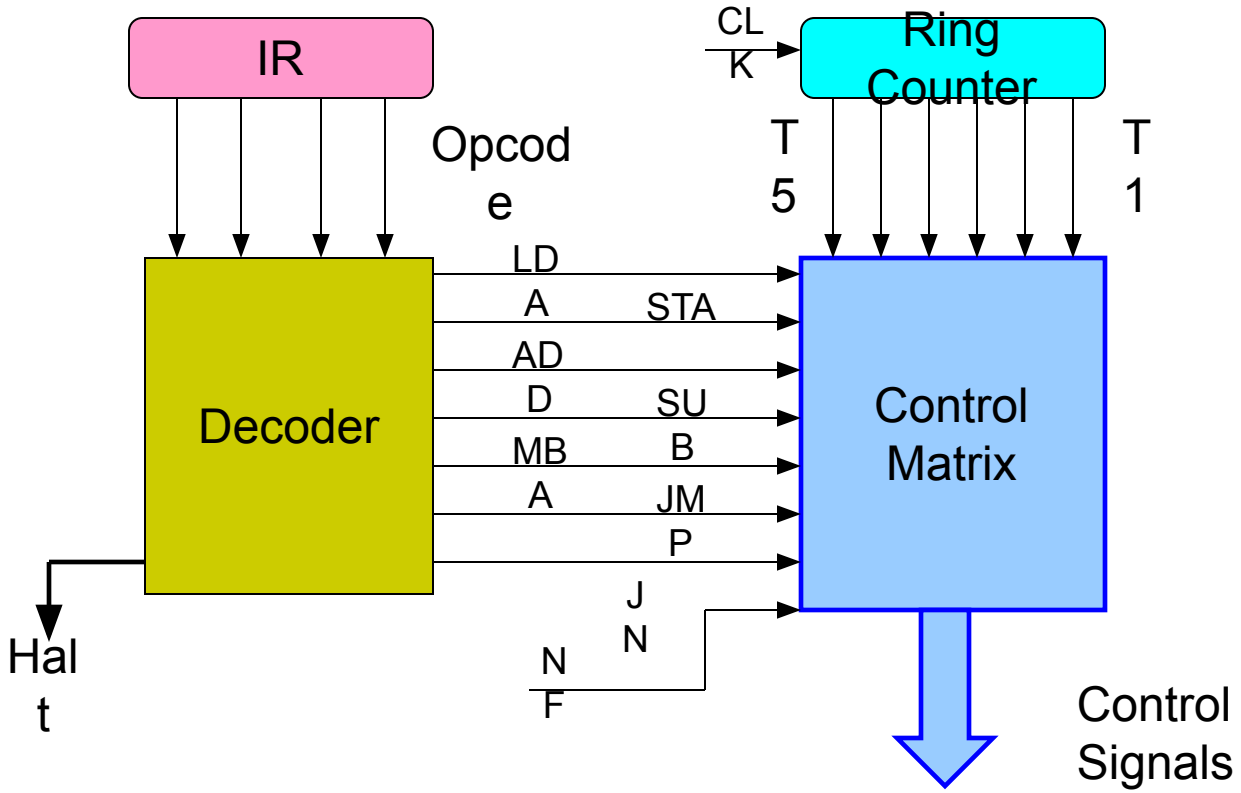


- Final circuit is obtained by physically connecting gates and flip flops
- Cost of control logic increases with system complexity

Control Unit with Decoded Inputs



Hardwired Unit



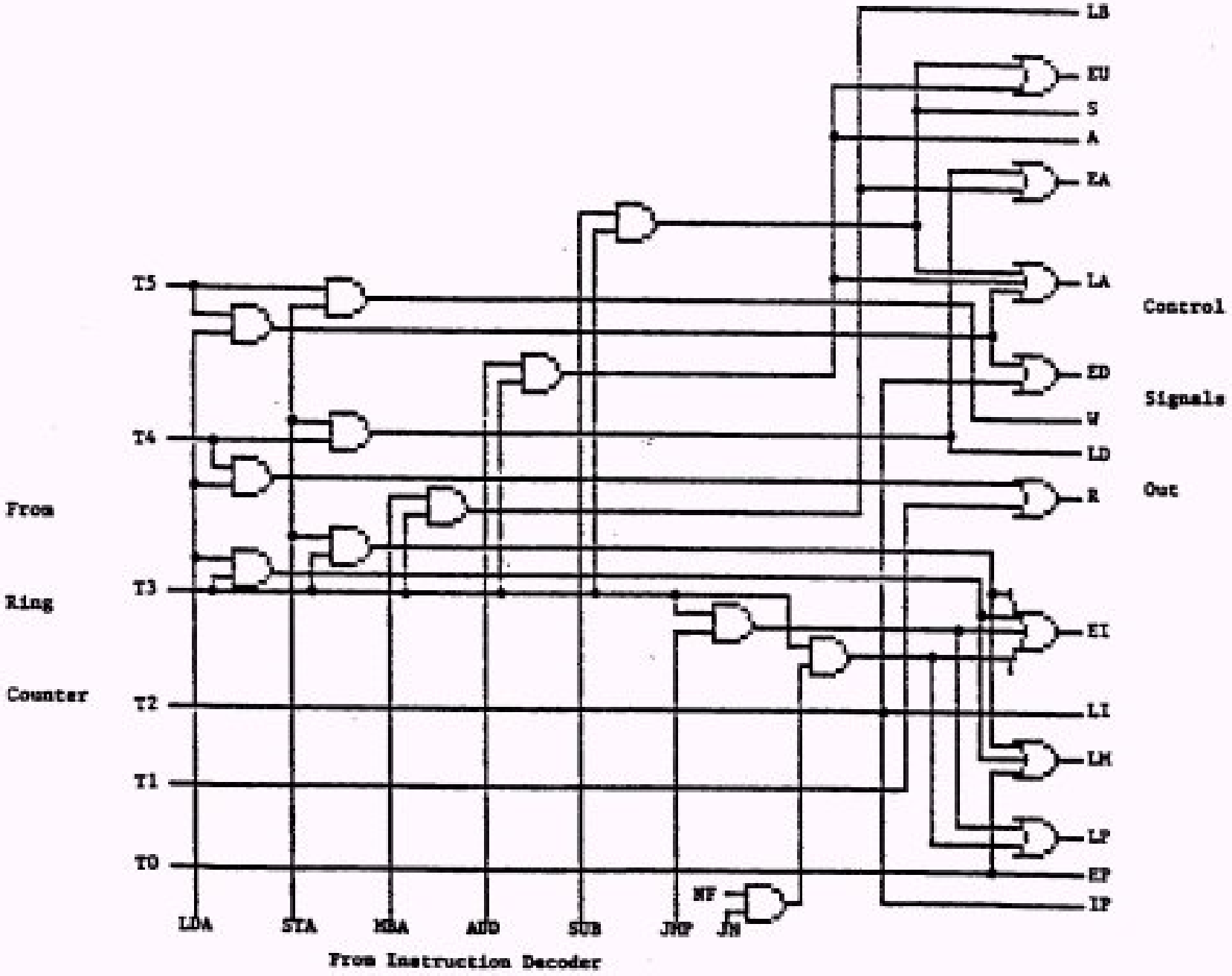
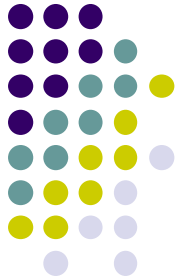


Figure 5. The hard-wired control matrix.

10 steps for hardwired control



1) Define task to be performed
2) Propose a trial processing section
3) Provide a register transfer description of the algo based on the processing section outlined in the previous step
4) Validate the algo by using trial data
5) Describe the basic characteristics of the HW 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 HW elements to be used in the controller
10) Complete the controller design and draw a logic diagram of final circuit

Ex: booth's multiplier to multiply 2 4-bit 2's complement no

Step 1: Task definition

Design a Booth's multiplier to multiply two 4-bit 2's complement numbers

Step 2: trial processing section

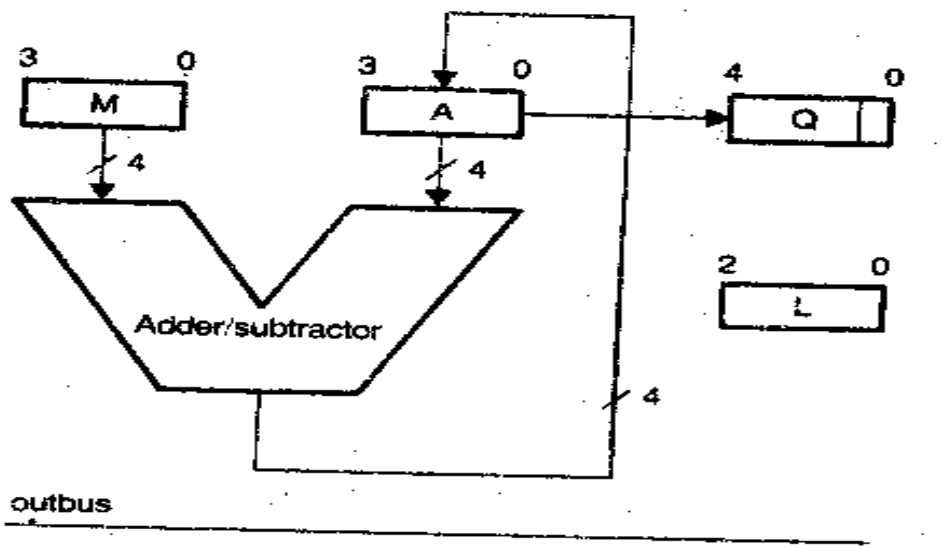
q_1 q_0

0 0→none

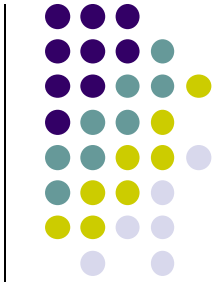
0 1→ add M

1 0→sub M

1 1→ None



Contd..



Step 3: register transfer description

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(AQ), L←L-1;

if L>0, then go to loop

outbus =A;

outbus=Q[4:1];

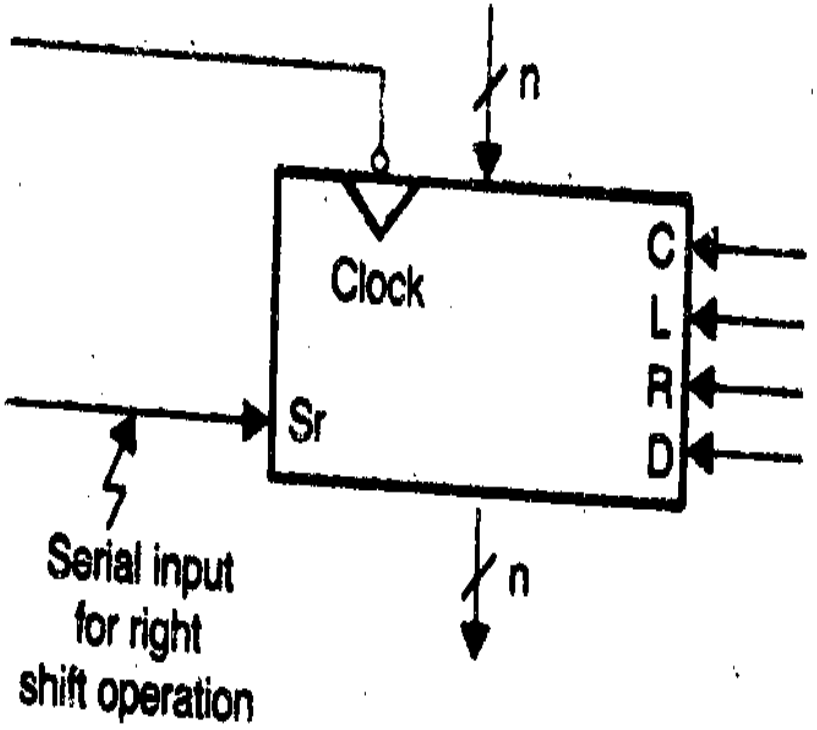
go to halt

Step 4: validate algo 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

Contd..

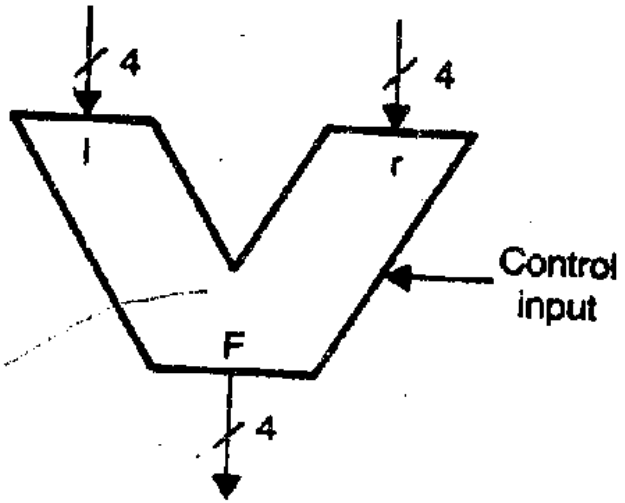
Step 5: functional characteristics of HW elements of processing section



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

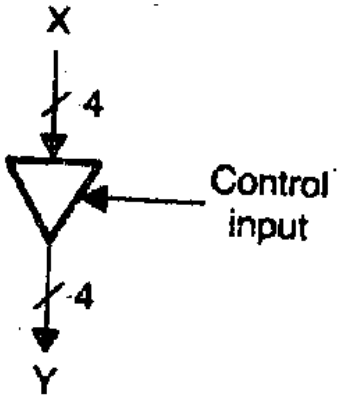
a. Storage Register

Step 5 contd…



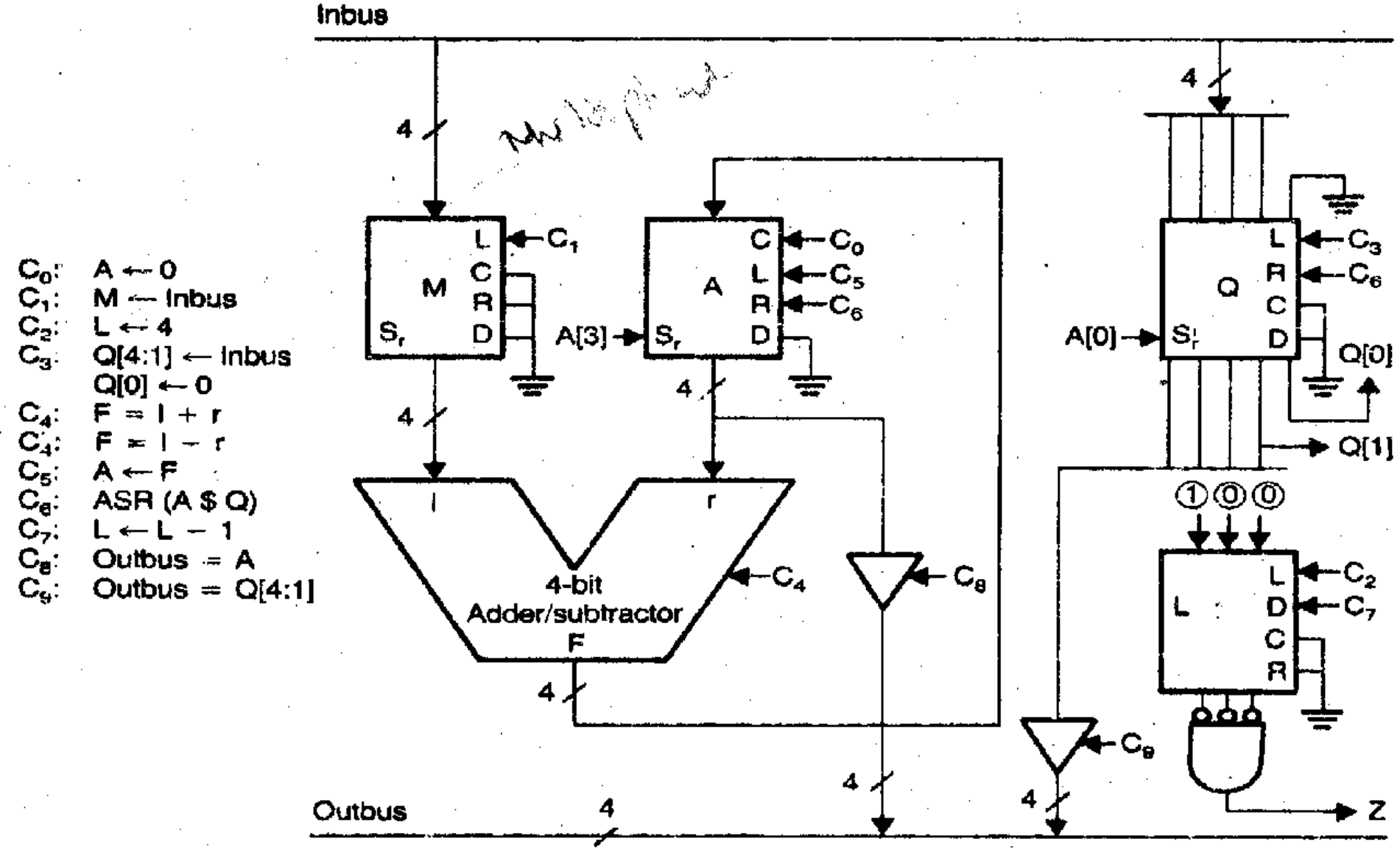
Control input	F
1	l + r
0	l − r

b. Adder/subtractor

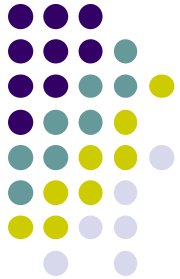


Control input	Y
1	X
0	High Z

Step 6: establishing control points of processing section

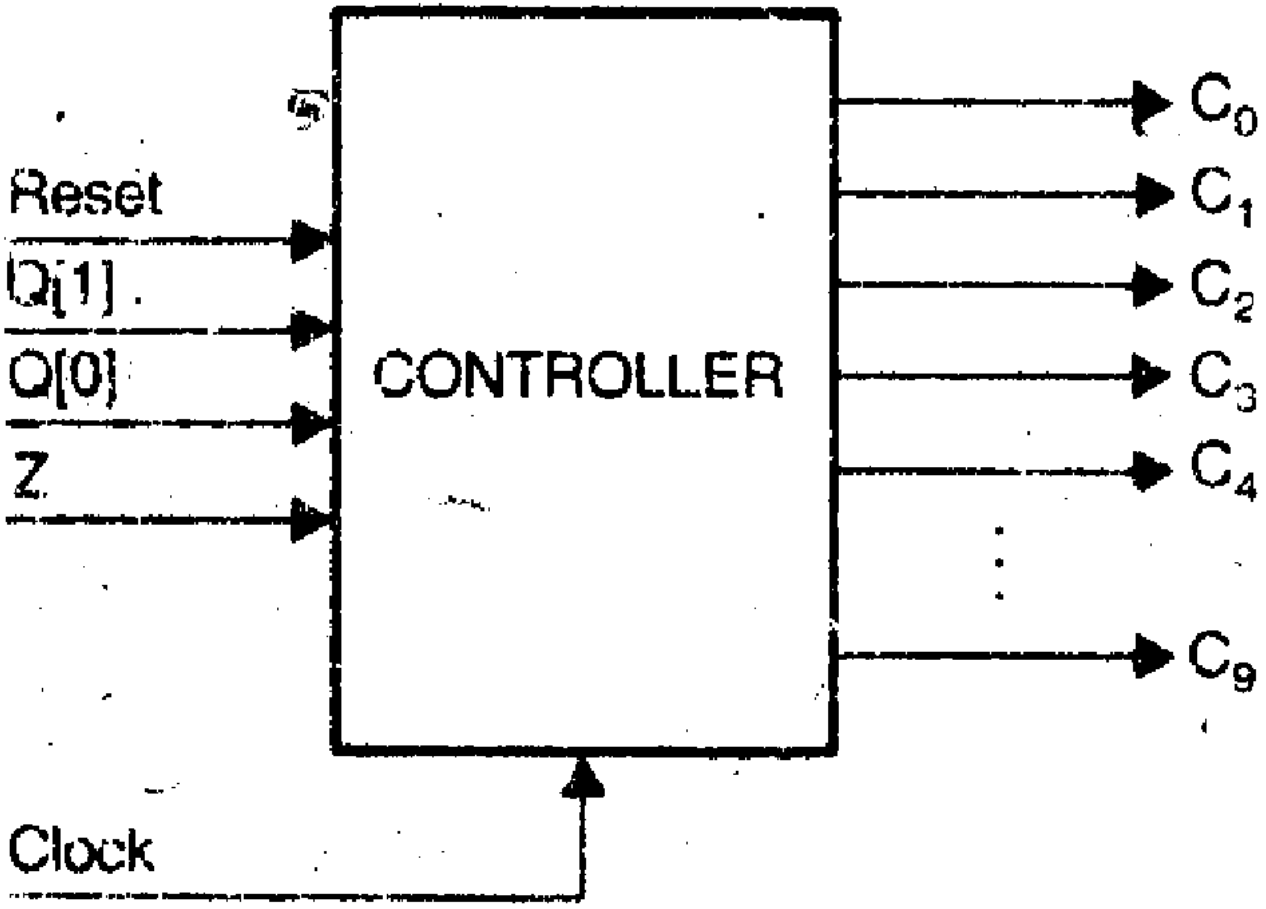


Step 6 contd…

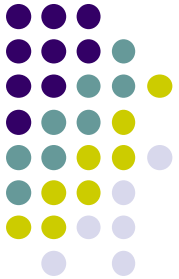


- There are 10 control points
- C0 held high → A reg is cleared with trailing edge of next clock
- 1 control point is introduced for each microoperation specified in the register transfer description
- Processing section extends 3 outputs Q[1], Q[0] and Z (decision making)
- Z=1 only when contents of the L register becomes zero
- Q[1], Q[0], Z are the STATUS outputs and are used as inputs to the controller to allow the controller to decide the future course of action

Step 7: block diagram of controller

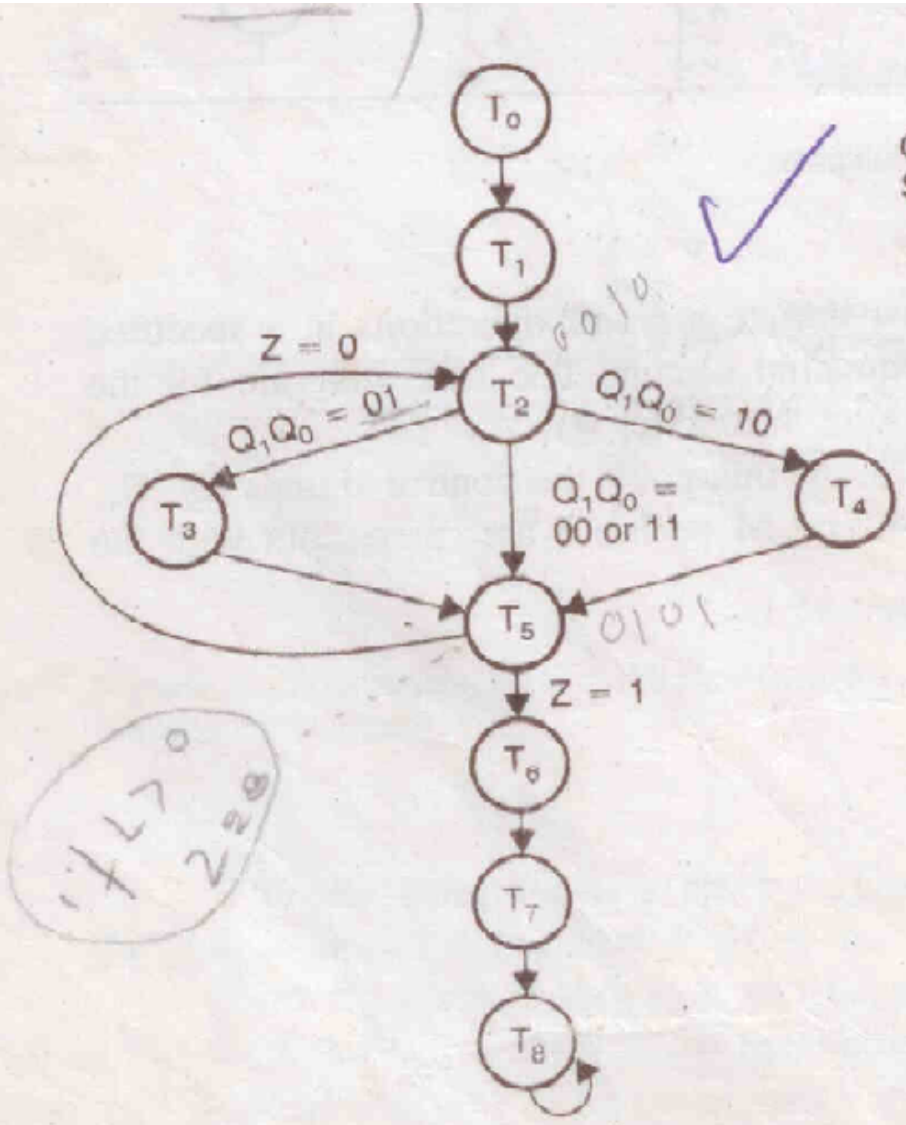


Contn



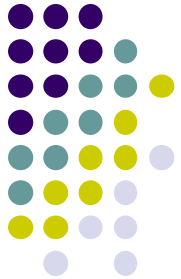
- 5 inputs to controller and 10 outputs
- RESET input is an asynchronous input – used to reset controller so a new computation can begin
- Clock input – used to synchronize the controller's action
- A controller must initiate a set of operations in a specified sequence, therefore it is modeled as Sequential circuit

Step 8: state diagram of controller



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 (A$Q), $L \leftarrow L - 1$	C_6, C_7
T_6	Outbus = A	C_8
T_7	Outbus = Q [4:1]	C_9
T_8	None	None

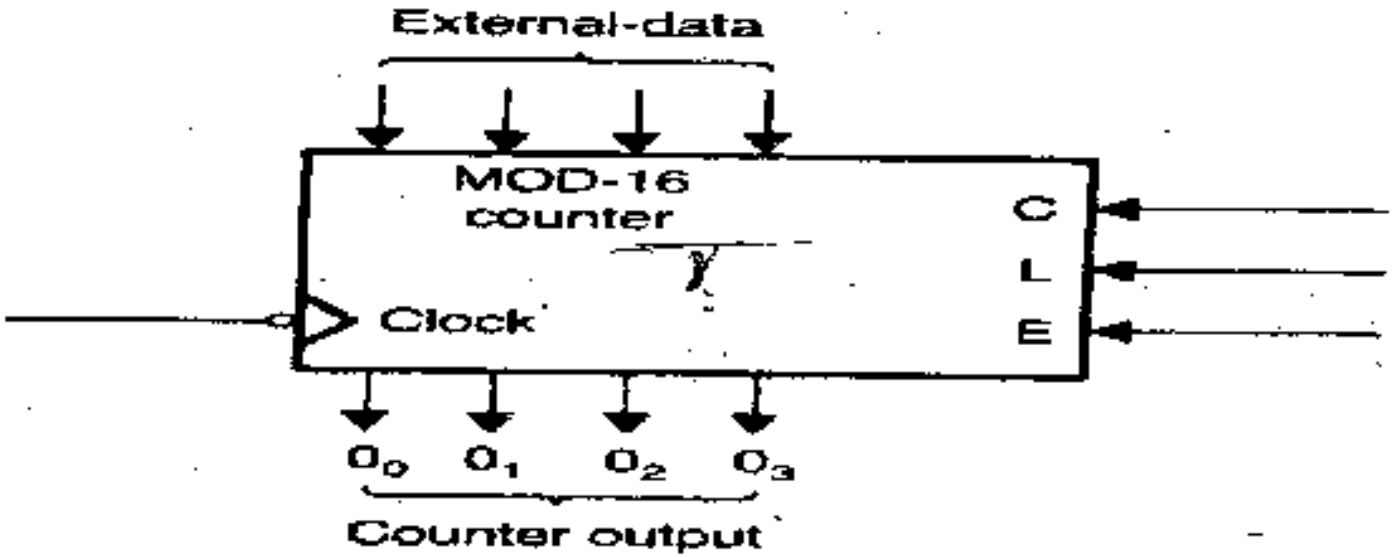
state diagram contd…



- There are 9 states and hence 9 non overlapping timing signals have to be generated
- Mod 16 counter and a 4 to 16 decoder can be used to accomplish this task

Step 9: characteristics of HW elements used in controller

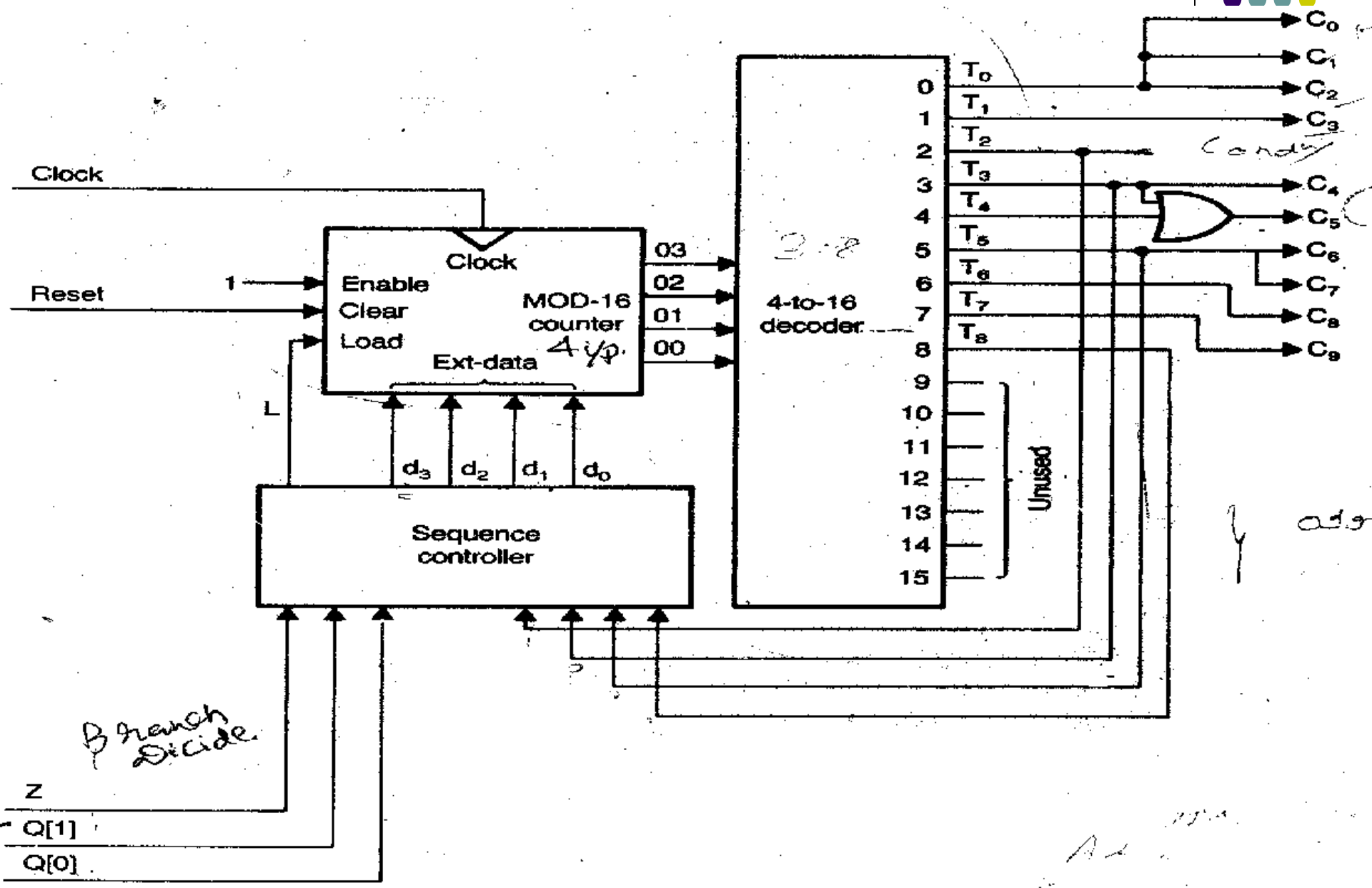
Mod 16 counter



a. Block Diagram

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

Step 10: logic diagram for controller



Step 10 contd…(truth table for SC)

- Sequence controller sequences the controller as indicated in state diagram
- Truth table for SC

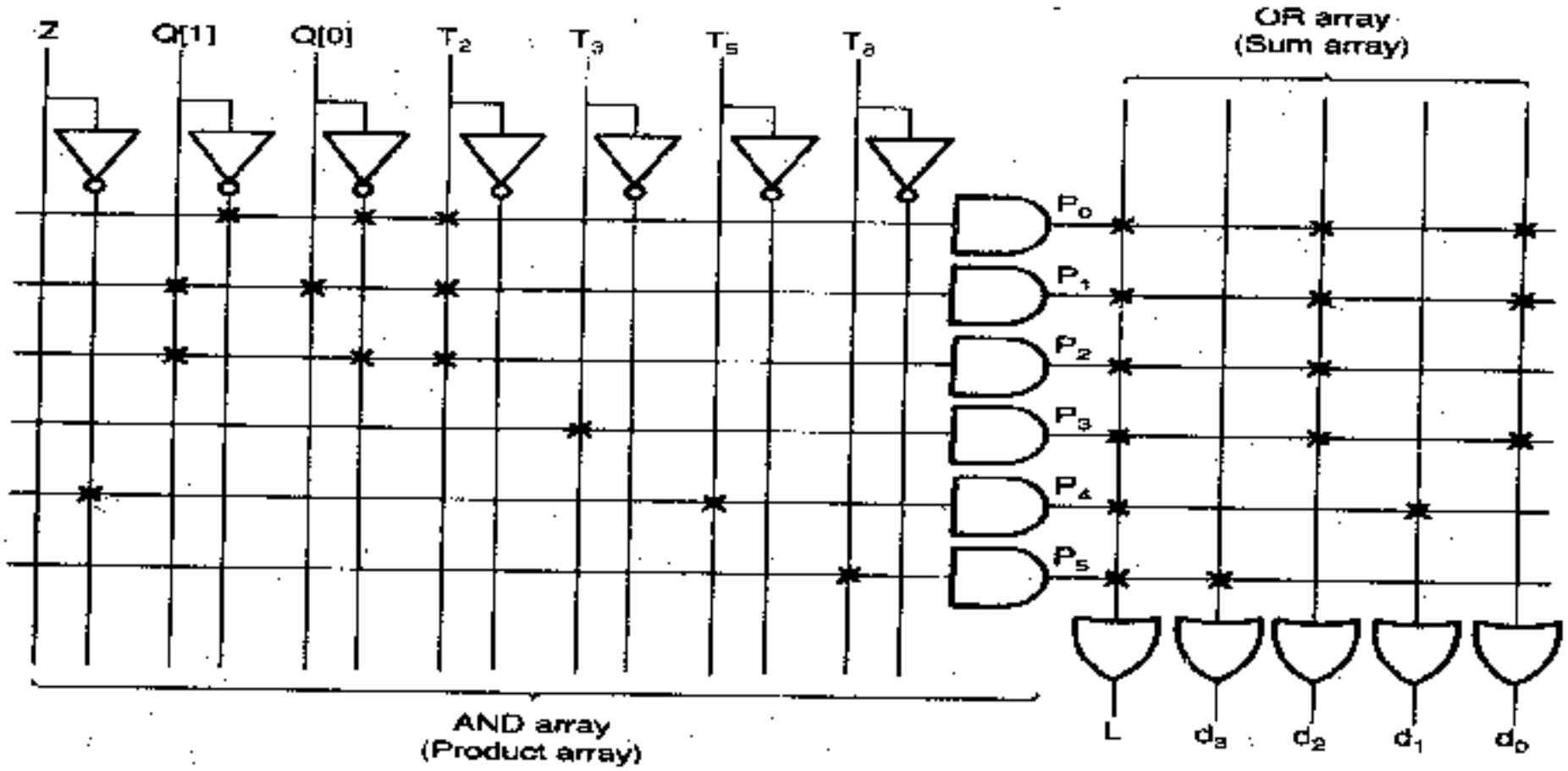
Z	Q [1]	Q [0]	T_2	T_3	T_5	T_8	L	External-data d3	d2	d1	d0
X	0	0	1	X	X	X	1	0	1	0	1
X	1	1	1	X	X	X	1	0	1	0	1
X	1	0	1	X	X	X	1	0	1	0	0
X	X	X	X	1	X	X	1	0	1	0	1
0	X	X	X	X	1	X	1	0	0	1	0
X	X	X	X	X	X	1	1	1	0	0	0

Step 10 contd…

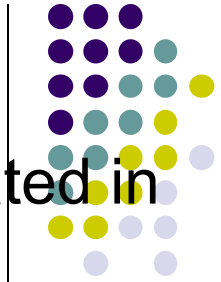
- When L is 0, counter will automatically count up in response to next clock pulse.
- Hence counter is a normal up counter and should change normal operation only during certain conditions.
- Such normal sequencing is required
  - Present control state is T0,T1,T4,T6 OR T7
  - Present control state is T2 and Q[1] Q[0]=01
  - Present control state is T5 and Z=1
- Hence the SC must exercise control only when there is a need for the counter to deviate from its normal counting sequence

PLA for designing SC

- Though SC has 8 inputs(256 combinations) , it must examine only a few possibilities
- Designers use programmed logic array (PLA) in this situation



PLA contd…



- For each row of SC truth table, product term is generated in PLA

P0= Q[1]’ Q[0]’ T2

P1= Q[1] Q[0] T2

P2= Q[1] Q[0]’ T2

P3=T3

P4=Z’T5

P5=T8

L= P0+P1+P2+P3+P4+P5

d3= P5

d2= P0+P1+P2+P5

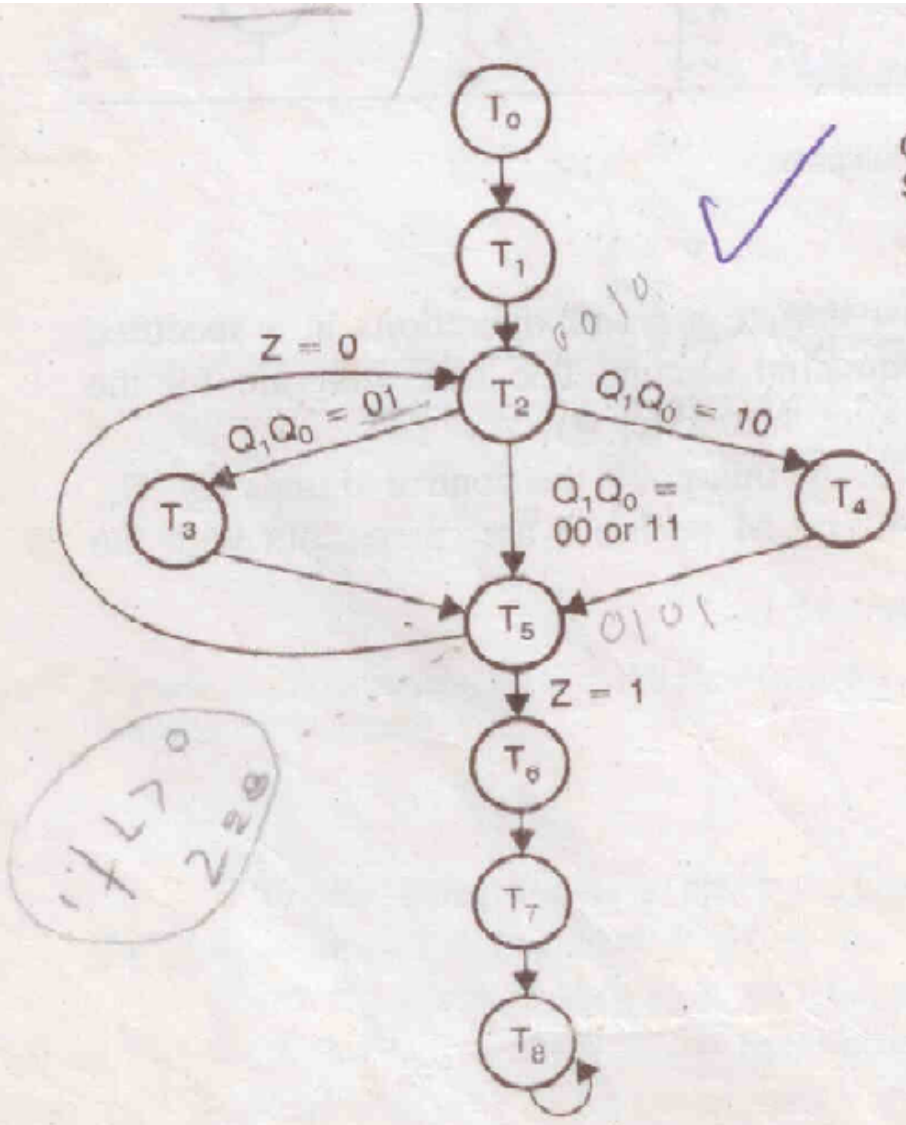
d1=P4

d0=P0+P1+P3

PLA contd…

- The controller design is completed by relating the control states (T0 through T8) with control signals(C0 to C9)
  - C0=C1=c2=T0
  - C3=T1
  - C4=T3
  - C5=T3+T4
  - C6=C7=T5
  - C8=T6
  - C9=T7

Step 8: state diagram of controller

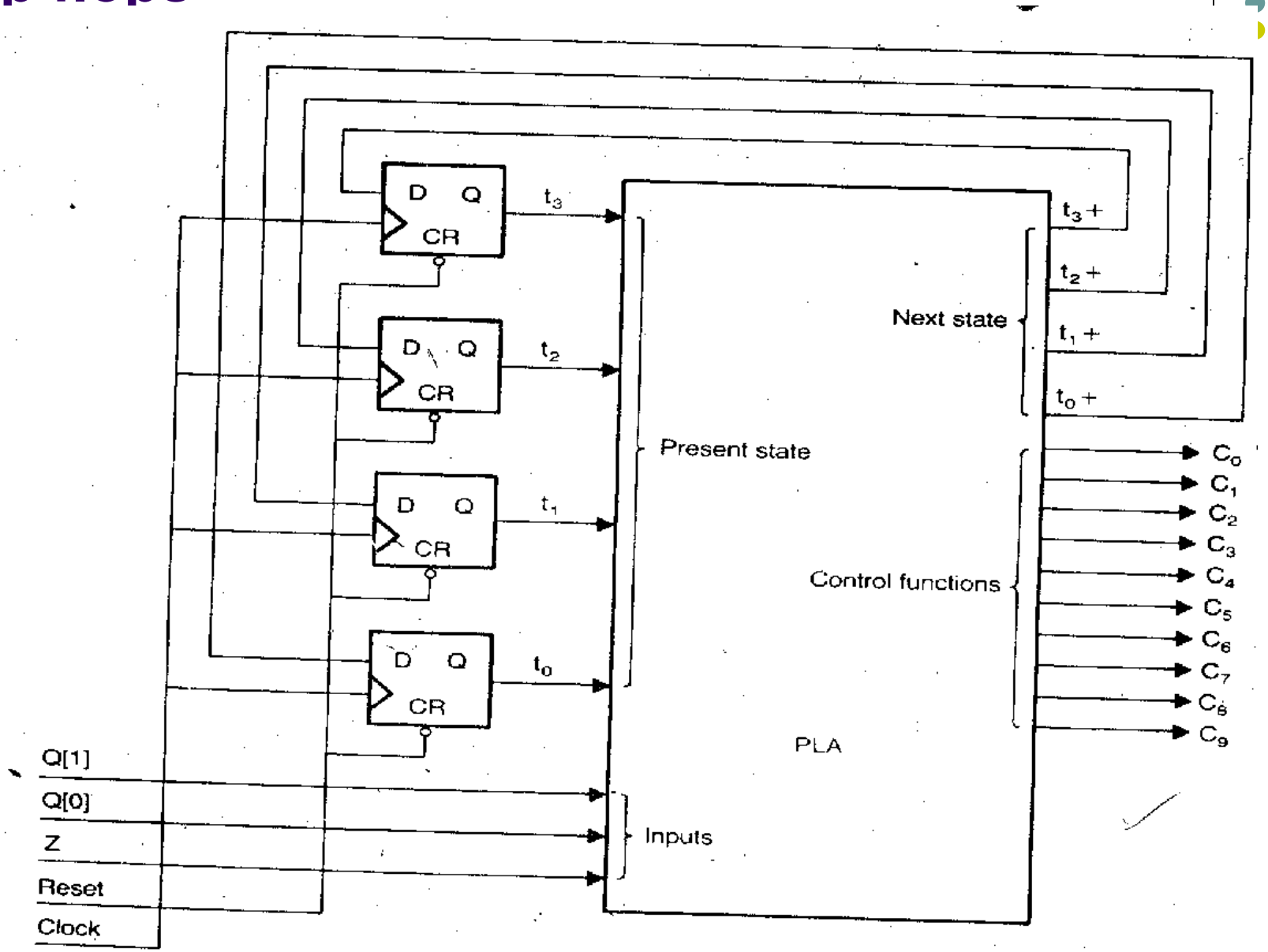


CONTROL STATE	OPERATION PERFORMED	CONTROL SIGNALS TO BE ACTIVATED
T_0	$A \leftarrow 0$, $L \leftarrow 4$, $M \leftarrow$ Inbus	C_0, C_1, C_2
T_1	$Q[4:1] \leftarrow$ Inbus, $Q[0] \leftarrow 0$	C_3
T_2	None	None
T_3	$A \leftarrow A + M$	C_4, ~~C_5~~
T_4	$A \leftarrow A - M$	C_5 ($C_4 = 0$)
T_5	ASR (A$Q),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

PLA table(controllers state table)

Present state is	PLA Input							PLA Output												
	Present state in binary				Inputs			Next state (in binary)				Control functions								
	t_3	t_2	t_1	t_0	Q [1]	Q [0]	Z	t_3^+	t_2^+	t_1^+	t_0^+	C_0	C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
T_0	0	0	0	0	X	X	X	0	0	0	1	1	1	1	0	0	0	0	0	0	0
T_1	0	0	0	1	X	X	X	0	0	1	0	0	0	0	1	0	0	0	0	0	0
T_2	0	0	1	0	0	1	X	0	0	1	1	0	0	0	0	0	0	0	0	0	0
T_2	0	0	1	0	0	0	X	0	1	0	1	0	0	0	0	0	0	0	0	0	0
T_2	0	0	1	0	1	1	X	0	1	0	1	0	0	0	0	0	0	0	0	0	0
T_2	0	0	1	0	1	0	X	0	1	0	0	0	0	0	0	0	0	0	0	0	0
T_3	0	0	1	1	X	X	X	0	1	0	1	0	0	0	0	1	1	0	0	0	0
T_4	0	1	0	0	X	X	X	0	1	0	1	0	0	0	0	0	1	0	0	0	0
T_5	0	1	0	1	X	X	0	0	0	1	0	0	0	0	0	0	0	1	1	0	0
T_5	0	1	0	1	X	X	1	0	1	1	0	0	0	0	0	0	0	1	1	0	0
T_6	0	1	1	0	X	X	X	0	1	1	1	0	0	0	0	0	0	0	0	1	0
T_7	0	1	1	1	X	X	X	1	0	0	0	0	0	0	0	0	0	0	0	0	1
T_8	1	0	0	0	X	X	X	1	0	0	0	0	0	0	0	0	0	0	0	0	0

controller design using single PLA and 4 D flip flops



PLA contents

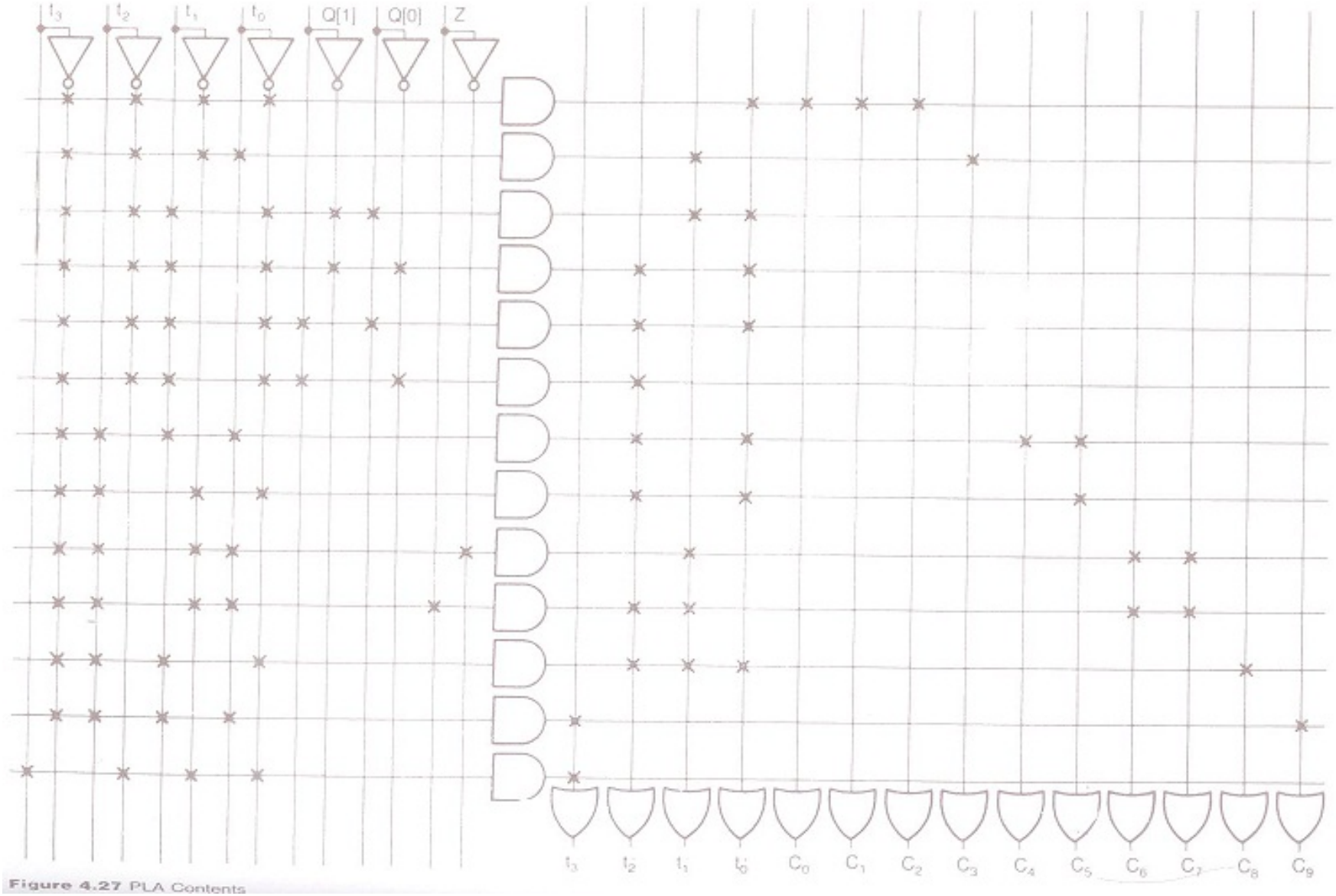
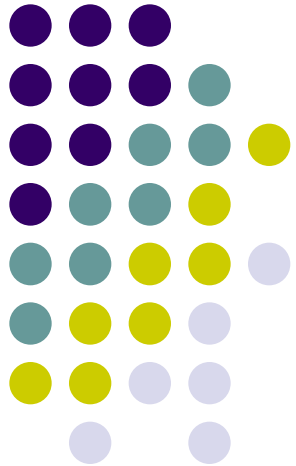


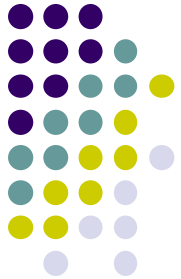
Figure 4.27 PLA Contents

Microprogrammed Control

- Basics
- Architecture of microprogrammed control unit
- Microprogrammed CU for booths multiplication

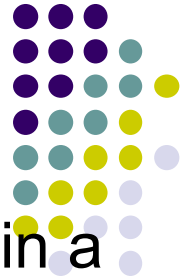


Review of Microprogramming Model



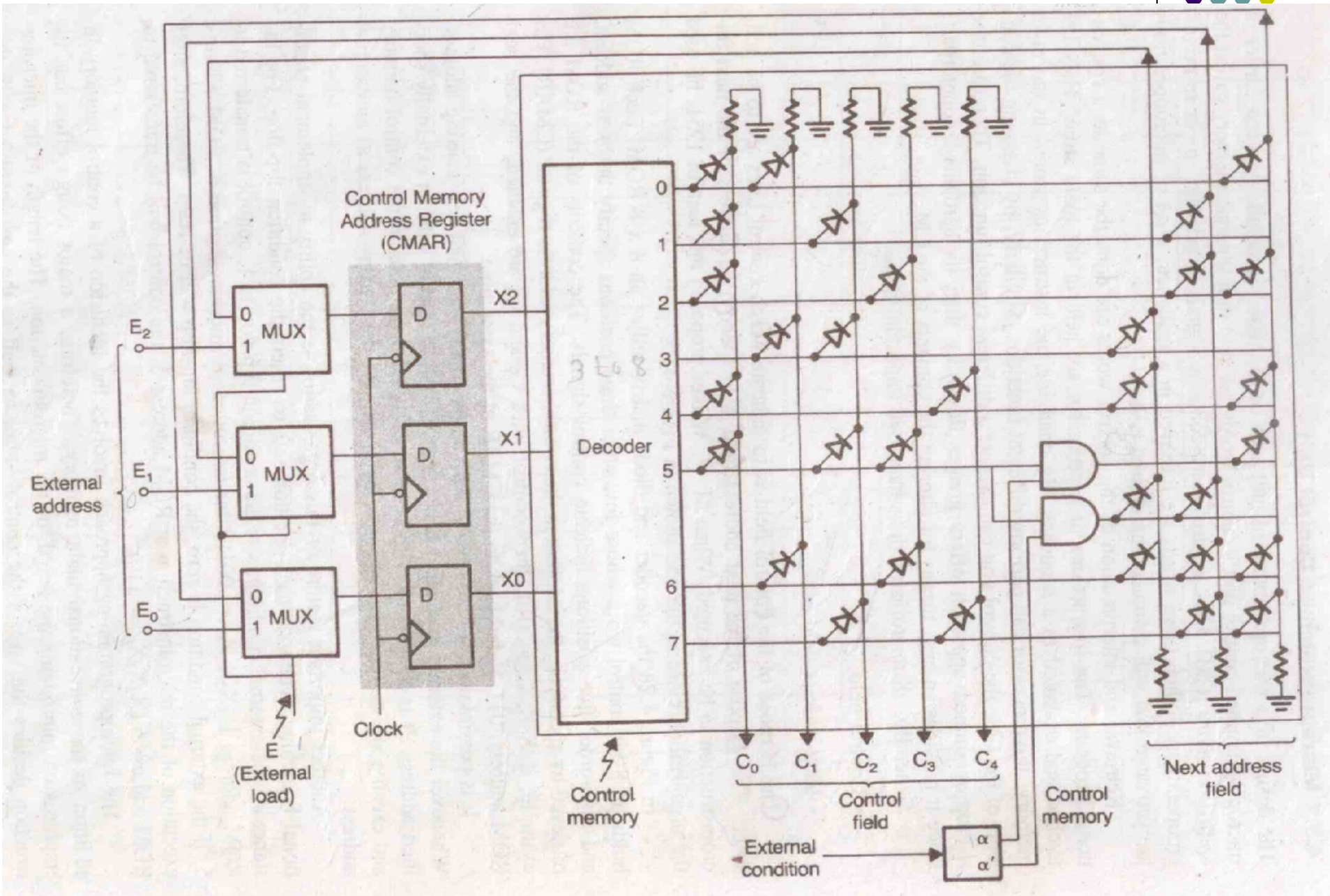
- Store the microprogram in control store
- Fetch the instruction
- Get the set of control signals from the control word
- Move the microinstruction address

Microprogrammed control unit



- Microprogrammed control unit's control words are held in a separate memory called control memory(CM)
- Each control word contains signals to activate one or more microoperations
- When these words are retrieved in a sequence , a set of microoperations are activated that will complete the desired task
- By changing the contents of CM, the CU can execute different control function
- This approach offers greater flexibility
- All microinstructions have 2 imp fields
  - Control field (which control lines to be activated)
  - Next address field (specify the address the next microinstruction to be executed)

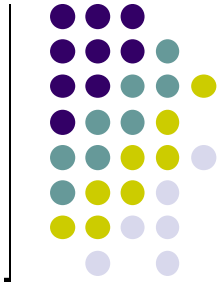
M V Wilkies proposal(referenece)



Features of wilkies

- If x2 x1 x0 = 010, control lines c0 c3 are enabled and so on’
- If E is set to 1, external load is supplied. Hence any desired microprogram can be executed by specifying starting address as external address
- Ability of implement conditional branching
  - External condition sets or resets the conditional flip flop
  - If flip flop set to 1, control transferred to ROM address 1 ,after ROM address 5,
  - Otherwise normal execution
- In microprogramming major design effort is to reduce the length of microinstruction
- Length is related to
  - Degree of parallelism
  - Control field organization
  - Method by which address of next microinstruction is specified

Microprogramming features



Degree of parallelism

All microoperations executed in parallel can be specified in a single microinstruction with a common opcode

Control field organization

Trivial way is to have 1 bit for each control line that controls the data processor (unencoded format)

Ex: registers A,B,C,D communicating with outbus when appropriate control line is activated

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

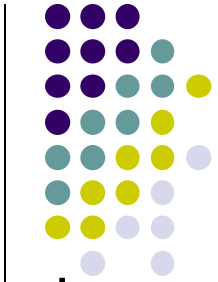
Microprogramming features contd..

- The 5 distinct patterns can be represented in just 3 bits
- This is called encoded format as control info is encoded into a 3 bit field

E2	E1	E0	
0	0	0	no op
0	0	1	outbus=A
0	1	0	outbus=B
0	1	1	outbus=C
1	0	0	outbus=D

- Decoder is required to decode(above case 3 to 8) the info
- 15 control lines can be encoded using a 4 to 16 decoder
- Microinstructions are classified into 2 groups
  - Horizontal (long, high parallelism, little encoding)
  - Vertical (short, limited parallelism, high encoding)

Microprogramming features contd..



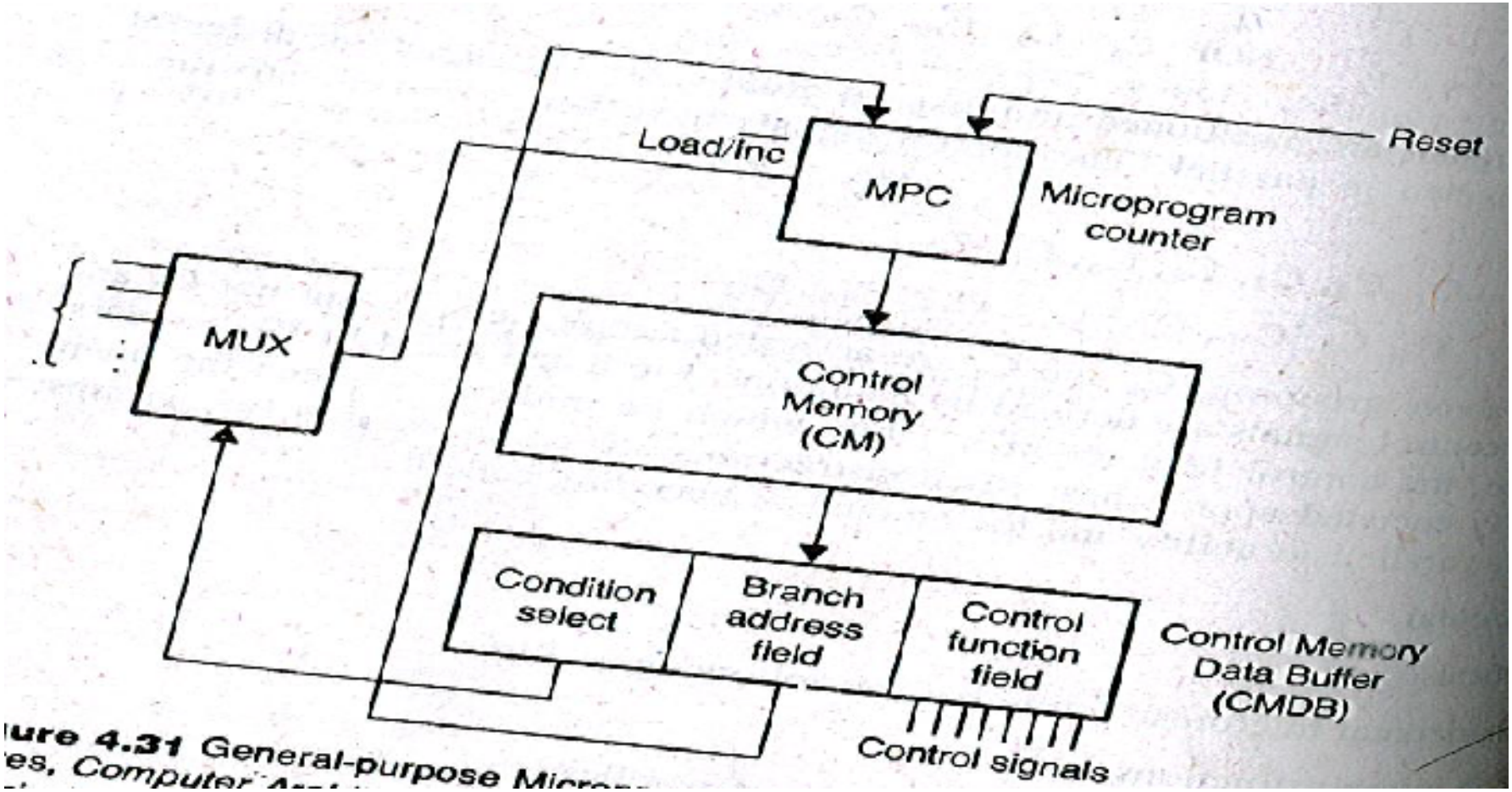
Specifying next address :

In original wilkies proposal, next address is specified in each microinstruction

But except in case of branch instruction, next address is the address that follows current memory word

Hence a register MPC is used

Architecture of microprogrammed CU



ure 4.31 General-purpose Micro
es, Computer

Architecture contd..

Control memory buffer register(CMBR) : buffer for microinstructions. Each microinstruction has 3 fields

condition select	branch address field	control function field

Condition select selects the external condition to be tested.

If selected condition is true, o/p of MUX will be 1 and MPC loaded with address specified in **branch address field**. Else MPC will point to next microinstruction

Control function field may hold the control info in an encoded form.

MPC: holds address of next microinstruction to be executed

Initially loaded with from an external source(starting addr)

Loaded with branch address field during branch instr

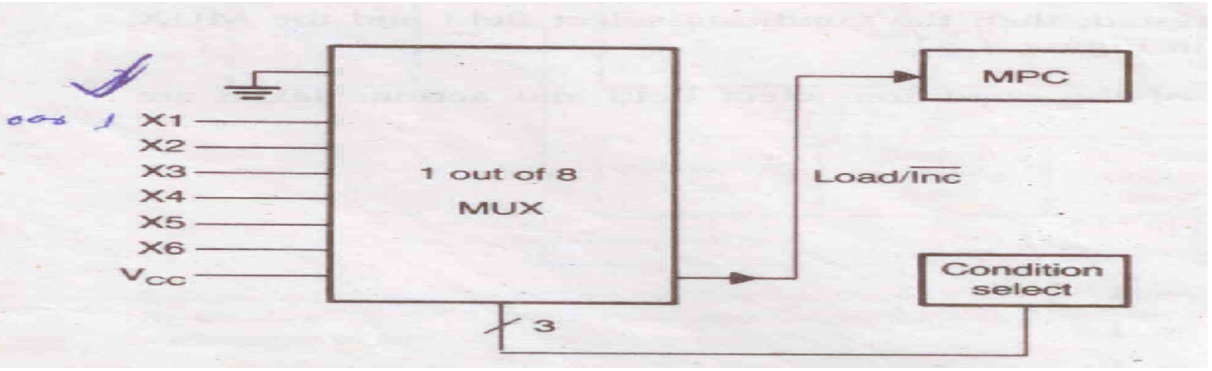
Architecture contd..

MUX:

Selects one of the external conditions according to the contents of condition select field.

Ex: Condition select field and MUX for 6 external conditions

000 → no branching, 001 →branch if X1 is true and so on, 111→unconditional branching



Microprogramming is an activity of writing microprograms for a microprogrammable computer

Designing a microprogrammed control unit for 4*4 booths multiplier

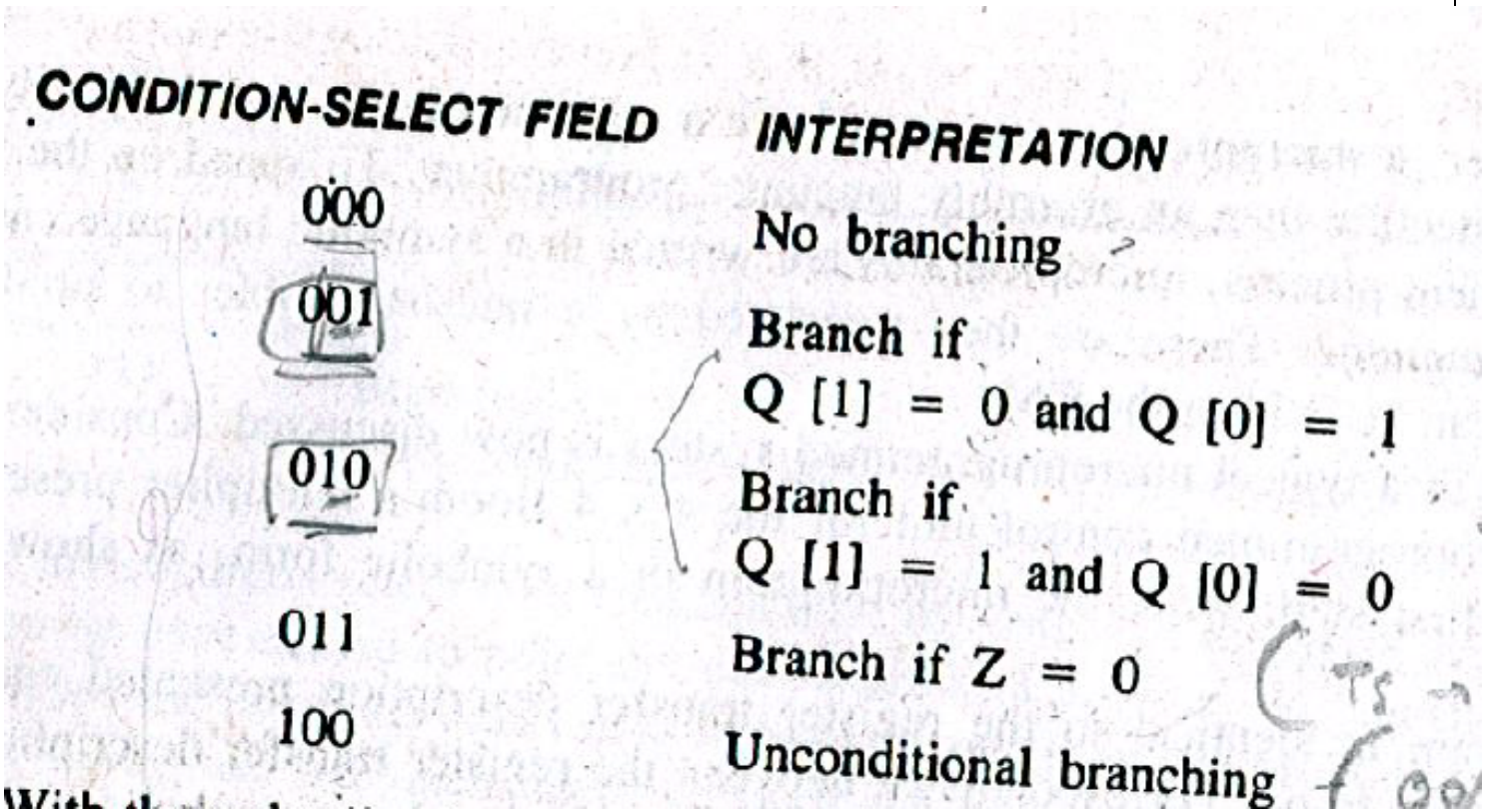
Step 1: Symbolic microprogram for booths multiplier

Control Memory Address		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 go to ADD
3		If Q [1:0] = 10 Then go to SUB
4		Go to RSHIFT;
5	ADD	A ← A + M;
6		Go to RSHIFT;
7	SUB	A ← A − M;
8	RSHIFT	ASR (A$Q), L ← L − 1;
9		If Z = 0 then go to LOOP
10		Outbus = A;
11		Outbus = Q [4:1];
12	HALT	Go to HALT

Contd…

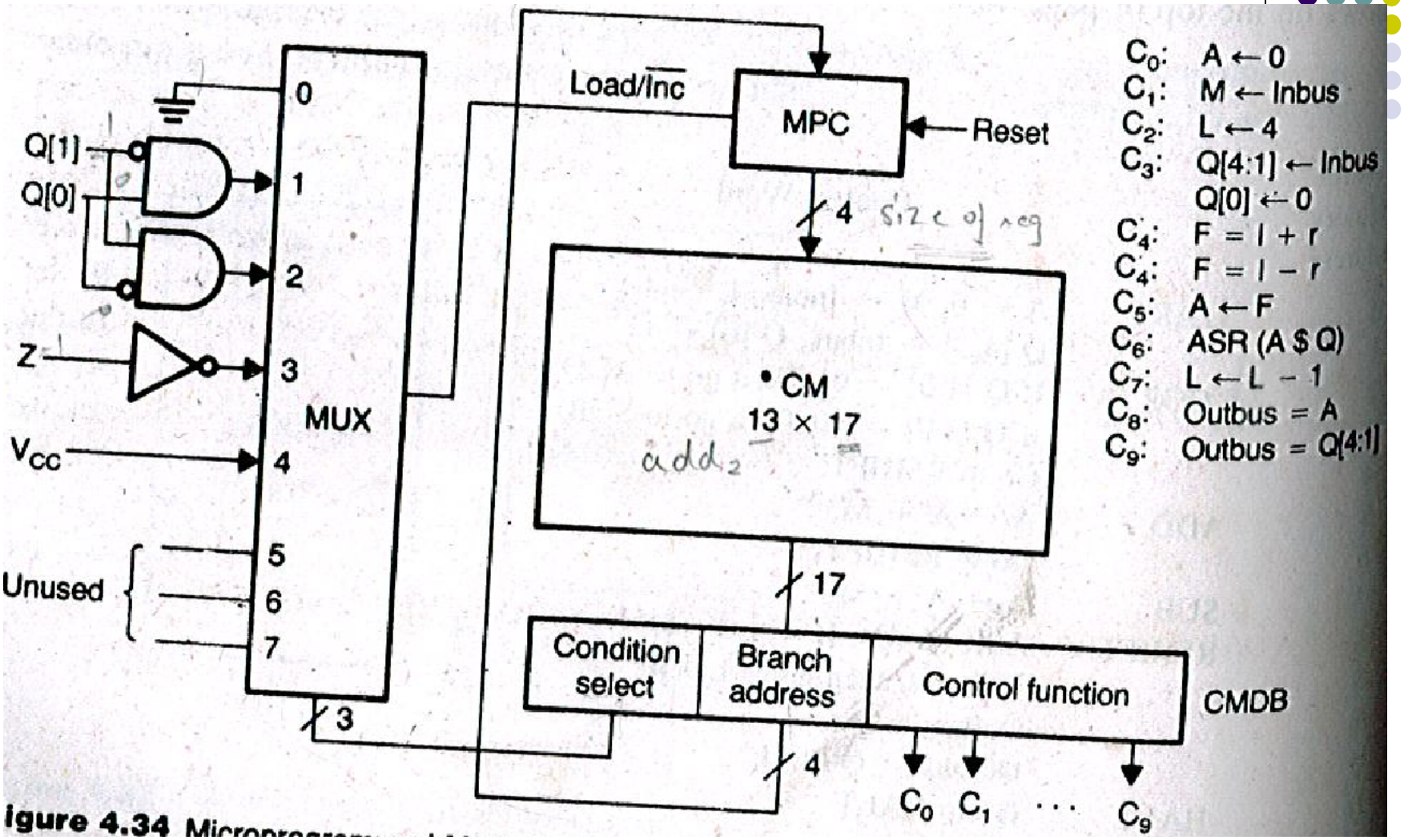
- Each line of symbolic microprogram is stored as a word in control memory
- Ex: word stored in address 4 implements the unconditional branch
- CM is capable of holding 13 words , requiring 4 bit branch address field
- Three actual conditions are checked: Q[1]Q[0]= 0 1, 1 0 and Z=0.these are given as inputs to MUX
- 2 more conditions: no branch and unconditional branching
- Hence 5 inputs to MUX, so 3 bits in condition select

Condition select interpretation for booth



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

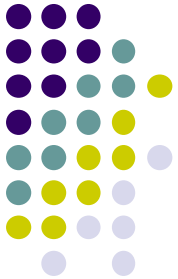
Microprogrammed CU for booth



Size of each Control word:

3+4+10(condn select + branch addr + no of control functions)

Microprogram stored in CM



- First line of symbolic microprogram has no branching, hence condition select is 000 and branch address is 0000. 3 control functions(c0, c1, c2) are active.

  control word:

000	0000	1110000000

- 10th line doesn't have any control lines. Branch if Z=0, hence condn select is 011 and branch address is LOOP which is at addr 2. so branch addr is 0010.

  control word:

011	0010	0000000000

Complete Microprogram stored in CM

ROM Address													Control Word									
In decimal	In binary				Cond. Sel			Branch Addr.					Control Function									Comments
					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)