Control Unit

Mohammed Rafiquzzaman and Rajan Chandra, Modern Computer Architecture, Galgotia Publications Pvt. Ltd., 2015.

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.

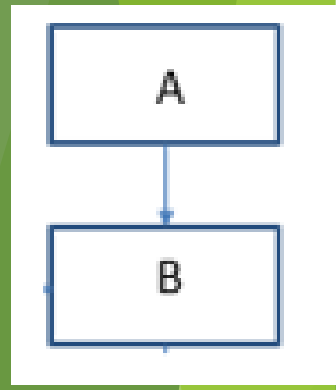
Basic Concepts

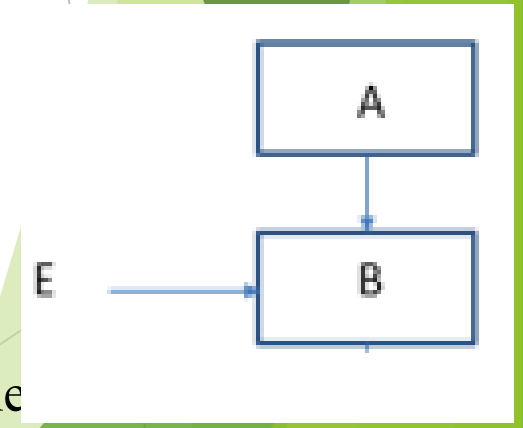
- Basis for CU design are register transfer operations
  1. 8-bit info moved from Register A to Register B.
     - Such operation is described as B←A
     - Declaring registers: Declare registers A[8], B[8], PC[16] ;
  2. Register can be defined as a portion of some other register.
     - Assigning higher 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:
     i. Enable input (E) or control input controls the data flow from A to B
     ii. Data input

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

  Such conditional transfer is expressed as

  E: B←A



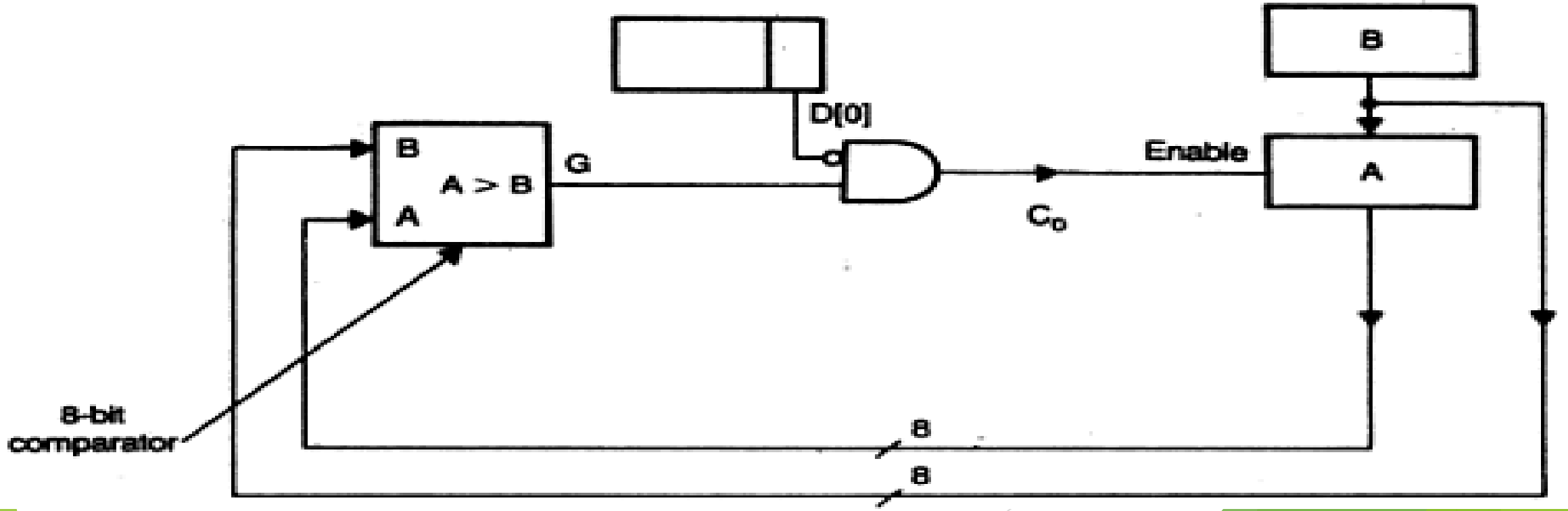


Basic Concepts (Contd.)

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←B

else

A←D

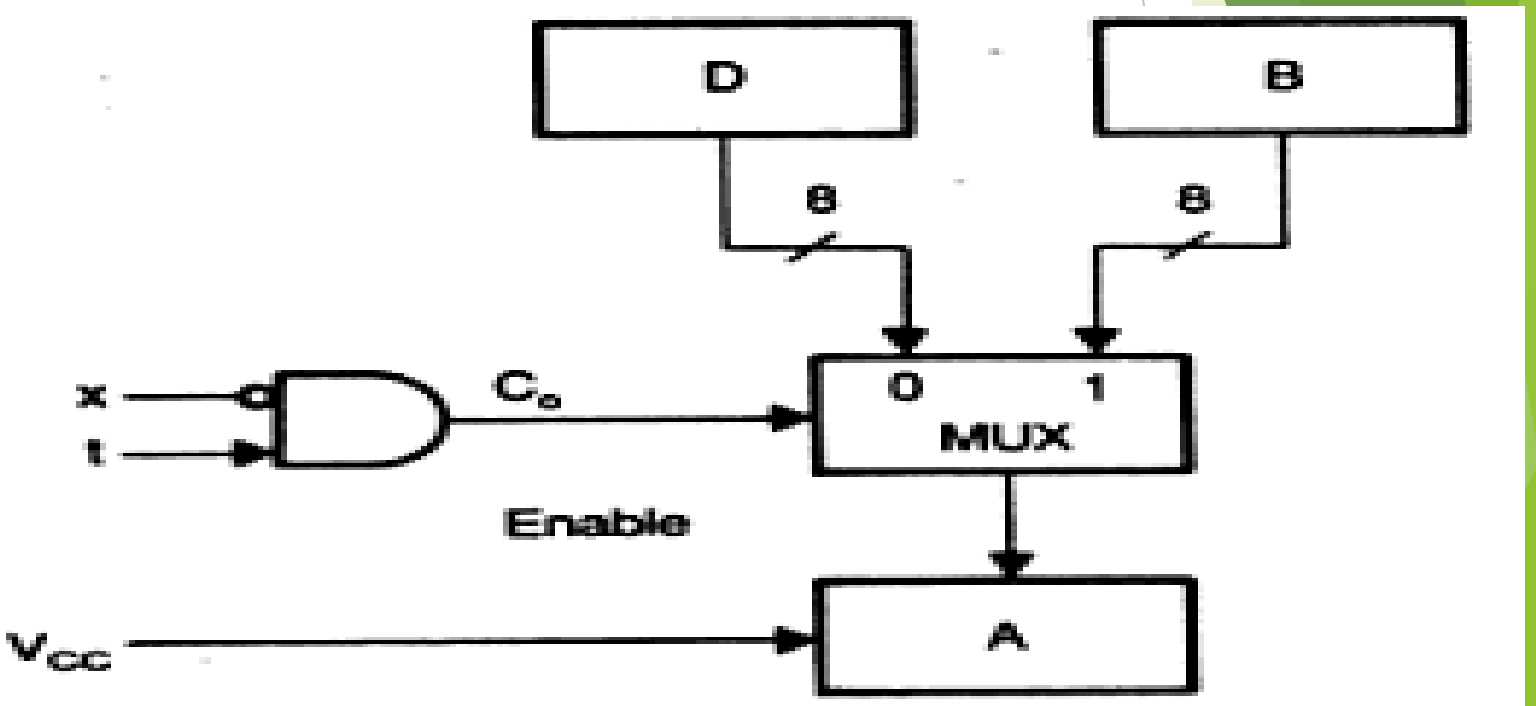
Such transfer is expressed as

C_0 : A←B ;

C_0 ' : A←D ;

Where C_0 = x't and

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



Basic Concepts (Contd.)

- The other register transfer operations are
- D←A’; Transfer the complement of A to D.
- A← A+1; Increment the content of A by 1.
- A← A-1; Decrement the content of A by 1.
- D← A V B; A OR B, store result in D
- D← A Λ 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) ;

HARDWIRED APPROACH

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.

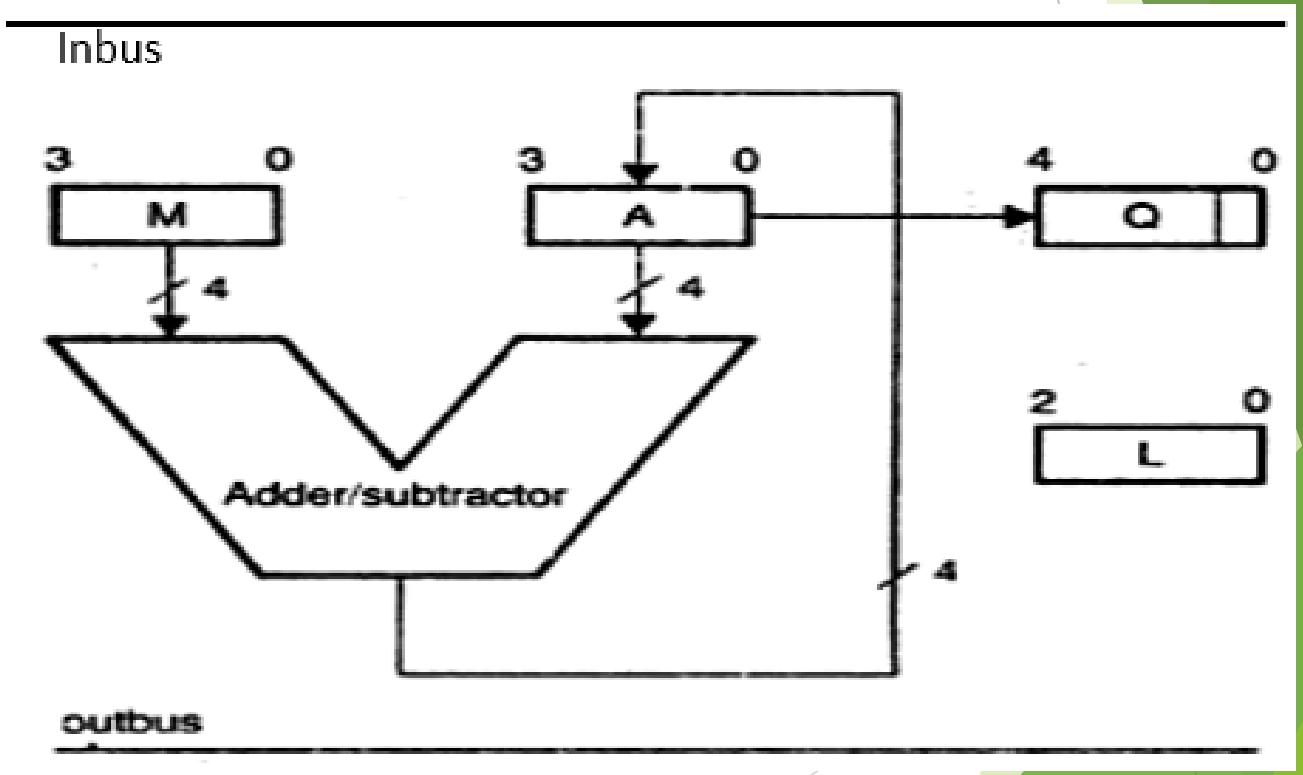
q_1 q_0

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

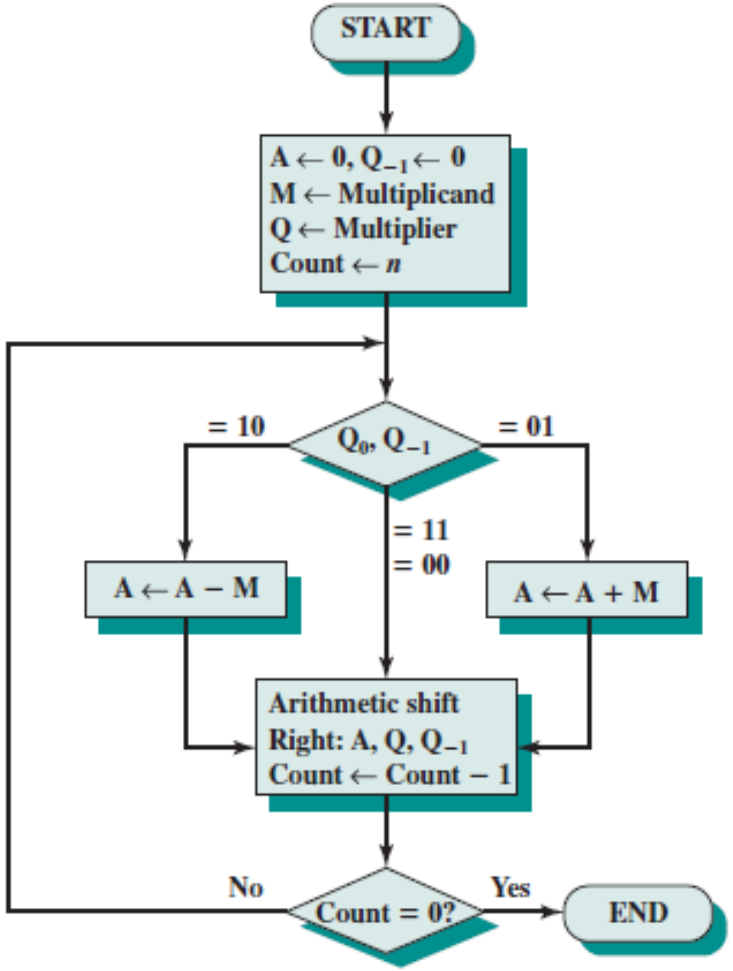


Figure 10.12 Booth's Algorithm for Twos Complement Multiplication

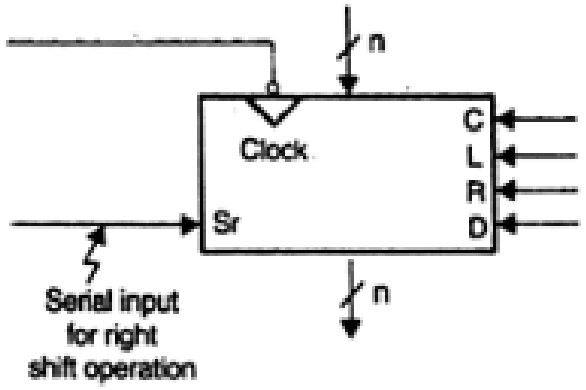
10 steps for hardwired control

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

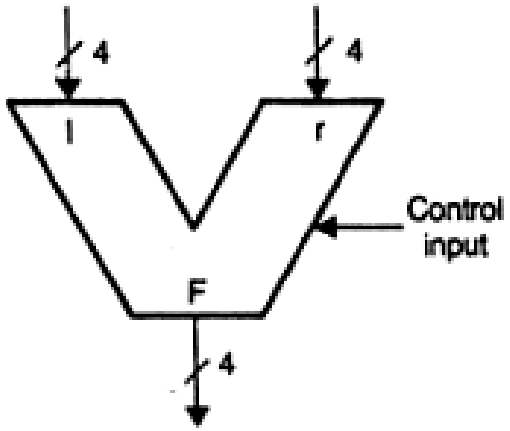
10 steps for hardwired control

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



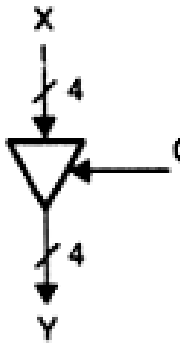
C	L	R	D	Clock	Action
1	0	0	0	↓	Clear
0	1	0	0	↓	Load external data
0	0	1	0	↓	Right shift
0	0	0	1	↓	Decrement by one
0	0	0	0	↓	No change

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.

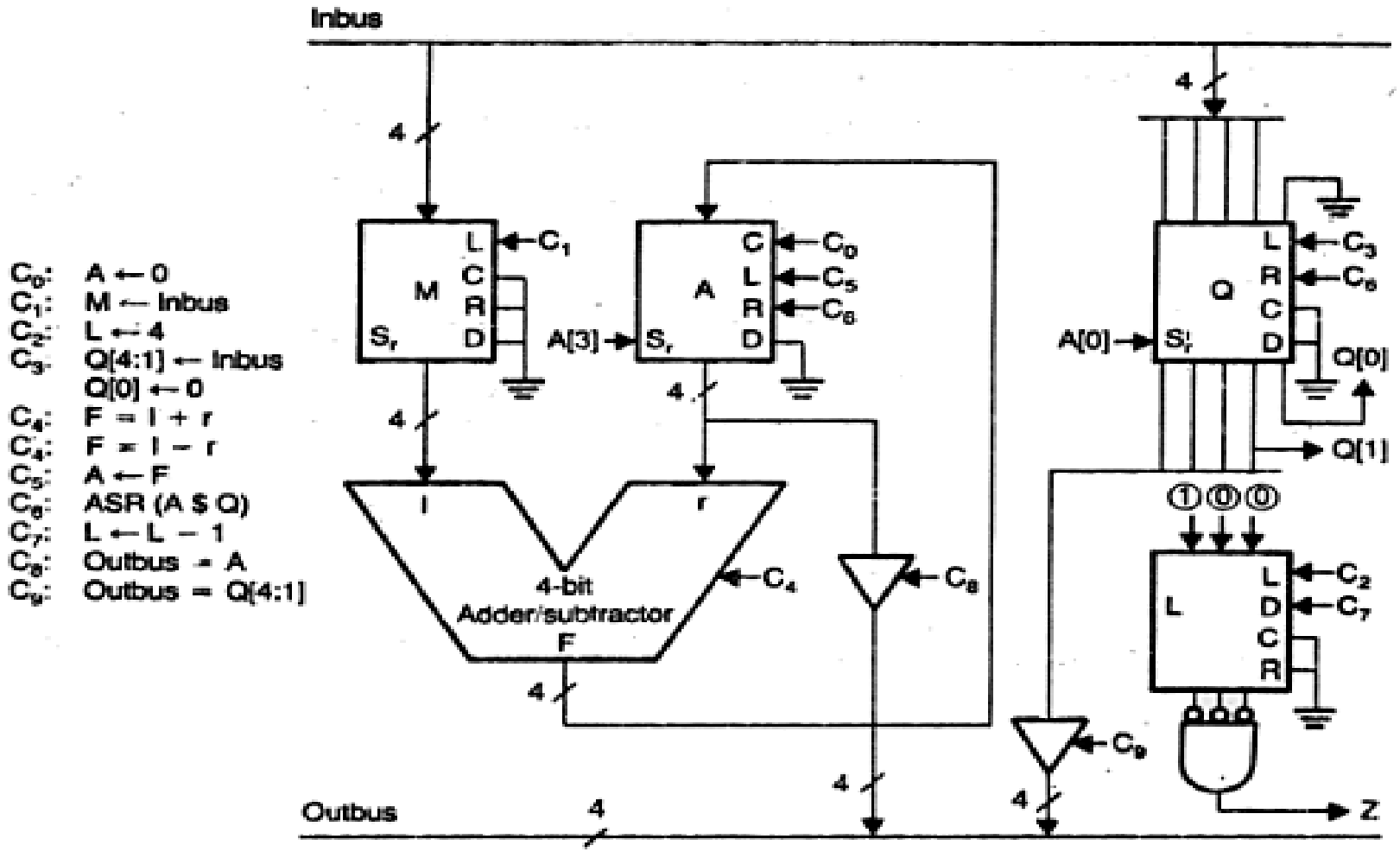
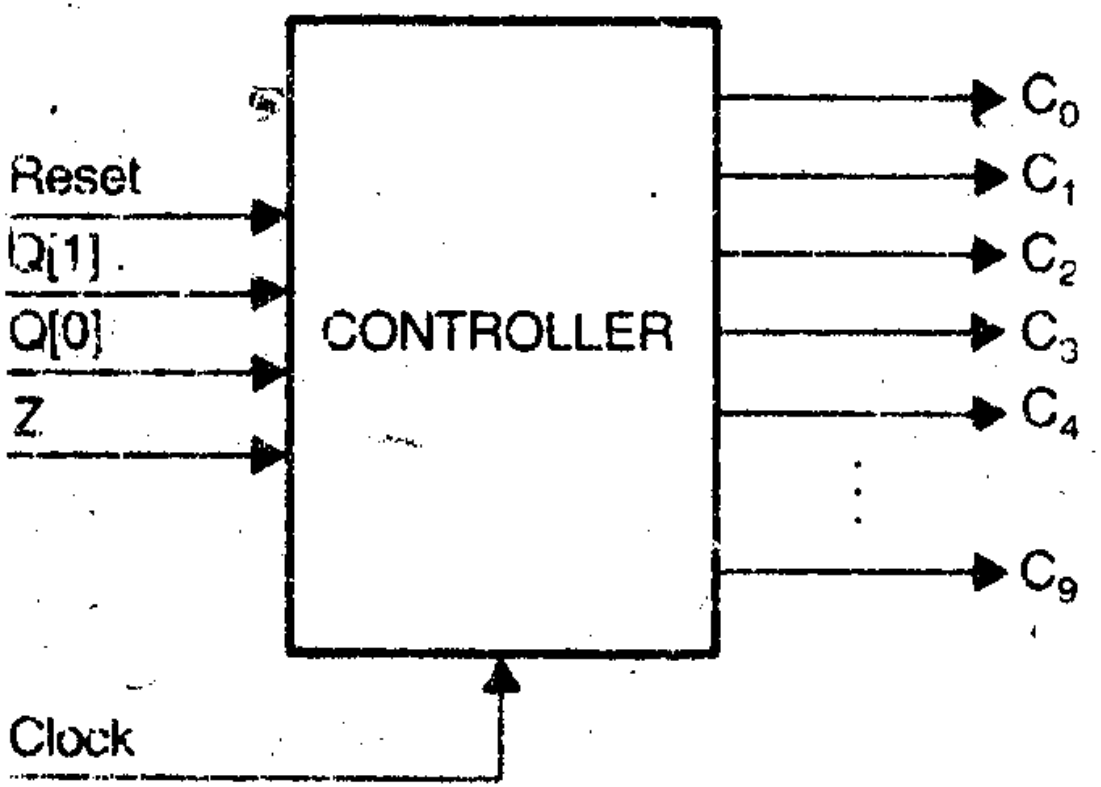


Figure 4.18 Processing Section of the Booth's Multiplier

10 steps for hardwired control

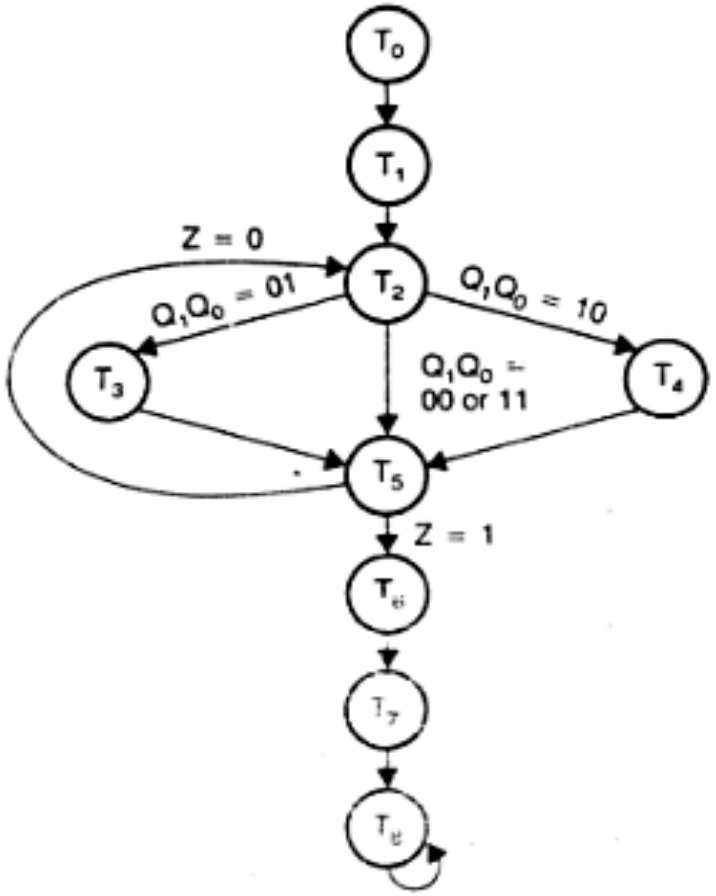
Step 7: Block diagram of controller:

- 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 ← 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 (ASQ), 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

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

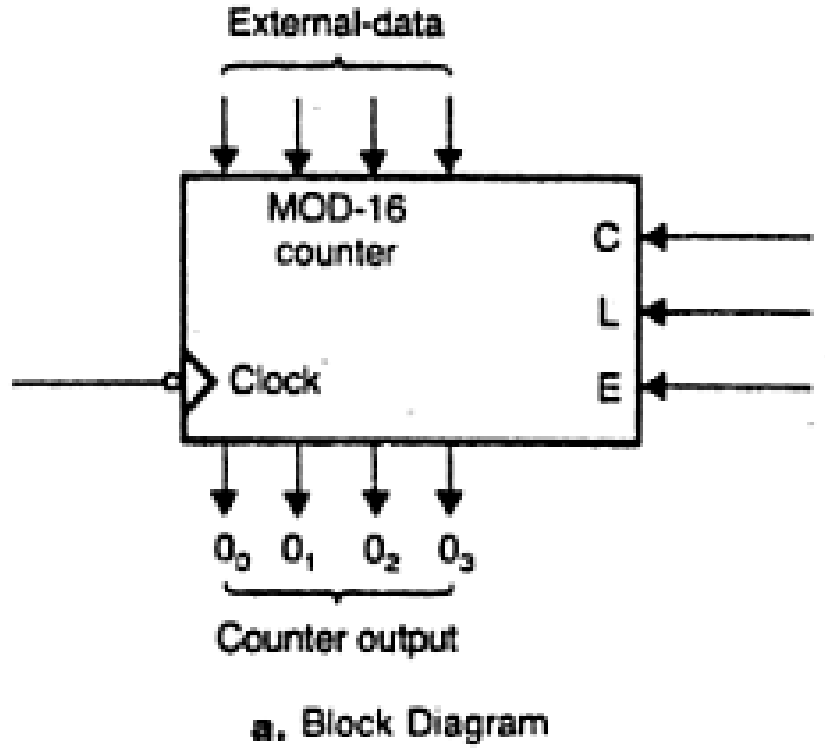
10 steps for hardwired control

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

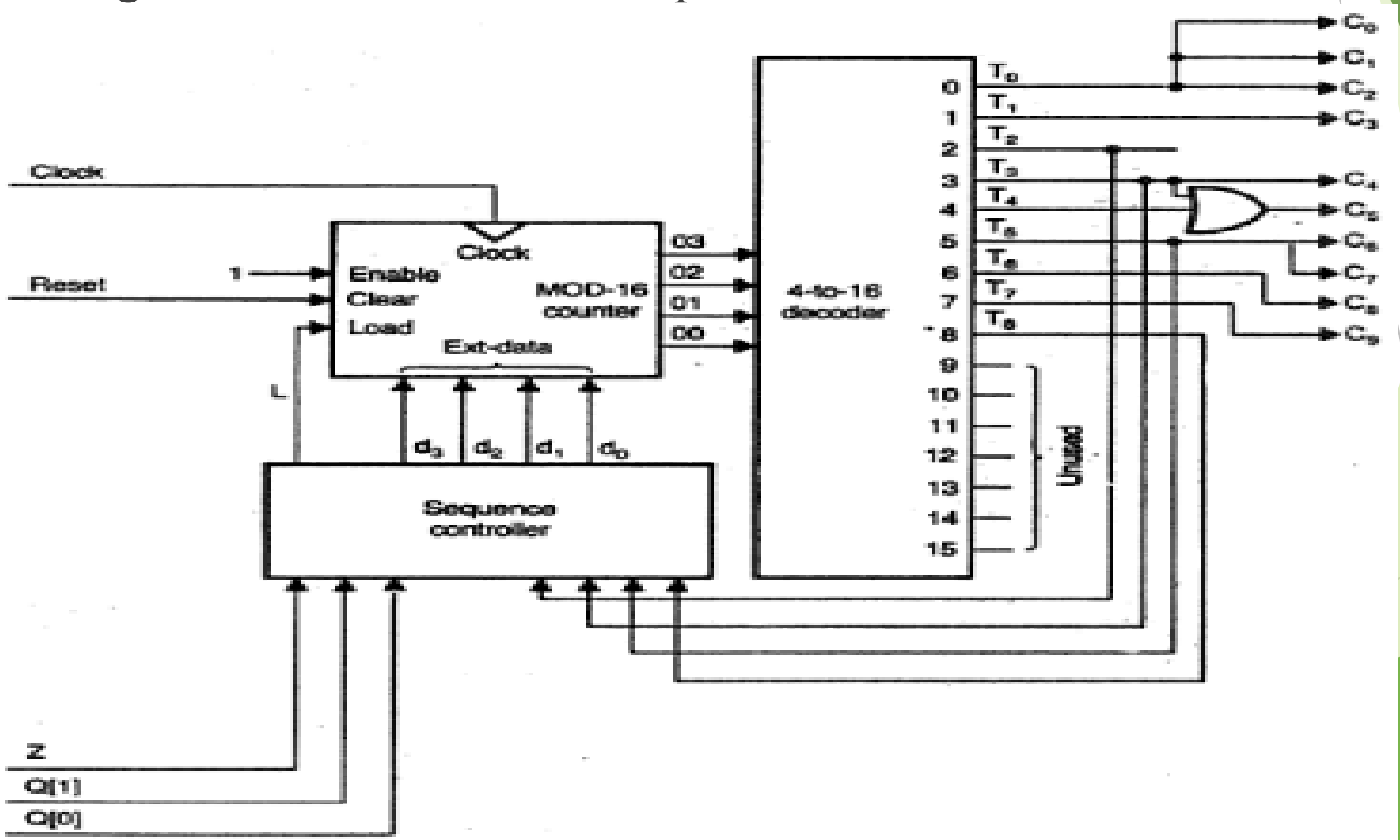


Figure 4.23 Logic Diagram of the Booth's Multiplier Controller

10 steps for hardwired control

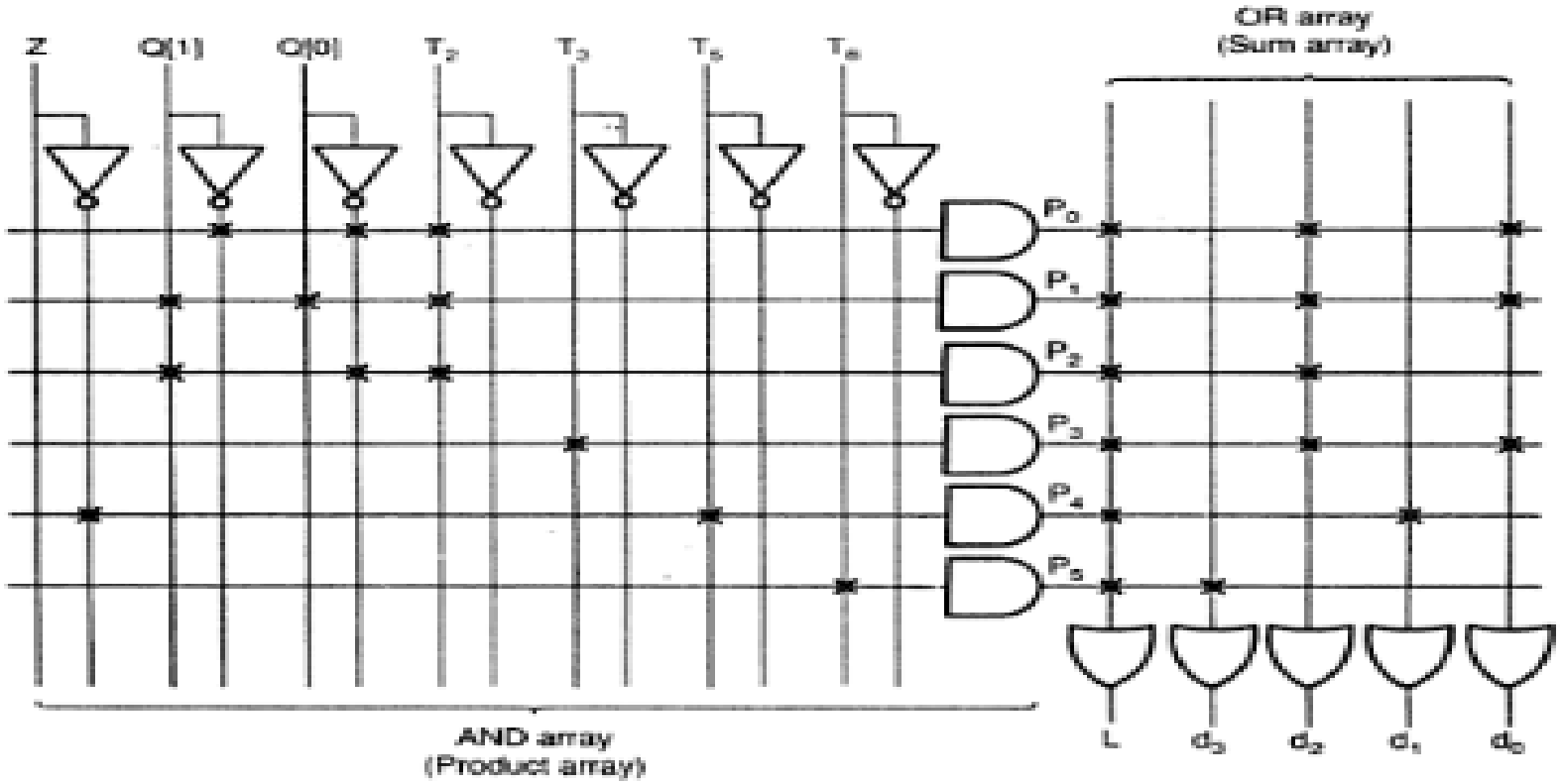
Step 10: Truth Table for SC Design

								External-data			
Z	Q [1]	Q [0]	T_2	T_1	T_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_6	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 unit (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

Additional Problems

A) Design a hardwired CU for the RTD given below. Establish all the control points and design the controller using appropriate counter, decoder and sequence controller. Also, provide the implementation of sequence controller using PLA.

B) Give the binary listing of the microprogram for the Register Transfer Description given below. Show all the necessary steps to arrive at the binary listing.

Declare registers A[8], B[8]

Declare inbus[8], outbus[8];

START: A ⟵ inbus, B ⟵ 0;

BACK: If A[0]=0 then go to ADD;

go to NEXT;

ADD: B⟵ B+A;

NEXT: A ⟵ A-1;

If A>0 then go to BACK;

Outbus=B;

HALT: go to HALT;