AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH (AIUB)
Where leaders are created

Lecture # 5 (Final)
Control Logic Design
Micro-program, flow-chart, state-diagram

Department of EEE

Introduction

- The process of logic design is a complex undertaking.
- The data processor part may be general purpose processor unit
- The data processor may consist of individual registers and associated digital functions

- The **control logic initiates all micro-operations in the data processor**.
- The control logic is **a sequential circuit** whose internal states dictate the control functions for the system by generating the signals for **sequencing the micro-operations**.

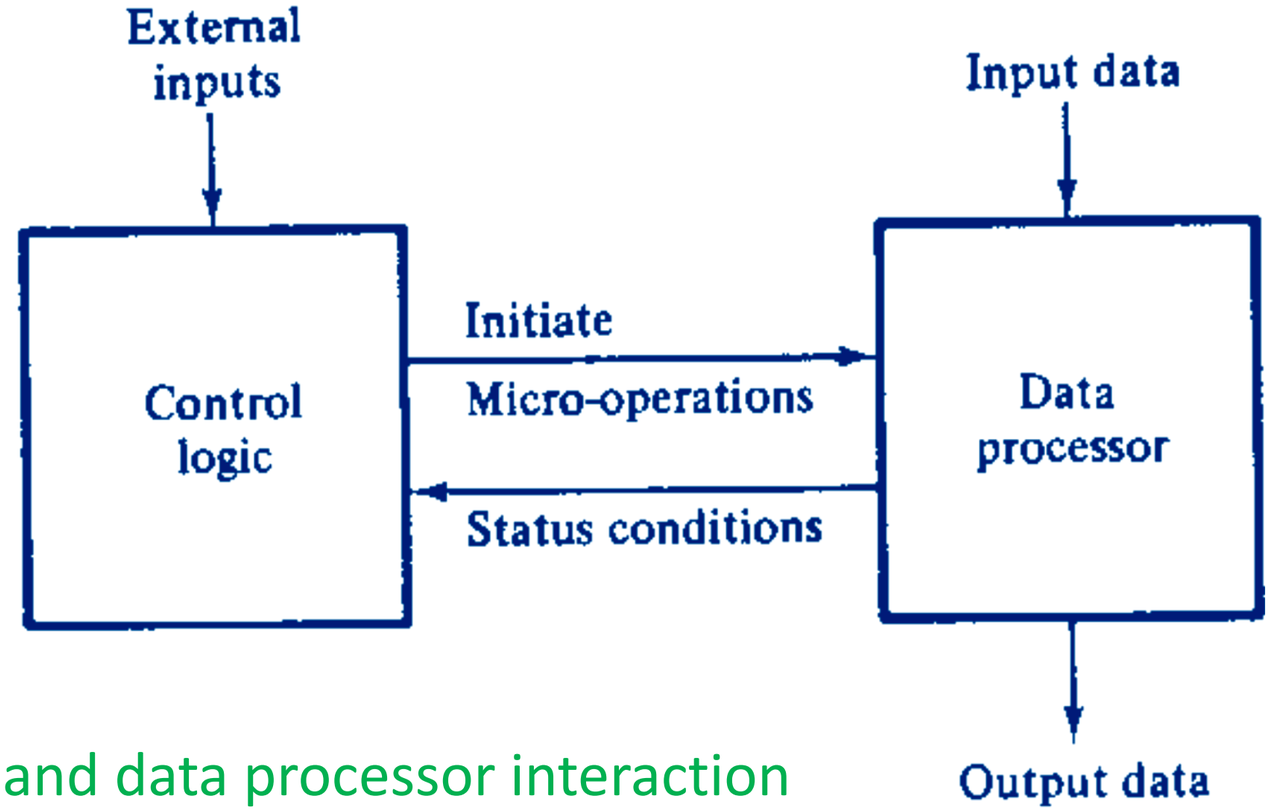


Fig. 10.1 Control logic and data processor interaction

Control Organization

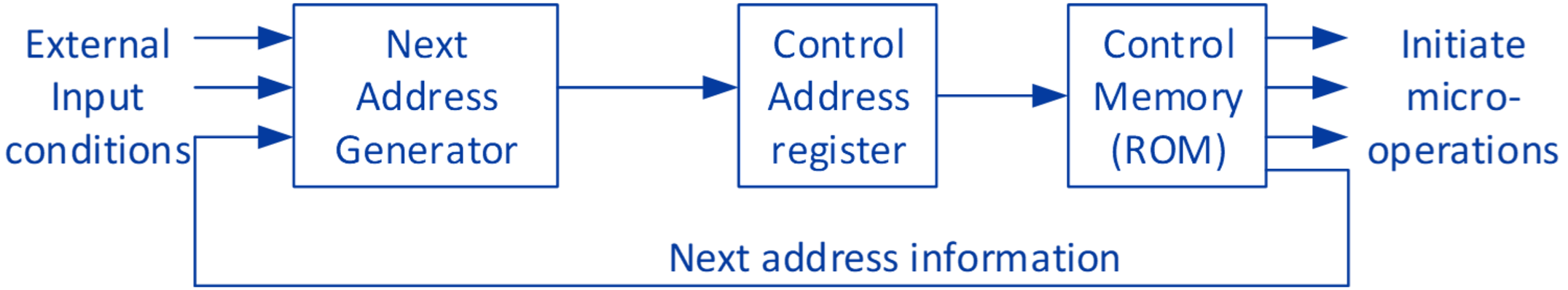
- **Methods of Control Organization**
  - One flip-flop per state method
  - Sequence register and decoder method
  - PLA control
  - Microprogram control

Scale	Components Count	Year
SSI (Small Scale Integration)	< 100	1963
MSI (Medium Scale Integration)	100-1000	1970
LSI (Large Scale Integration)	1000-10000	1975
VLSI (Very Large Scale Integration)	$10000 - 10^6$	1980
ULSI (Ultra Large Scale Integration)	$> 10^6$	1990
GSI (Giga Scale Integration)	$> 10^{10}$	2010

- The first two methods must use SSI and MSI circuits for the implementations.
- PLA or microprogram which uses an LSI device.

Microprogram Control

- The purpose of the control unit is to initiate a series of sequential steps of microoperations.
- A control unit whose control variables are stored in a memory is called a **Microprogrammed Control Unit (MCU)**.
- Each control word of memory is called a **microinstruction.**
- A sequence of microinstructions is called a **microprogram.**



Hard-wired Control - Example

- The design is carried out in **five consecutive steps**-
  1. The problem is stated
  2. An initial equipment configuration is assumed
  3. An algorithm is formulated
  4. The data processor part is specified
  5. The control logic is designed

1. Statement of the Problem

- Addition and subtraction of binary fixed-point numbers when negative numbers are in sign 2's complement form.
- The addition of two numbers stored in registers of finite length (8 bits) may result in a sum that exceeds the storage capacity of the register by one bit.
- The extra bit is said to cause an overflow.

+ 6	0 000110		− 6	1 111010	
		+			+
+ 9	0 001001		+ 9	0 001001	
+ 15	0 001111		+ 3	0 000011	
+ 6	0 000110		− 9	1 110111	
		+			+
− 9	1 110111		− 9	1 110111	
− 3	1 111101		− 18	1 101110	

2. Equipment Configuration

- The two signed binary numbers to be added or subtracted contains *n* bits.
- The magnitudes of numbers contain $k = n - 1$ bits and are stored in registers *A* and *B*.
- The sign bits are stored in flip-flops named A_s and B_s.

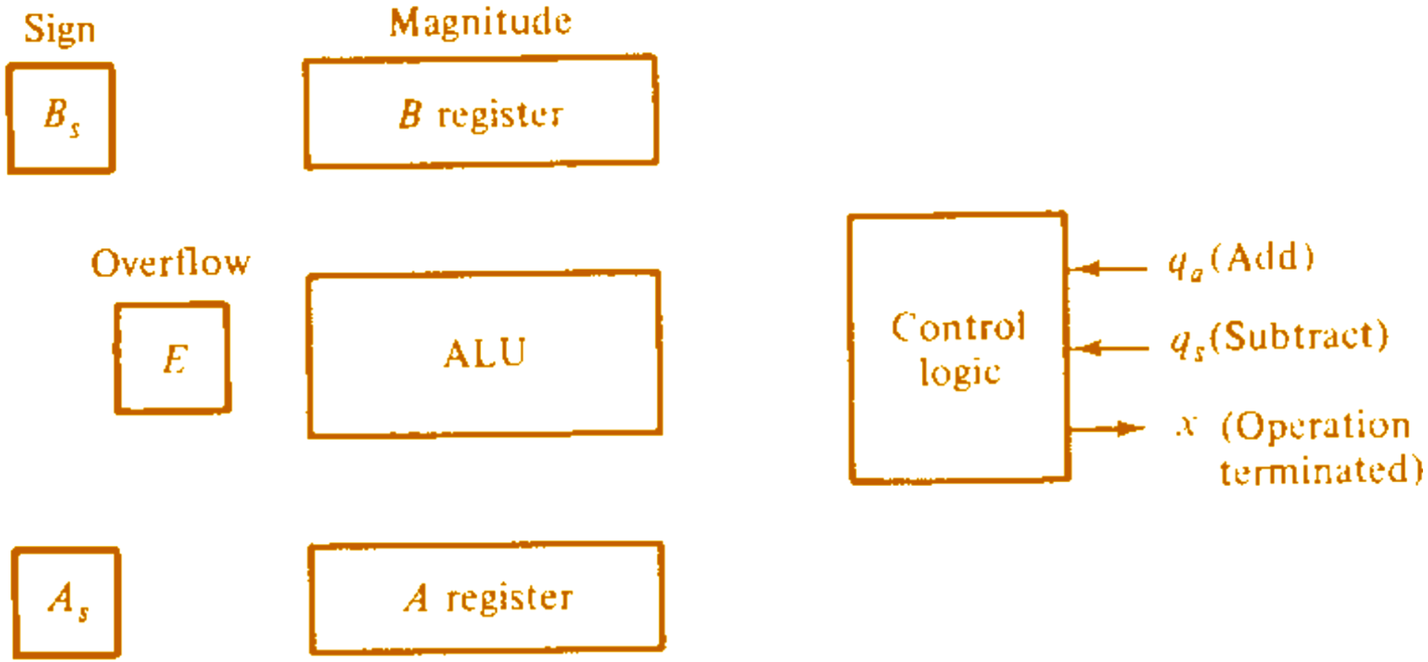


Fig. 10-6 Register and associated equipment configuration for the adder-subtractor

2. Equipment Configuration

- The ALU performs the arithmetic operations, and the 1-bit register *E* serves as the overflow flip-flop, where the output carry is transferred.
- It is assumed that the 2 numbers and their signs have been transferred to their respective registers and that the result of the operation is to be available in registers *A* and A_s.
- Two input signals in the control logic specify the add (q_a) and subtract (q_s) operations.
- One output variable *x* indicates the end of the operation.

2. Equipment Configuration

- The control logic communicates with the outside or external environment through the input and output variables.
- Control logic recognizes input signals, q_a or q_s and provides the required operation.
- Upon completion of the operation, the control logic informs the external environment with the output, *x* that the sum or difference is in registers *A* and A_s and the overflow bit is in *E*.

3. Derivation of the Algorithm

- When the numbers are added or subtracted algebraically, there are eight different conditions to be considered and may be expressed in compact form as follows:

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

- In the arithmetic operation specified in subtraction, we change the sign of B and add. So, the relationships:

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

- This reduce the number of possible conditions to 4, namely:

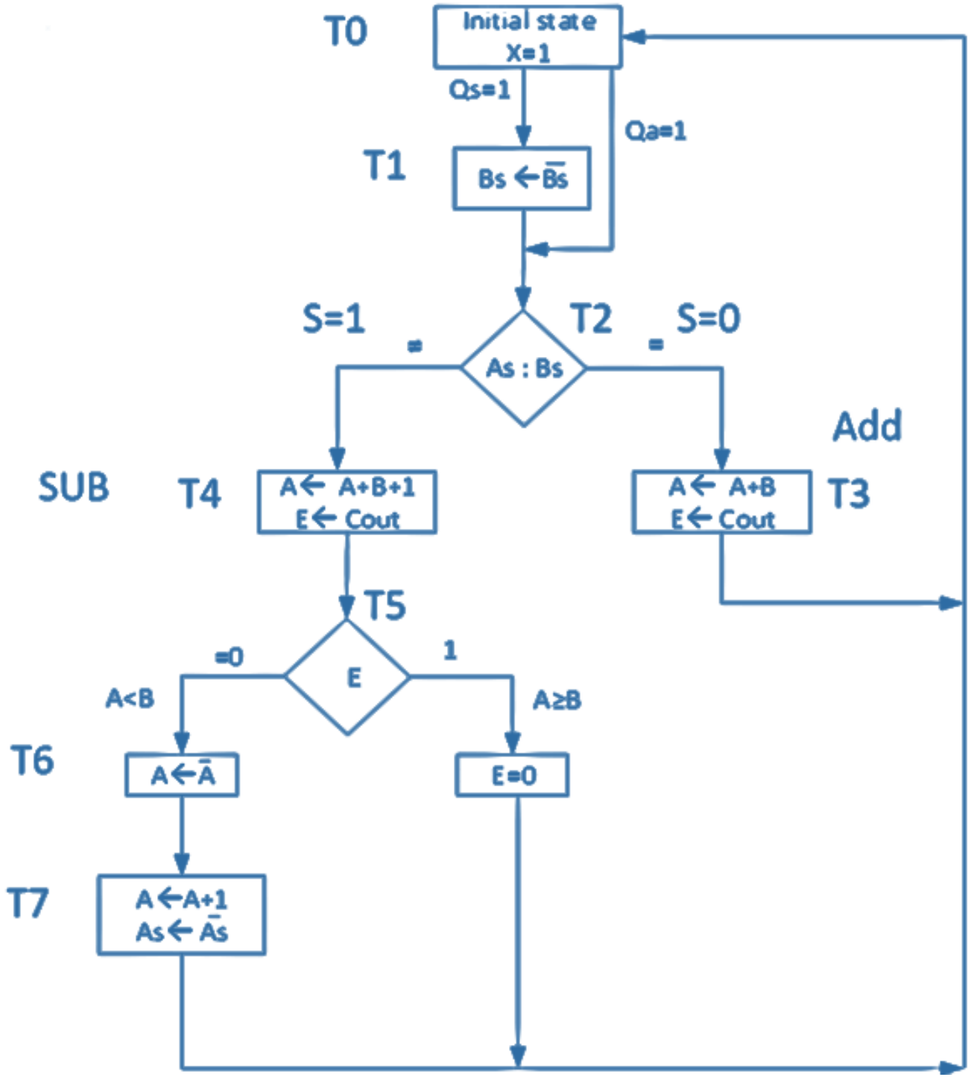
$$(\pm A) + (\pm B)$$

3. Derivation of the Algorithm contd….

- When the signs of *A* and *B* are the same, we add the two magnitudes and the sign of the result in the same as the common sign.
- When the signs of *A* and *B* are **not** the same, we subtract the smaller number from the larger and the sign of the result is the sign of the larger number.

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

Flowchart and state diagram for sign magnitude addition and subtraction operation



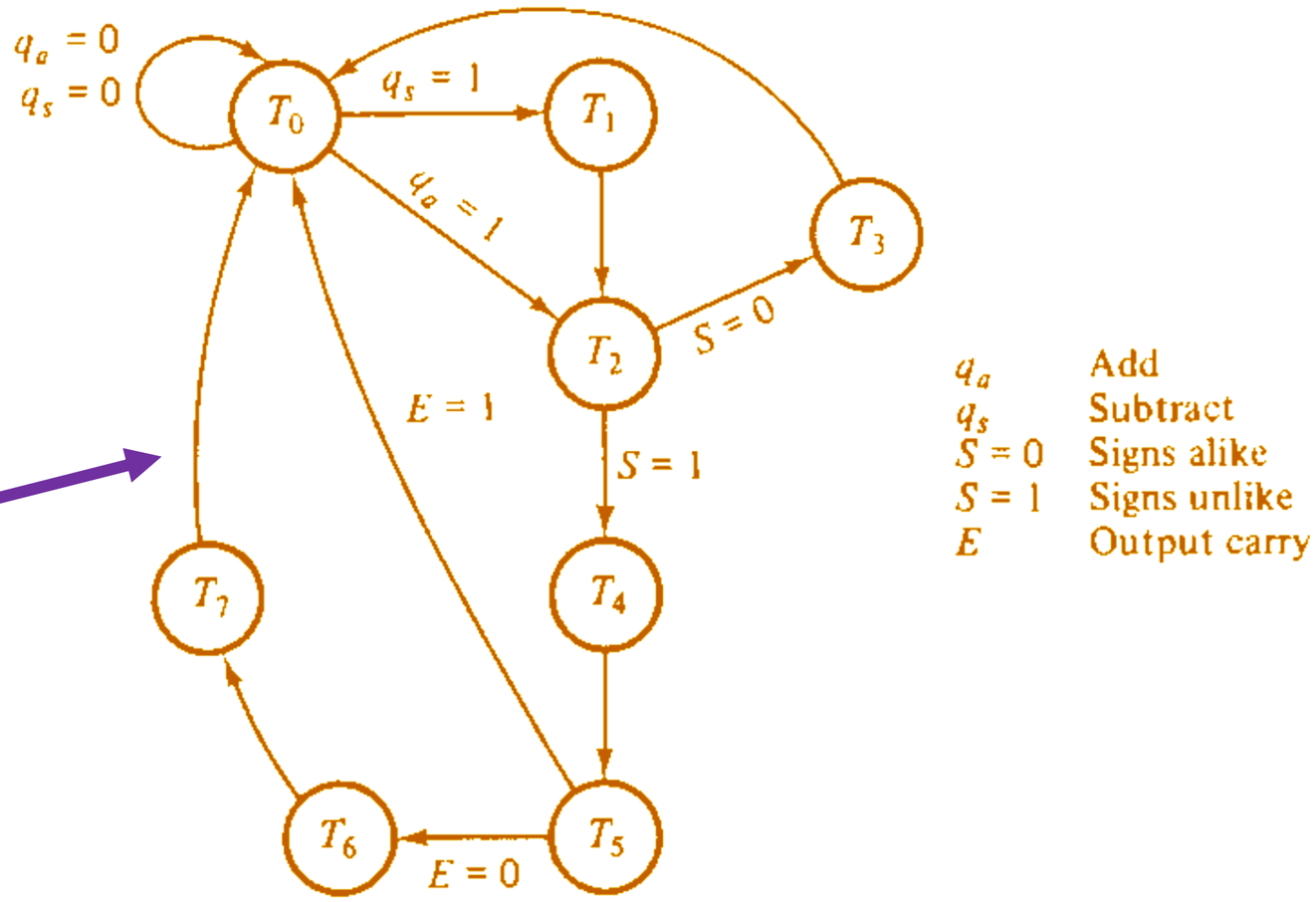


Fig. 10-7 Flow chart for sign-magnitude addition and subtraction operation with the equipment of Fig. 10-6

Explanation of the Flowchart and State Diagram

1. An operation is initiated by either q_s or q_a inputs
2. Input q_s initiates a subtraction operation, so sign of B is complemented
3. Input q_a initiates an addition operation, so sign of B is left unchanged
4. Then two signs are compared, A_s:B_s symbolizes this decision, if two signs are equal then the path marked with = sign is chosen; otherwise, the path marked with ≠ sign is chosen

Explanation of the Flowchart and State Diagram

5a. If signs are equal then the contents of A is added to the contents of B, and the sum is transferred to A. The value of the end carry is an overflow, as such the E flip-flop is made equal to the output carry, C_{out}.

5b. If signs are not the same then the contents of B is subtracted from the contents of A, this is done by adding A to the 2's complement of B and the sum is transferred to A. The value of the end carry is zero, that is, there is no overflow, as such the E flip-flop is cleared to zero. However, a 1 in E indicates that $A \geq B$, and the number in A is the correct result. The sign of the result again is equal to the original value of A_s.

Explanation of the Flowchart and State Diagram

5c. A 0 in E indicates that $A < B$, then the 2's complement of A is taken and the sign in A_s is complemented. The 2's complement of A can be done with a micro-operation $A \leftarrow A' + 1$, where A' is the 1's complement of A . However, the 2's complement operation is not available, so it is performed by taking the 1's complement and then the increment operation, which are available in the designed ALU.

6. Finally, the circuit goes to its initial state and output x is equal to 1 (terminate). If the sign of the result is the same as the original sign of A_s, the sign bit is left unchanged.

4. Data Processor Register

- The operations between *A* and *B* can be done with the ALU.
- **The operations with A_s, B_s and *E* must be initiated with the separate control variables**.

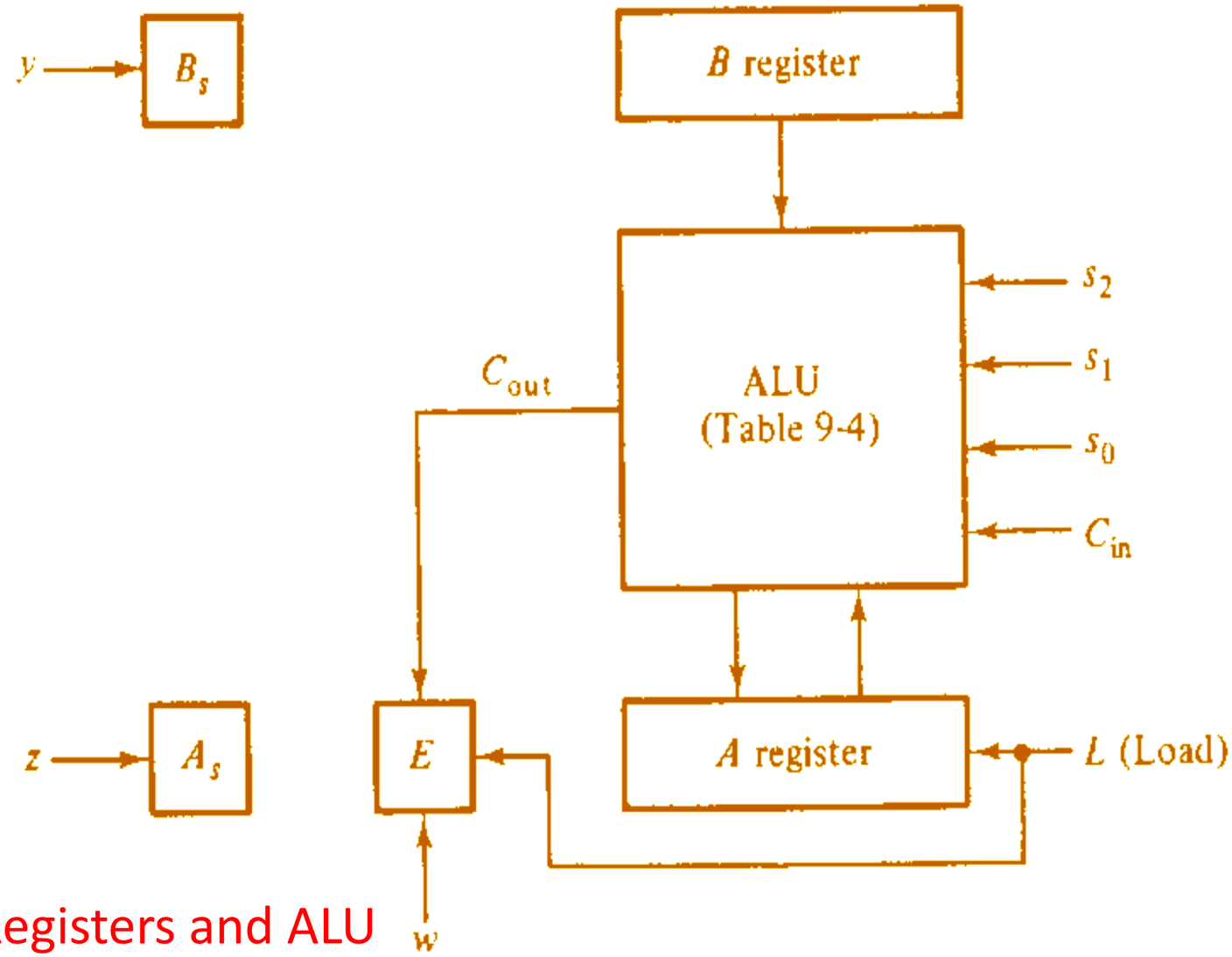
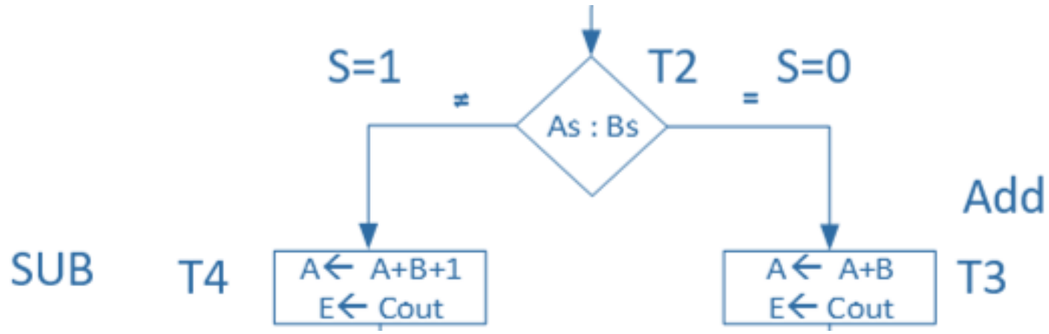


Fig. 10-8 (a) Data Processor Registers and ALU

5. Control Block Diagram

- The control logic block receives five inputs: two from the external environment and three from the data-processor.
- To simplify the design, we define new variable *S*:

$$S = A_s \text{ XOR } B_s$$



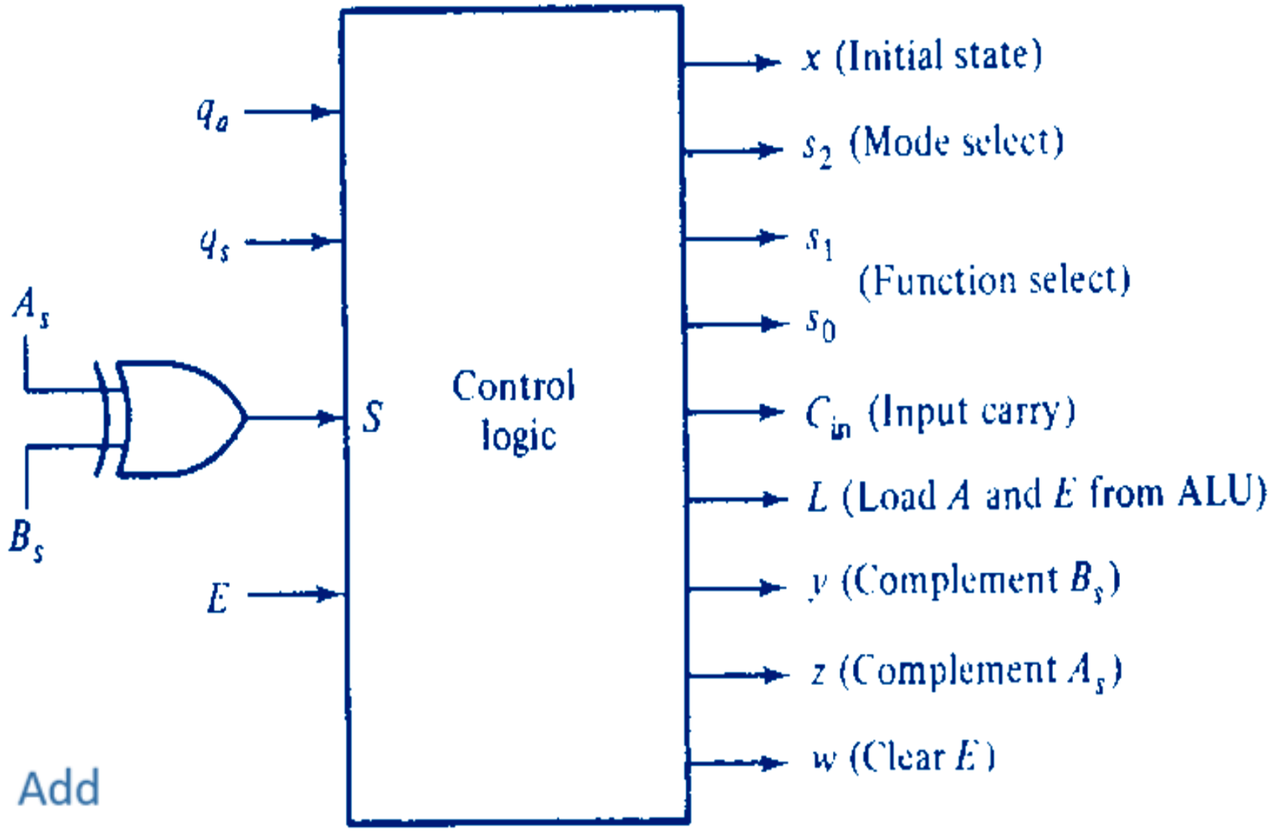


Fig. 10-8 (b) Control Logic Block Diagram

Control State Diagram

T_0: Initial state $x = 1$

$T_1: B_s \leftarrow \overline{B}_s$

T_2: nothing

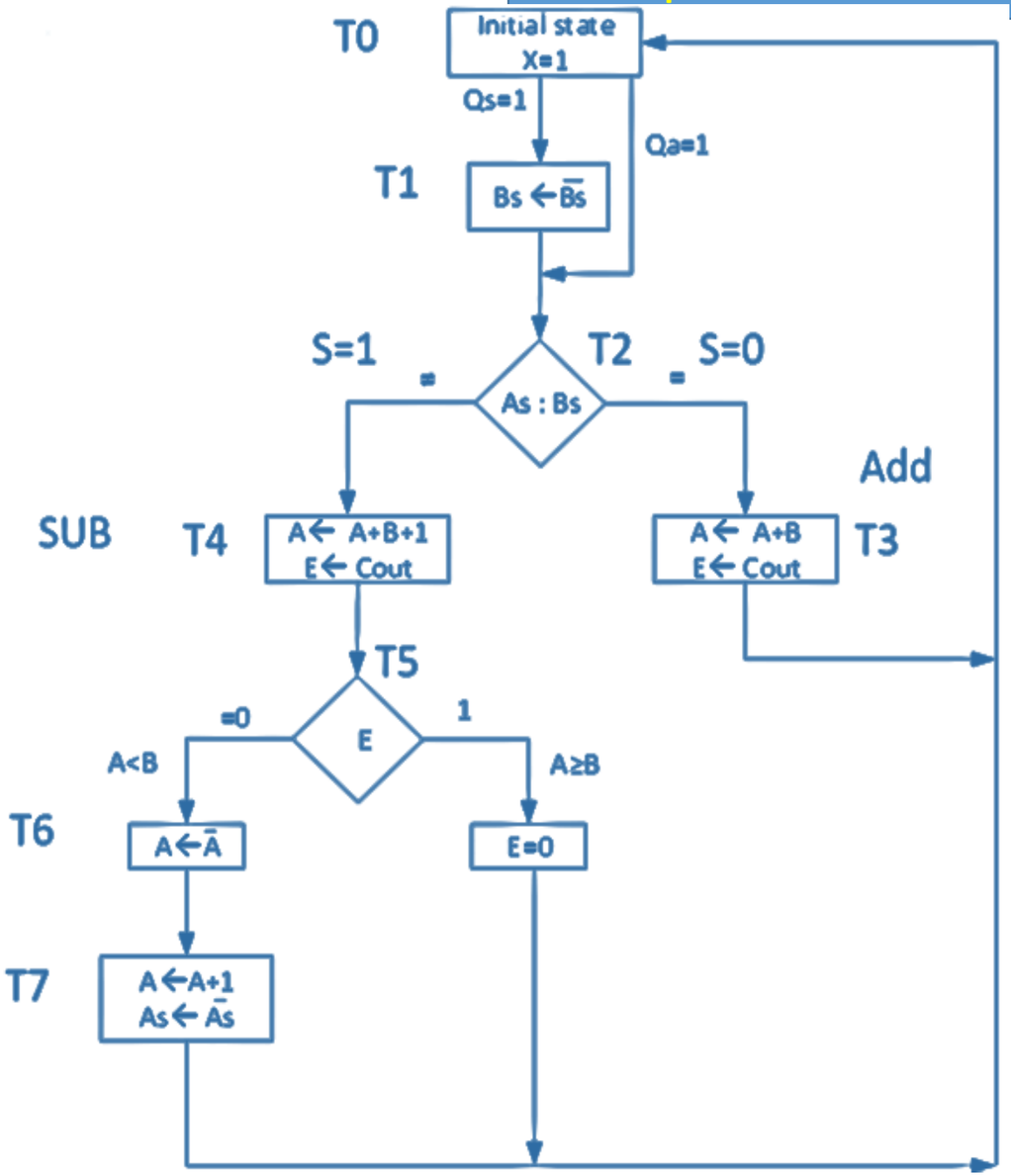
$T_3: A \leftarrow A + B,\ E \leftarrow C_{out}$

$T_4: A \leftarrow A + \overline{B} + 1,\ E \leftarrow C_{out}$

$T_5: E \leftarrow 0$

$T_6: A \leftarrow \overline{A}$

$T_7: A \leftarrow A + 1,\ A_s \leftarrow \overline{A}_s$



Control State Diagram

T_0: Initial state $x = 1$

$T_1: B_s \leftarrow \overline{B}_s$

T_2: nothing

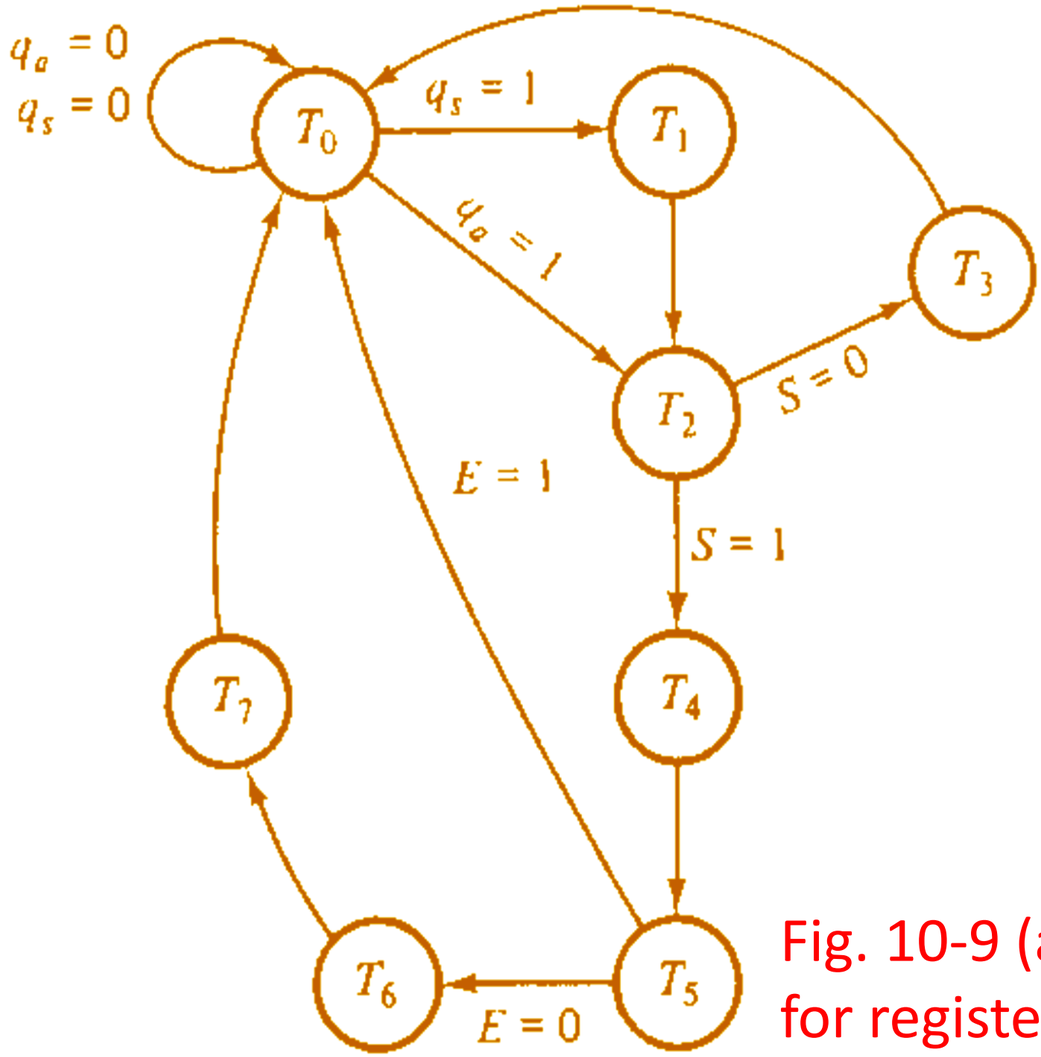
$T_3: A \leftarrow A + B, \quad E \leftarrow C_{out}$

$T_4: A \leftarrow A + \overline{B} + 1, \quad E \leftarrow C_{out}$

$T_5: E \leftarrow 0$

$T_6: A \leftarrow \overline{A}$

$T_7: A \leftarrow A + 1, \quad A_s \leftarrow \overline{A}_s$



q_a	Add
q_s	Subtract
$S = 0$	Signs alike
$S = 1$	Signs unlike
E	Output carry

Fig. 10-9 (a) State diagram for register transfers

Control State Diagram

- We start by assigning an initial state, T_0, to the sequential controller. We then determine the transition to other states T_1, T_2, T_3, and so on.
- For each state, we determine the micro-operations that must be initiated by the control logic circuit.
- This procedure produces the state diagram for the controller circuit together with a list of register-transfer operations, which are to be initiated while the control logic circuit is in each and every state.

Sequence of Register Transfers

Fig. 10-9 (b) Sequence of register transfers

	Control outputs							
	x	s_2	s_1	s_0	C_{in}	L	y	z	w
T_0: Initial state $x = 1$	1	0	0	0	0	0	0	0	0
T_1: $B_s \leftarrow \overline{B}_s$	0	0	0	0	0	0	1	0	0
T_2: nothing	0	0	0	0	0	0	0	0	0
T_3: $A \leftarrow A + B$, $E \leftarrow C_{out}$	0	0	0	1	0	1	0	0	0
T_4: $A \leftarrow A + \overline{B} + 1$, $E \leftarrow C_{out}$	0	0	1	0	1	1	0	0	0
T_5: $E \leftarrow 0$	0	0	0	0	0	0	0	0	1
T_6: $A \leftarrow \overline{A}$	0	1	1	1	0	1	0	0	0
T_7: $A \leftarrow A + 1$, $A_s \leftarrow \overline{A}_s$	0	0	0	0	1	1	0	1	0

x (Initial state)

s_2 (Mode select)

s_1
s_0 (Function select)

C_{in} (Input carry)

L (Load A and E from ALU)

y (Complement B_s)

z (Complement A_s)

w (Clear E)

Sequence of Register Transfers

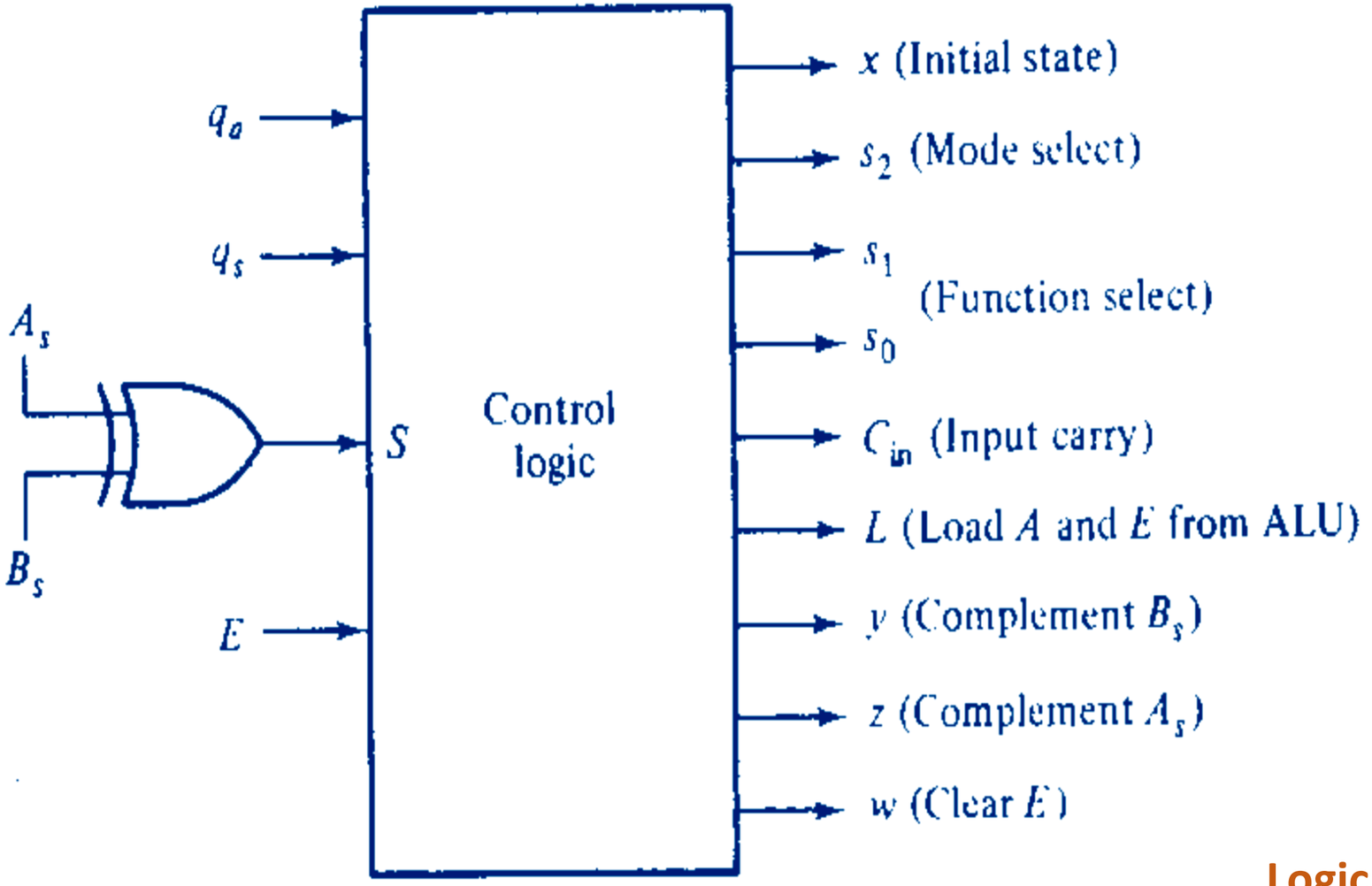


Fig. 10-8 (b) Control Logic Block Diagram

$$x = T_0$$
$$s_2 = T_6$$
$$s_1 = T_4 + T_6$$
$$s_0 = T_3 + T_6$$
$$C_{in} = T_4 + T_7$$
$$L = T_3 + T_4 + T_6 + T_7$$
$$y = T_1$$
$$z = T_7$$
$$w = T_5$$

Logic Equations for Hardwired Control Unit

Boolean Functions for Output Control

Sequence of Register Transfers

TABLE 9-4 Function table for the ALU of Fig. 9-13

Selection				Output	Function
s_2	s_1	s_0	C_{in}		
0	0	0	0	$F = A$	Transfer A
0	0	0	1	$F = A + 1$	Increment A
0	0	1	0	$F = A + B$	Addition
0	0	1	1	$F = A + B + 1$	Add with carry
0	1	0	0	$F = A - B - 1$	Subtract with borrow
0	1	0	1	$F = A - B$	Subtraction
0	1	1	0	$F = A - 1$	Decrement A
0	1	1	1	$F = A$	Transfer A
1	0	0	X	$F = A \vee B$	OR Logical disjunction (∨)
1	0	1	X	$F = A \oplus B$	XOR
1	1	0	X	$F = A \wedge B$	AND Logical conjunction (∧)
1	1	1	X	$F = \overline{A}$	Complement A

Thanks for Attending….