Processor Logic Design

Lesson-11:

Topic	Lesson Learning Outcomes	Teaching-Learning Methodology	Assessment Method
Processor organization	• To understand the bus organization, memory organization, etc. • To know the importance of scratchpad memory and accumulator register. • Compare different processor organization	Class Lecture Question and answer	Test, exams, quiz, etc

Introduction

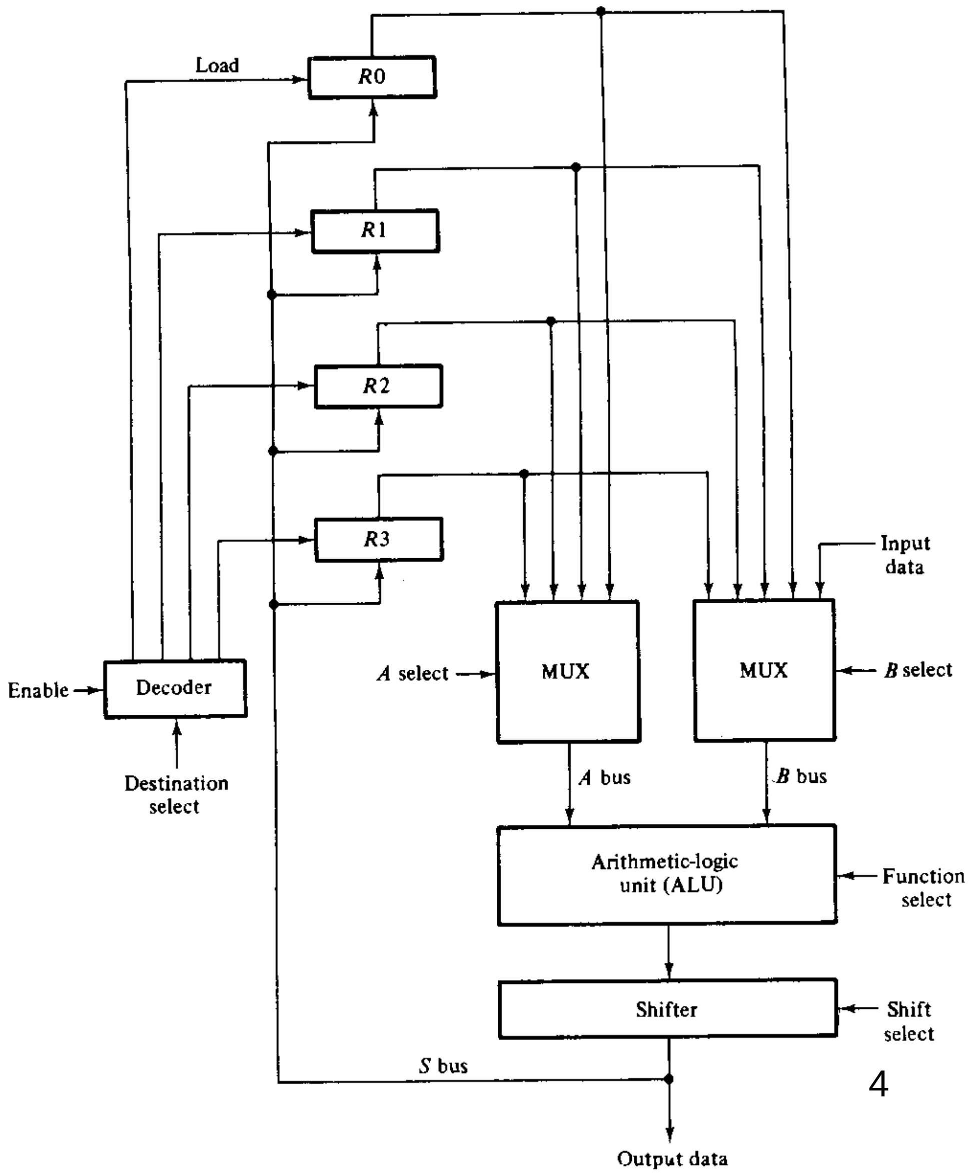
- ❑Terminologies
  - – Processer unit
    - Processes digital data
  - – Control Unit
    - Specifies the operation to be performed by processor
  - – CPU
    - Combination of processor and control unit
  - – ALU
    - A combinational circuit which is able to perform arithmetic and logical operations.

Processor Organization

- Bus Organization

Perform

R1←R2+R3



Processor Organization

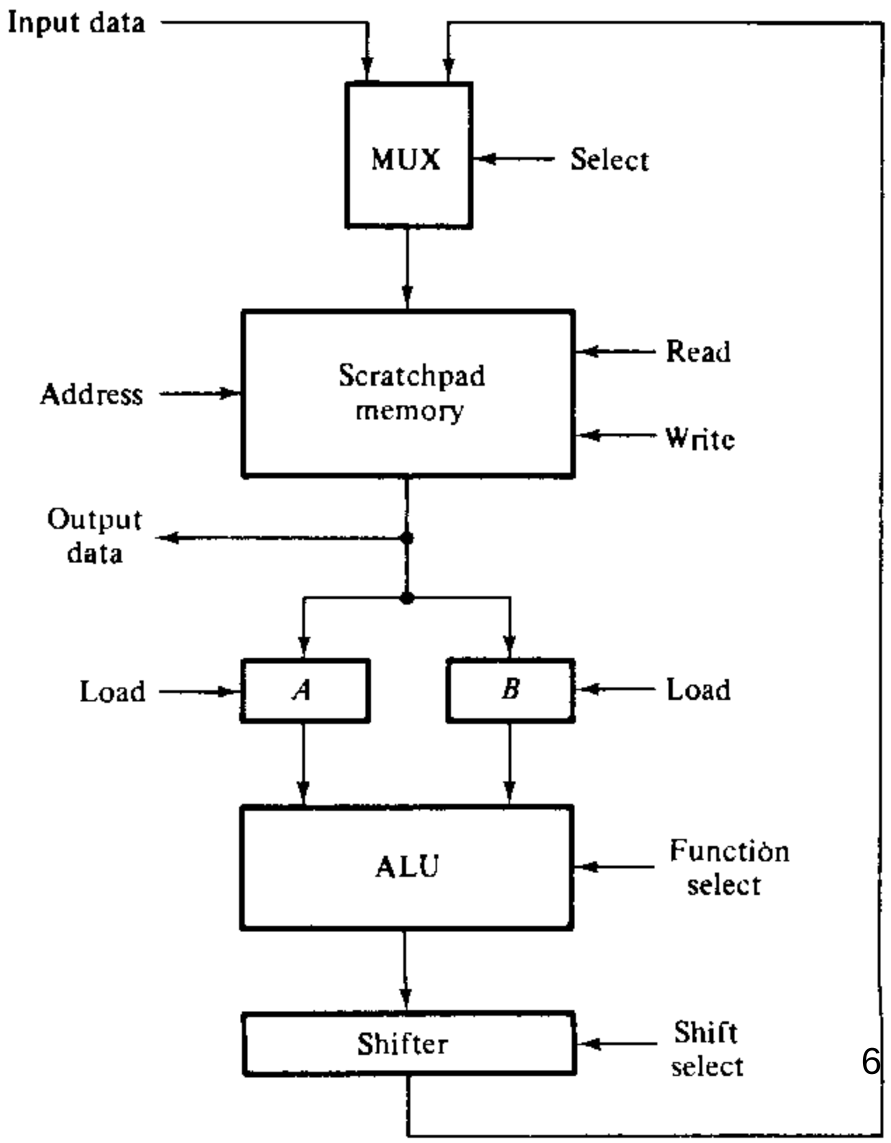
- Scratchpad memory
  - Alternative to many registers a single memory unit can be used
  - Such memory unit is called scratchpad memory.
  - It is different from the main memory.
  - Figure shows a processor with scratchpad memory

Processor Organization

- ❑ Scratchpad memory

Perform

R1←R2+R3

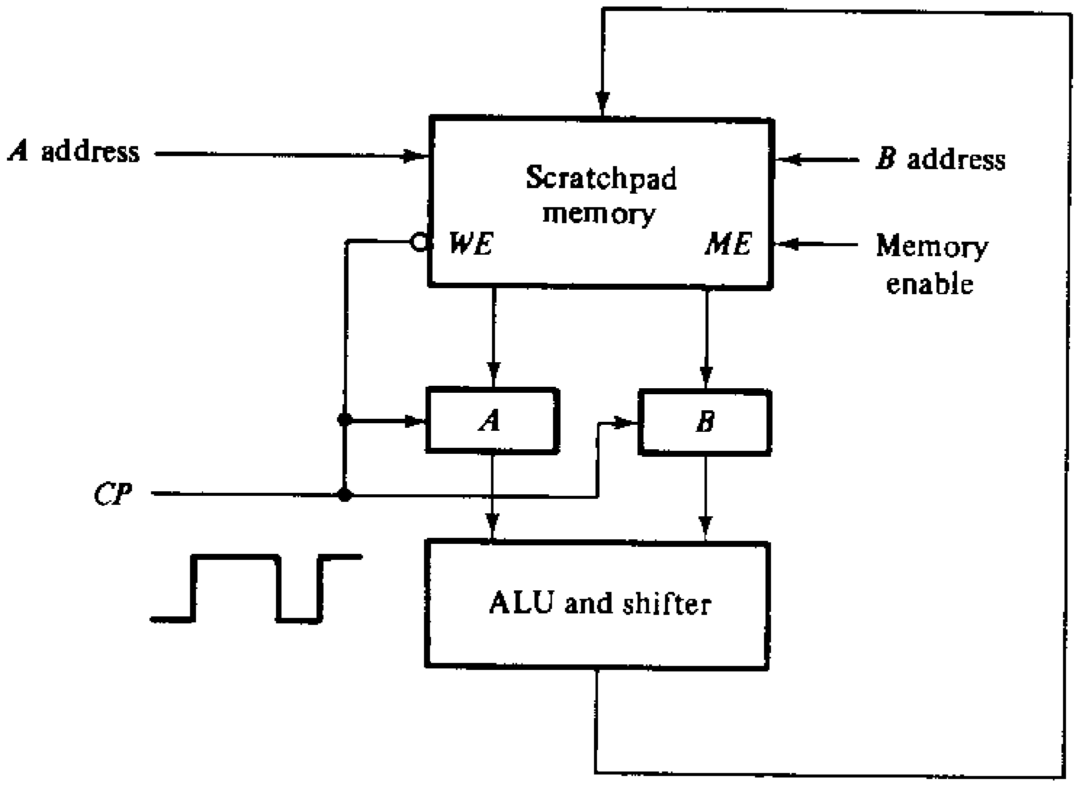


Processor Organization

- Two-port Scratchpad memory

Perform

R1←R2+R3

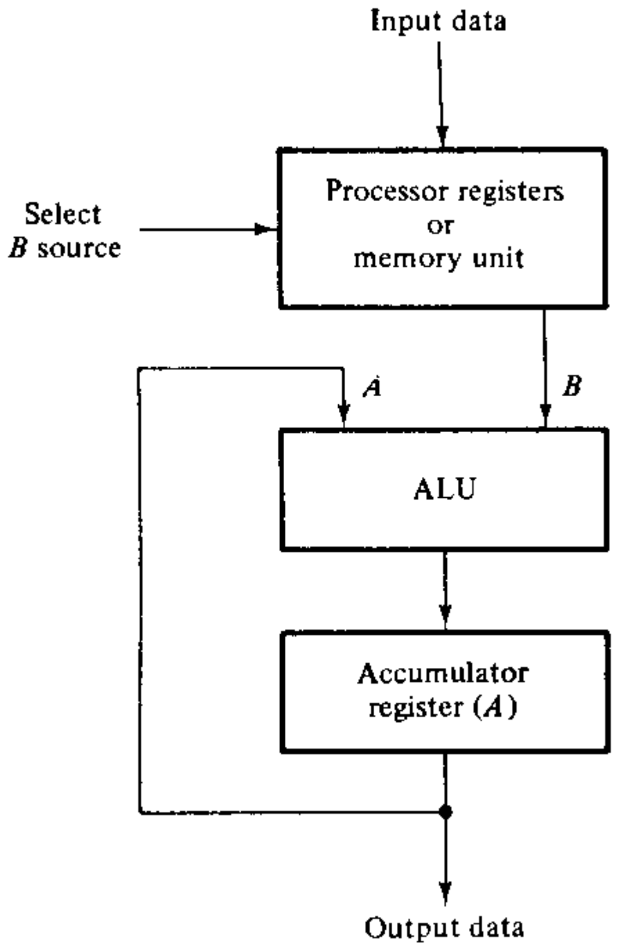


Processor Organization

- ❑Accumulator Register
  - – Different from other memory registers
  - – Used to perform serial addition

Perform

$R1 \leftarrow R2+R3$

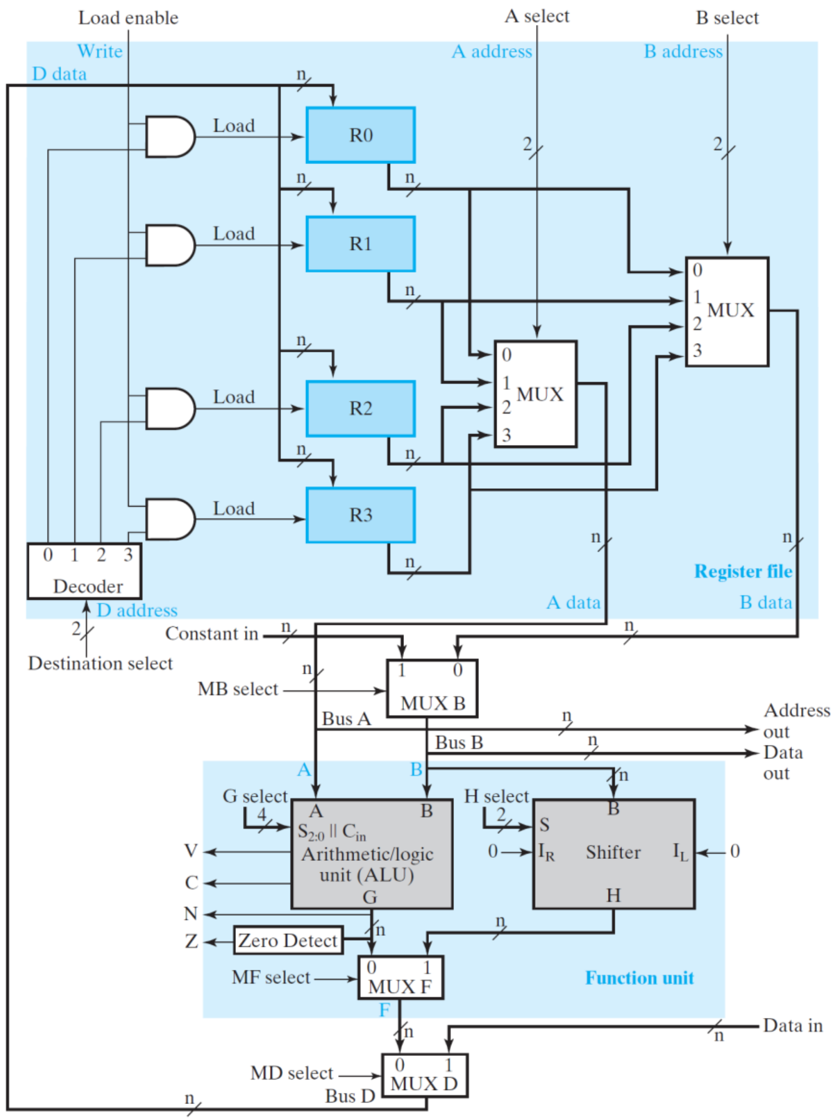


Processor Organization

- Generic Datapath

Perform

R1←R2+R3

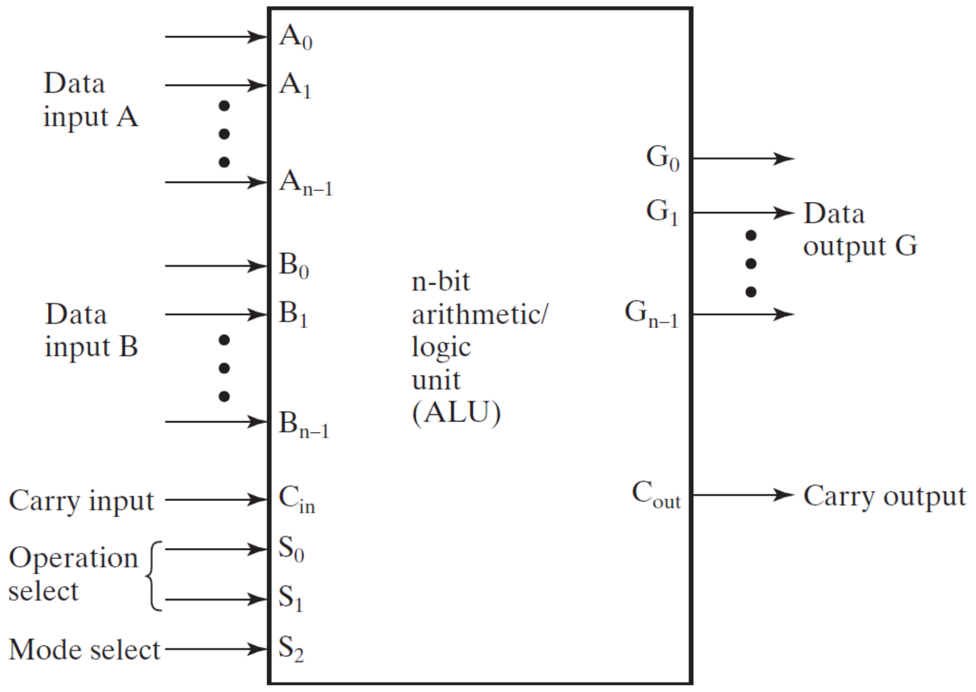


Lesson-12:

Topic	Lesson Learning Outcomes	Teaching-Learning Methodology	Assessment Method
Arithmetic unit	• To understand the design principles of arithmetic unit • To design arithmetic unit with a given set of arithmetic operations.	Class Lecture PBL (Solving some problems in class)	Test, exams, quiz, etc

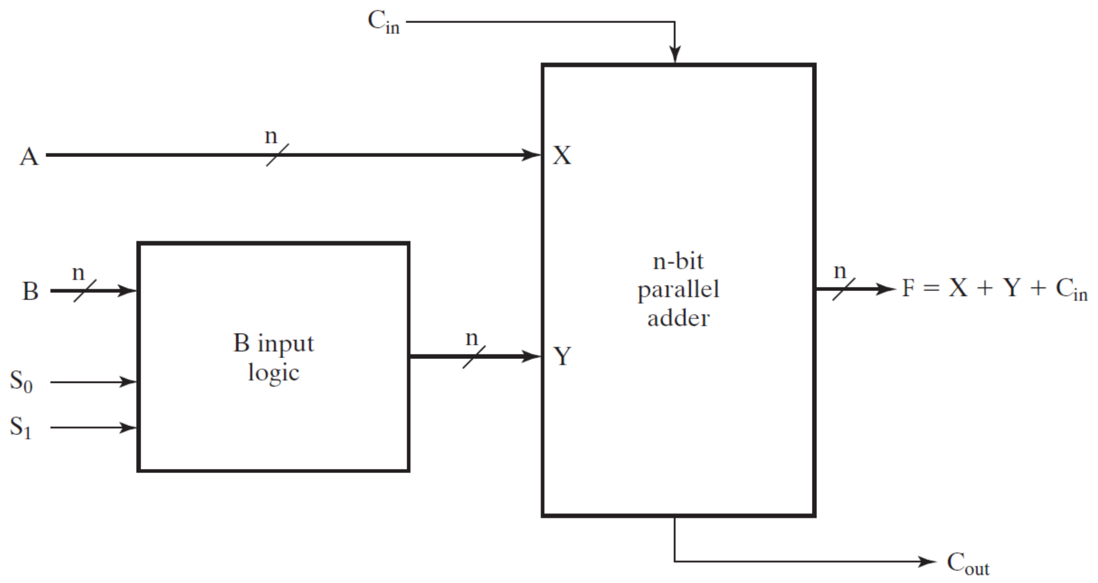
Arithmetic Logic Unit

- The ALU
  - Multi-operational Combinational Circuit.
  - Fig. Shows a n-bit ALU
  - Design Steeps
    - **Design of Arithmetic Unit.**
    - **Design of Logic Unit**
    - **Combine the above two.**



Arithmetic Logic Unit

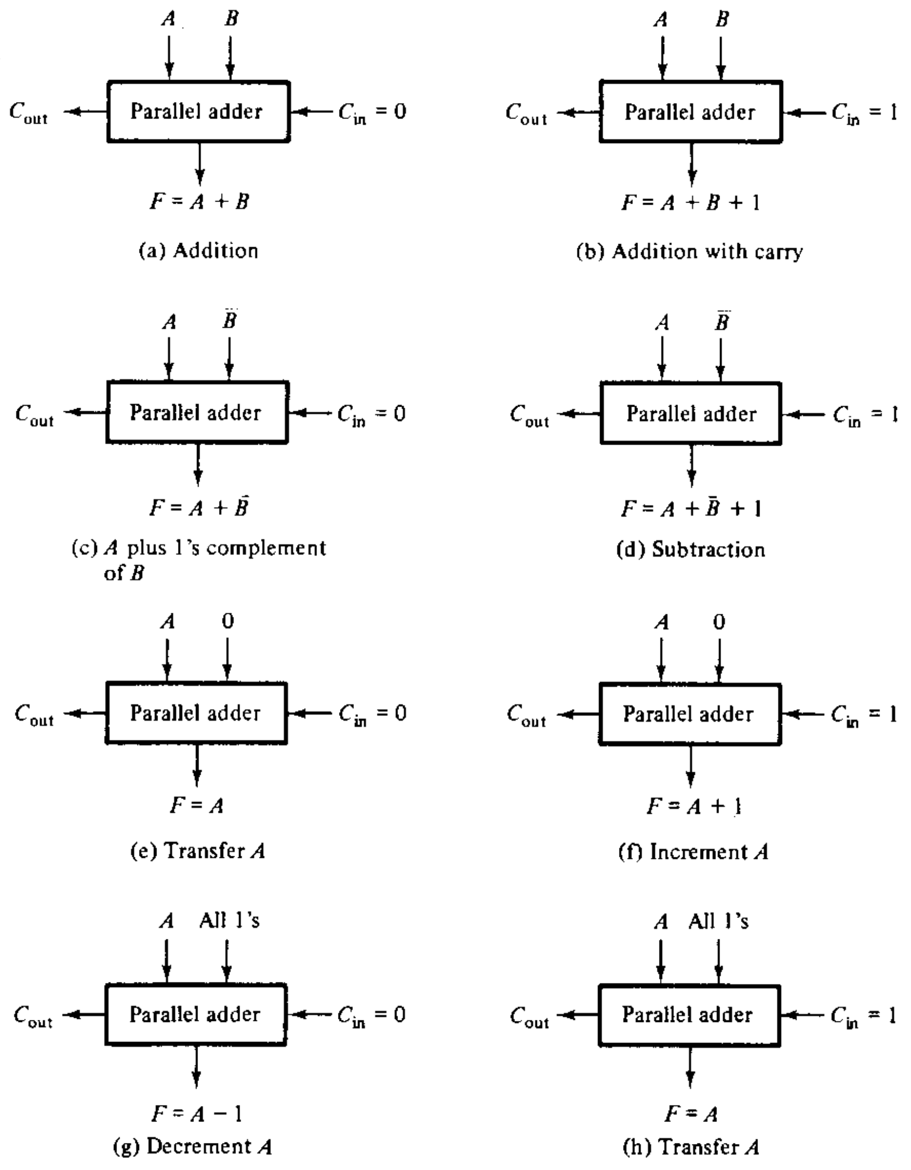
- Design of Arithmetic Unit
  - Basic component is a parallel-adder.
  - Fig. shows a n-bit arithmetic circuit
  - Basic operations to be performed by the full-adder
    - **Addition**
    - **Subtraction**
    - **Increment**
    - **Decrement**
    - **Transfer**



Arithmetic Logic Unit

- ❑ Parallel-adder
  - Operations performed by changing only one variable of PA

Select		Input	$F = (A + Y + C_{in})$	
S_1	S_0	Y	$C_{in} = 0$	$C_{in} = 1$
0	0	all 0s	$G = A$ (transfer)	$G = A + 1$ (increment)
0	1	B	$G = A + B$ (add)	$G = A + B + 1$
1	0	$\overline{B}$	$G = A + \overline{B}$	$G = A + \overline{B} + 1$ (subtract)
1	1	all 1s	$G = A - 1$ (decrement)	$G = A$ (transfer)



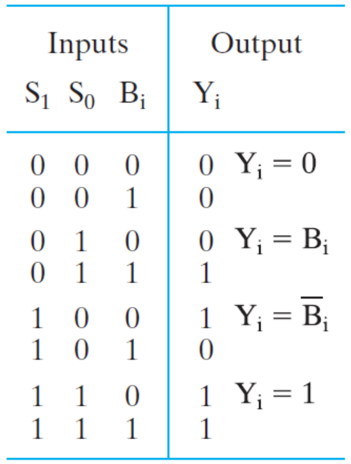
Arithmetic Logic Unit

❑Design of Arithmetic Unit

– To combine all this operations we need 2 selection variable along with carry.

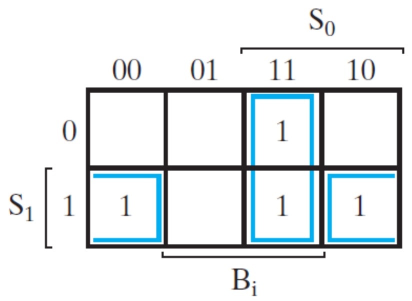
– **Note that the operations can be performed only by changing the B input and the carry.**

– **The selection circuit**

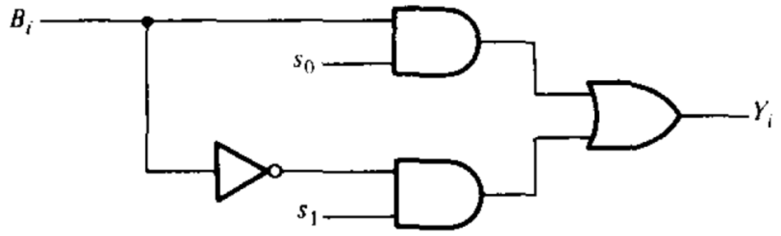


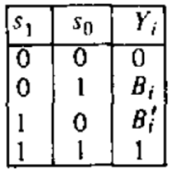
Inputs			Output	
S_1	S_0	B_i	Y_i	
0	0	0	0	$Y_i = 0$
0	0	1	0	
0	1	0	0	$Y_i = B_i$
0	1	1	1	
1	0	0	1	$Y_i = \overline{B}_i$
1	0	1	0	
1	1	0	1	$Y_i = 1$
1	1	1	1	

(a) Truth table



(b) Map simplification:
$Y_i = B_iS_0 + \overline{B}_iS_1$





s_1	s_0	Y_i
0	0	0
0	1	B_i
1	0	B_i'
1	1	1

Arithmetic Logic Unit

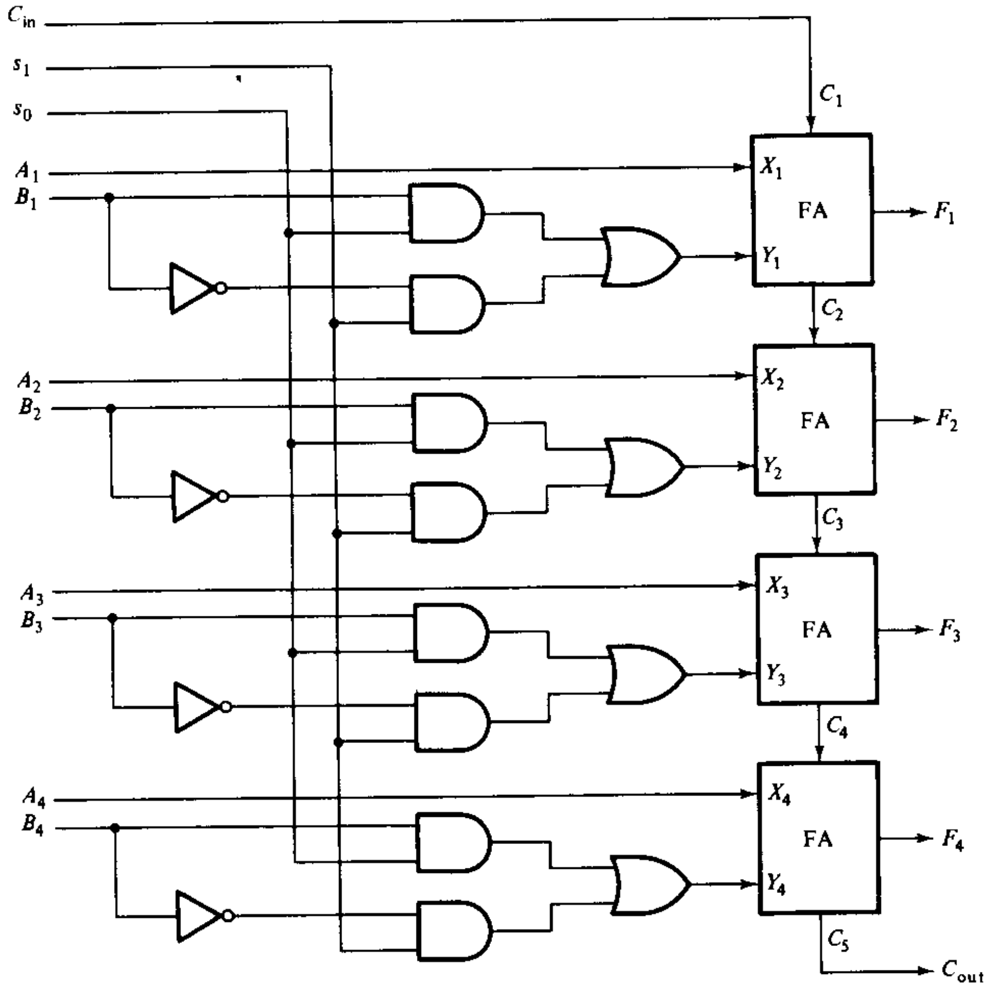
❑Design of Arithmetic Unit

– The logic diagram

$$X_i = A_i$$

$$Y_i = B_i s_0 + B_i' s_1 \qquad i = 1, 2, \ldots, n$$

Function select			Y equals	Output equals	Function
s_1	s_0	C_{in}			
0	0	0	0	$F = A$	Transfer A
0	0	1	0	$F = A + 1$	Increment A
0	1	0	B	$F = A + B$	Add B to A
0	1	1	B	$F = A + B + 1$	Add B to A plus 1
1	0	0	$\overline{B}$	$F = A + \overline{B}$	Add 1's complement of B to A
1	0	1	$\overline{B}$	$F = A + \overline{B} + 1$	Add 2's complement of B to A
1	1	0	All 1's	$F = A - 1$	Decrement A
1	1	1	All 1's	$F = A$	Transfer A



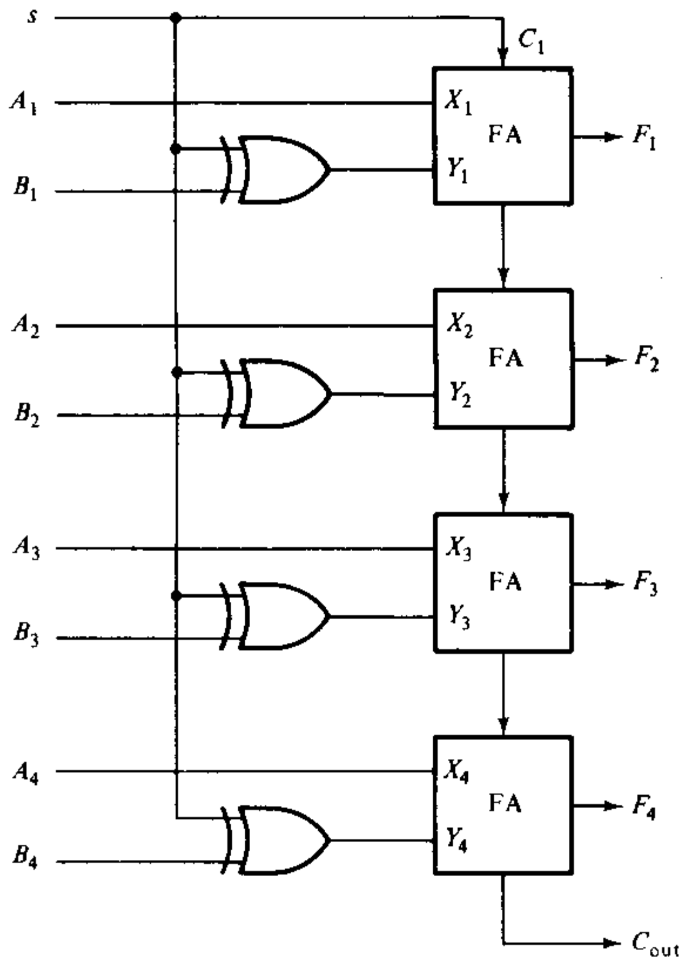
Arithmetic Logic Unit

- Design other arithmetic circuit
  - **Example:** Design an arithmetic circuit with
    - **When s=0 perform A+B**
    - **When s=1 perform A-B**
  - **Solution**
    - **A+B requires**
      - **C_{in}=0, X_i=A_i and Y_i=B_i**
    - **A-B requires**
      - **C_{in}=1, X_i=A_i and Y_i=B_i'**

Arithmetic Logic Unit

- ❑Design other arithmetic circuit

S=0 : Perform A+B
S=1 : Perform A-B

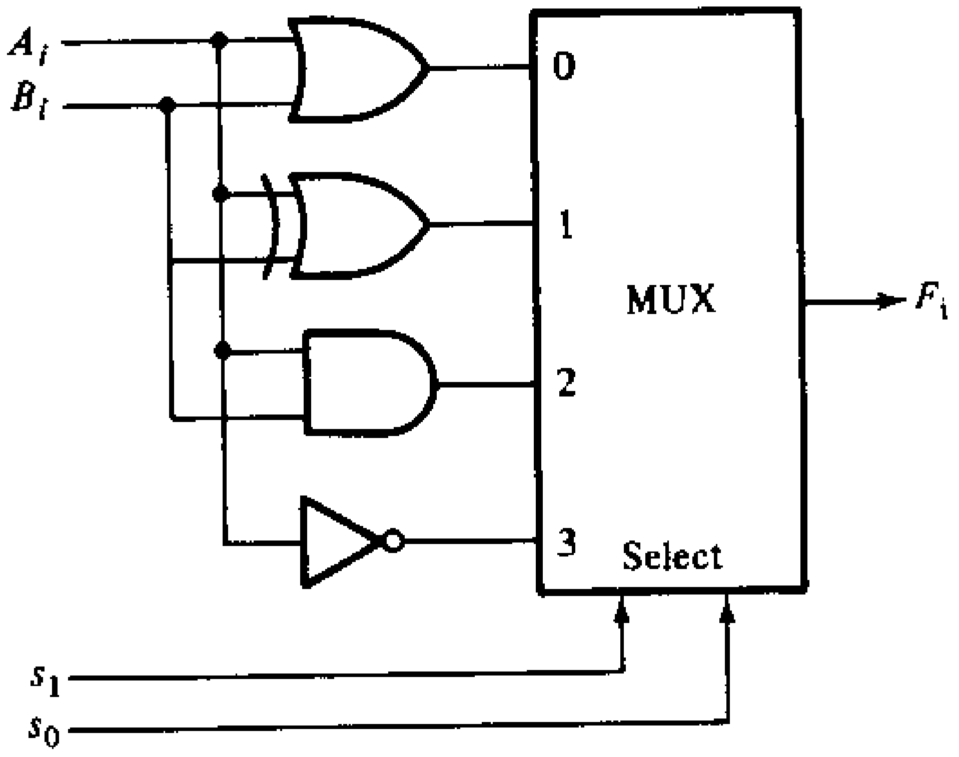


Lesson-13:

Topic	Lesson Learning Outcomes	Teaching-Learning Methodology	Assessment Method
Logical Unit and ALU	• To design of logical unit • To design ALU by combining arithmetic and logic unit.	Class Lecture PBL (Solving some problems in class)	Test, exams, quiz, etc

Arithmetic Logic Unit

- Design of Logic Unit



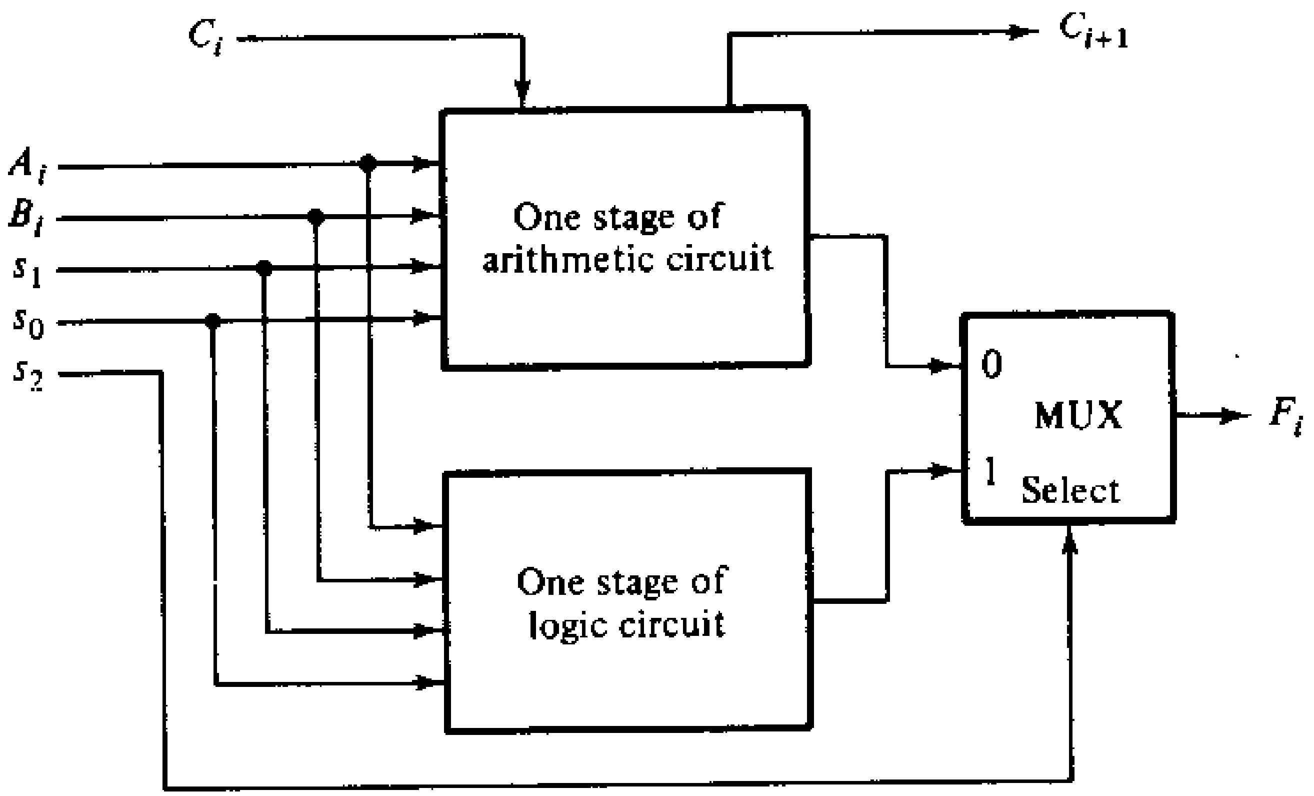
(a) Logic diagram

s_1	s_0	Output	Operation
0	0	$F_i = A_i + B_i$	OR
0	1	$F_i = A_i \oplus B_i$	XOR
1	0	$F_i = A_i B_i$	AND
1	1	$F_i = A_i'$	NOT

(b) Function table

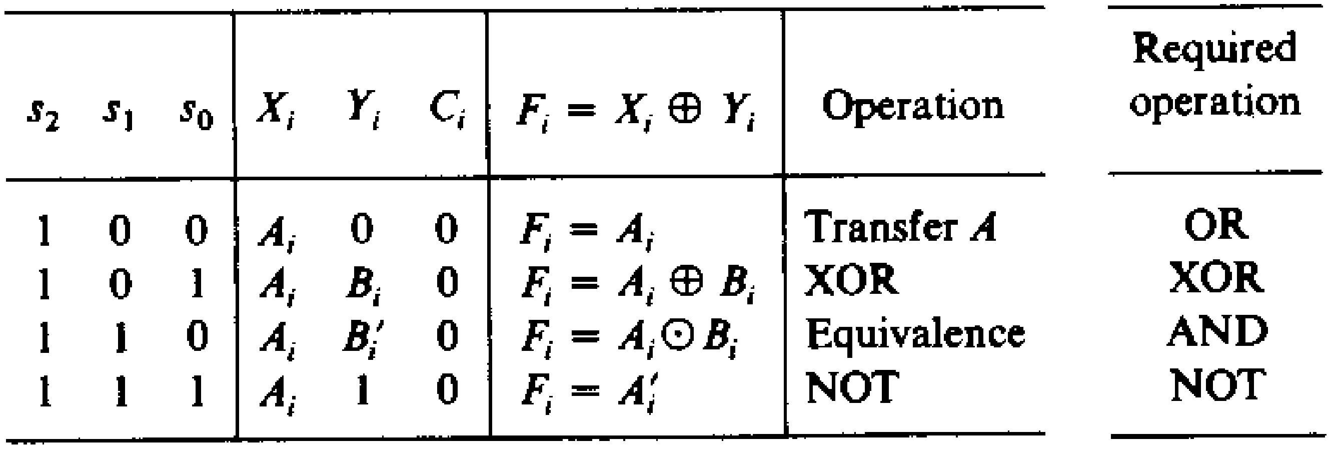
Arithmetic Logic Unit

- ❑Combining Arithmetic and Logic Unit



Arithmetic Logic Unit

- How to Design More Efficient ALU
  - Note
    - **Full-adder with C_{in}=0 performs XOR**
    - **Table lists the possible operations**

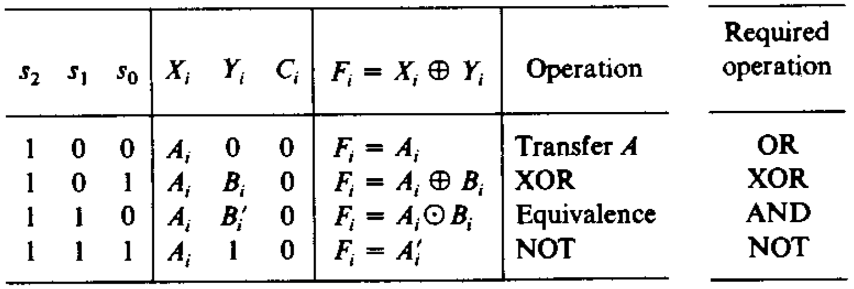


s_2	s_1	s_0	X_i	Y_i	C_i	$F_i = X_i \oplus Y_i$	Operation	Required operation
1	0	0	A_i	0	0	$F_i = A_i$	Transfer A	OR
1	0	1	A_i	B_i	0	$F_i = A_i \oplus B_i$	XOR	XOR
1	1	0	A_i	B_i'	0	$F_i = A_i \odot B_i$	Equivalence	AND
1	1	1	A_i	1	0	$F_i = A_i'$	NOT	NOT

  - Therefore, more efficient ALU design may be possible

Arithmetic Logic Unit

- More Efficient ALU
  - With $s_2s_1s_0$=100
    - Y_i=0, hence we need to change X_i from A_i to A_i+B_i
  - With $s_2s_1s_0$=110
    - Lets make a similarity with previous case
    - Change X_i from A_i to A_i+K_i



s_2	s_1	s_0	X_i	Y_i	C_i	$F_i = X_i \oplus Y_i$	Operation	Required operation
1	0	0	A_i	0	0	$F_i = A_i$	Transfer A	OR
1	0	1	A_i	B_i	0	$F_i = A_i \oplus B_i$	XOR	XOR
1	1	0	A_i	B_i'	0	$F_i = A_i \odot B_i$	Equivalence	AND
1	1	1	A_i	1	0	$F_i = A_i'$	NOT	NOT

$$F_i = X_i \oplus Y_i = (A_i + K_i) \oplus B_i' = A_iB_i + K_iB_i + A_i'K_i'B_i'$$

- Consider $K_i = B_i'$

$$F_i = A_iB_i + B_i'B_i + A_iB_iB_i' = A_iB_i$$

Arithmetic Logic Unit

- More Efficient ALU
  - Finally the inputs of the full adders are

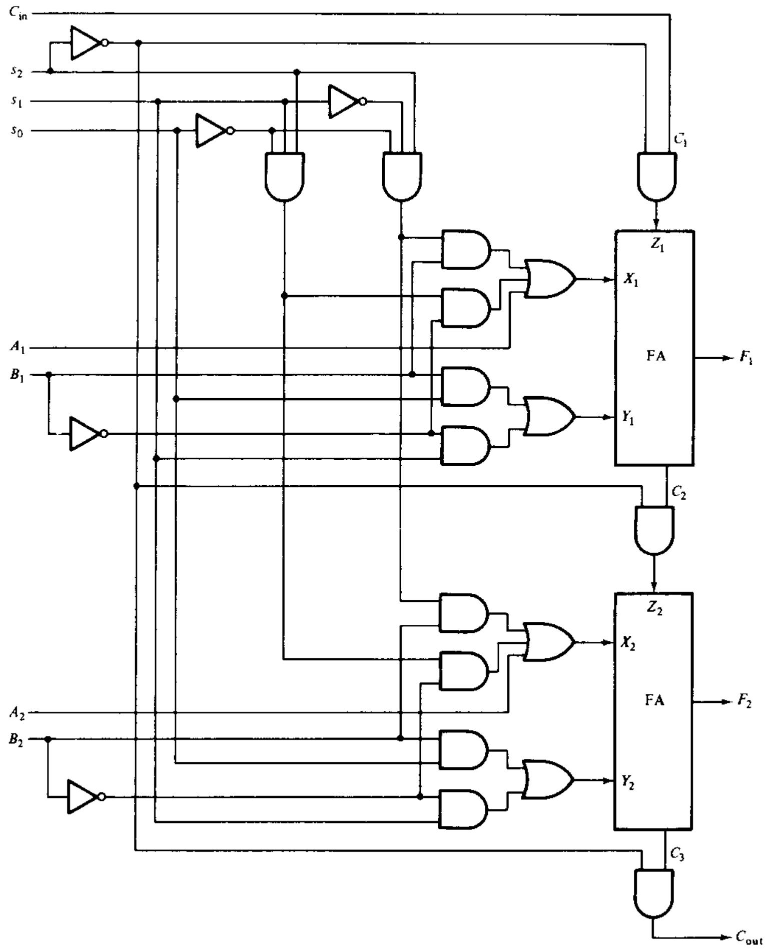
$$X_i = A_i + s_2 s_1' s_0' B_i + s_2 s_1 s_0' B_i'$$
$$Y_i = s_0 B_i + s_1 B_i'$$
$$Z_i = s_2' C_i$$

  - If s_2=0

$$X_i = A_i$$
$$Y_i = s_0 B_i + s_1 B_i'$$
$$Z_i = C_i$$

Arithmetic Logic Unit

- More Efficient ALU

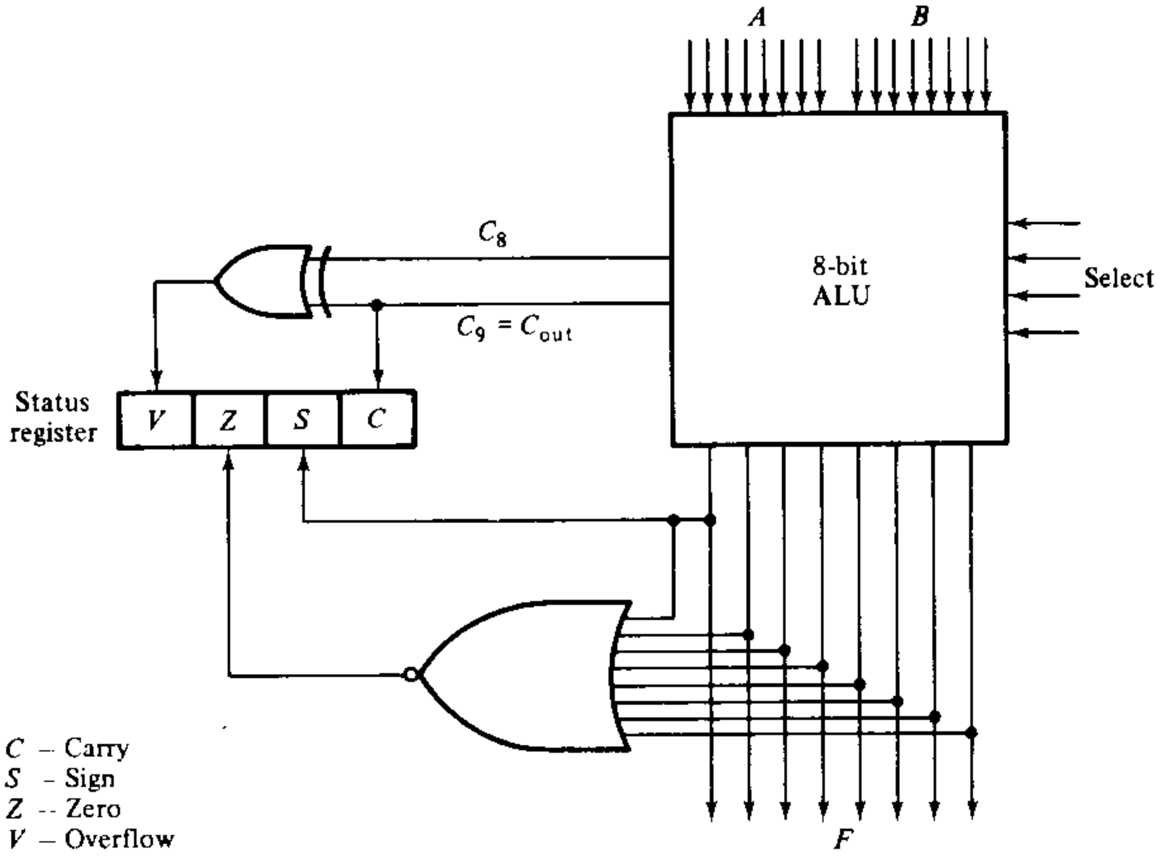


Lesson-14:

Topic	Lesson Learning Outcomes	Teaching-Learning Methodology	Assessment Method
Status register and shifter in ALU	• To understand the design principle of Complete ALU and its components • To know the importance and use of status register in ALU. • To design shift register by using combinational circuit.	Class Lecture PBL (Solving some problems in class)	Test, exams, quiz, etc

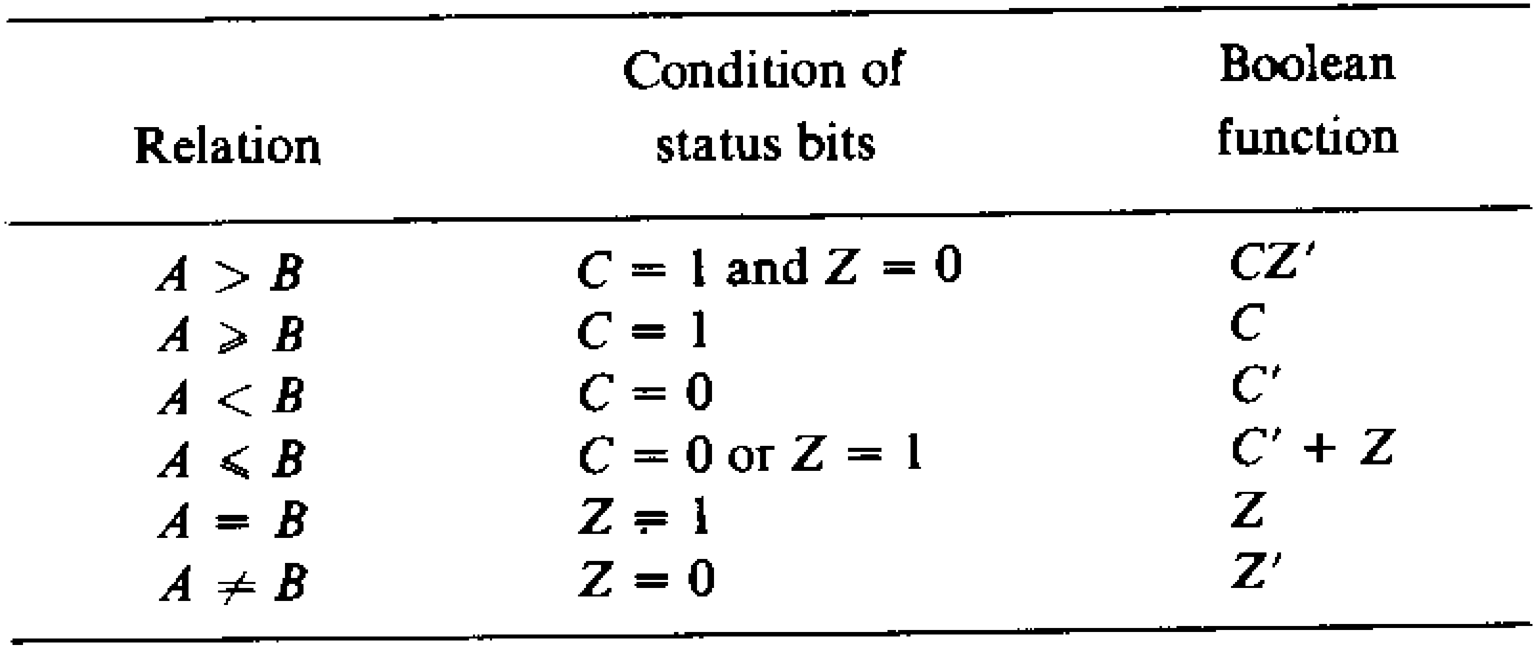
Arithmetic Logic Unit

- Status Register



Arithmetic Logic Unit

- Status bits after subtraction of two unsigned numbers (**A-B**)



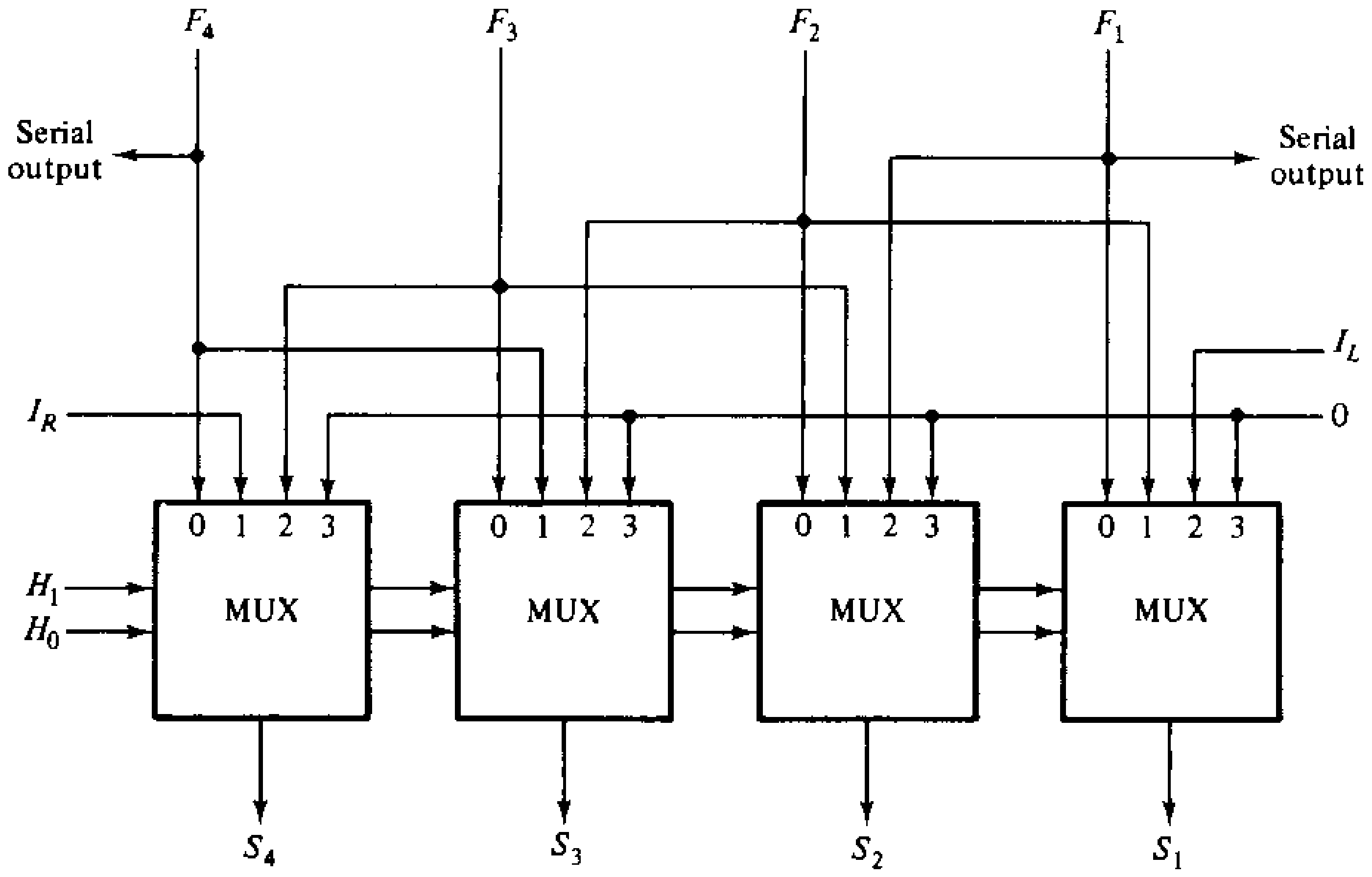
Relation	Condition of status bits	Boolean function
$A > B$	$C = 1$ and $Z = 0$	CZ'
$A \geqslant B$	$C = 1$	C
$A < B$	$C = 0$	C'
$A \leqslant B$	$C = 0$ or $Z = 1$	$C' + Z$
$A = B$	$Z = 1$	Z
$A \neq B$	$Z = 0$	Z'

Design of Shifter

- The Shift Unit
  - Connected after ALU.
    - **Bidirectional Shift register with parallel load can be used.**
      - **But it requires 3 clock-pulses**
    - **A combination circuit that can perform shift operation is a good choice**
      - **Requires only one clock-puls.**

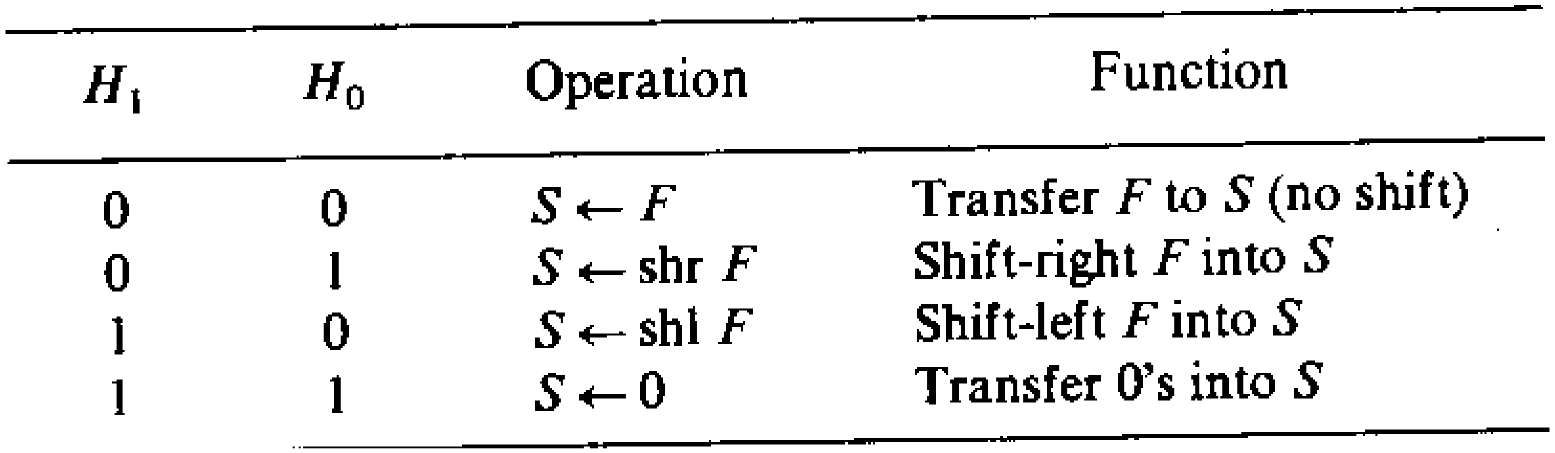
Design of Shifter

- Combinational Shift Unit



Design of Shifter

- Function table of the shifter

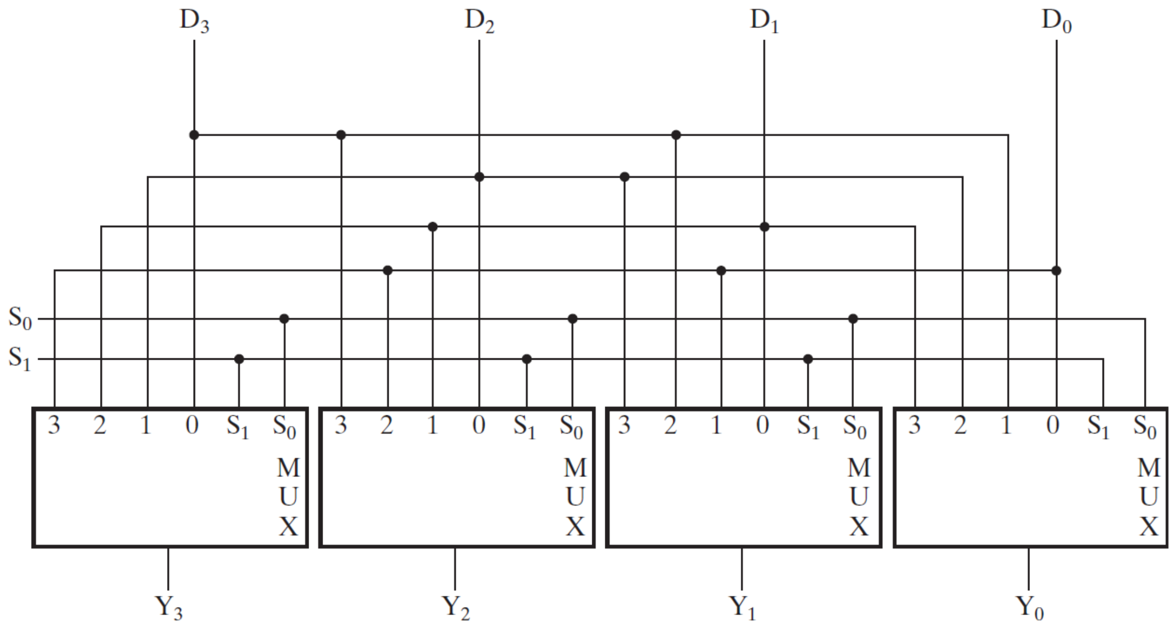


H_1	H_0	Operation	Function
0	0	$S \leftarrow F$	Transfer F to S (no shift)
0	1	$S \leftarrow \text{shr } F$	Shift-right F into S
1	0	$S \leftarrow \text{shl } F$	Shift-left F into S
1	1	$S \leftarrow 0$	Transfer 0's into S

Design of Shifter

❑Barrel shifter

- Often the data must be shifted more than one bit position in a single clock cycle.
- A *barrel shifter* iss designed to do this as shown in Fig.

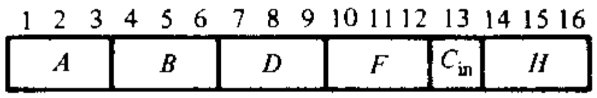


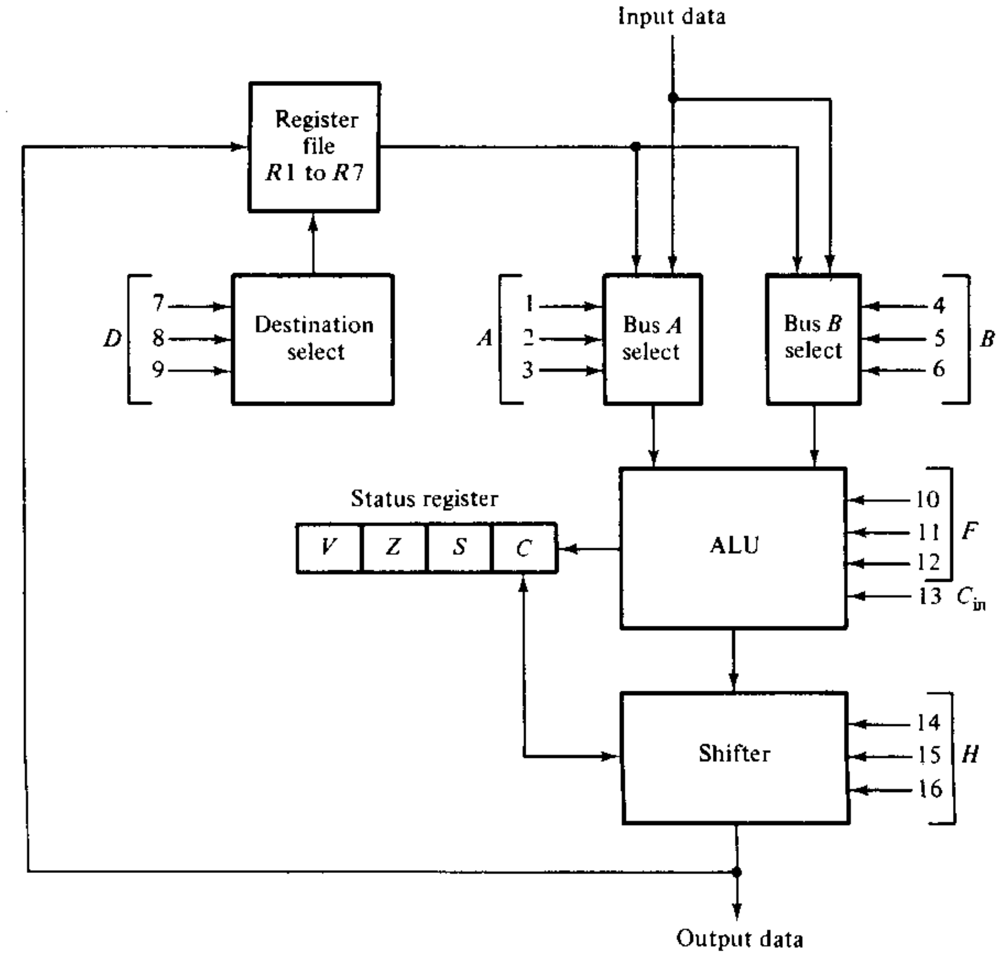
Lesson-15:

Topic	Lesson Learning Outcomes	Teaching-Learning Methodology	Assessment Method
Processor Unit	• To construct a processor unit by using the designed ALU • To understand the control word of processor and how they can use to perform various microoperations	Class Lecture Question and answer	Test, exams, quiz, etc

Processor Unit

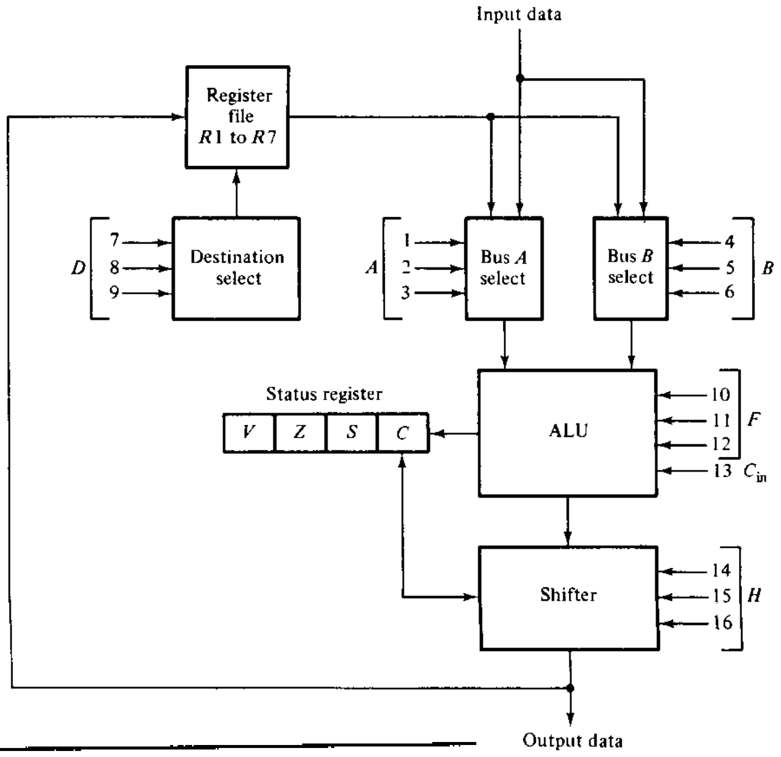
- Block Diagram

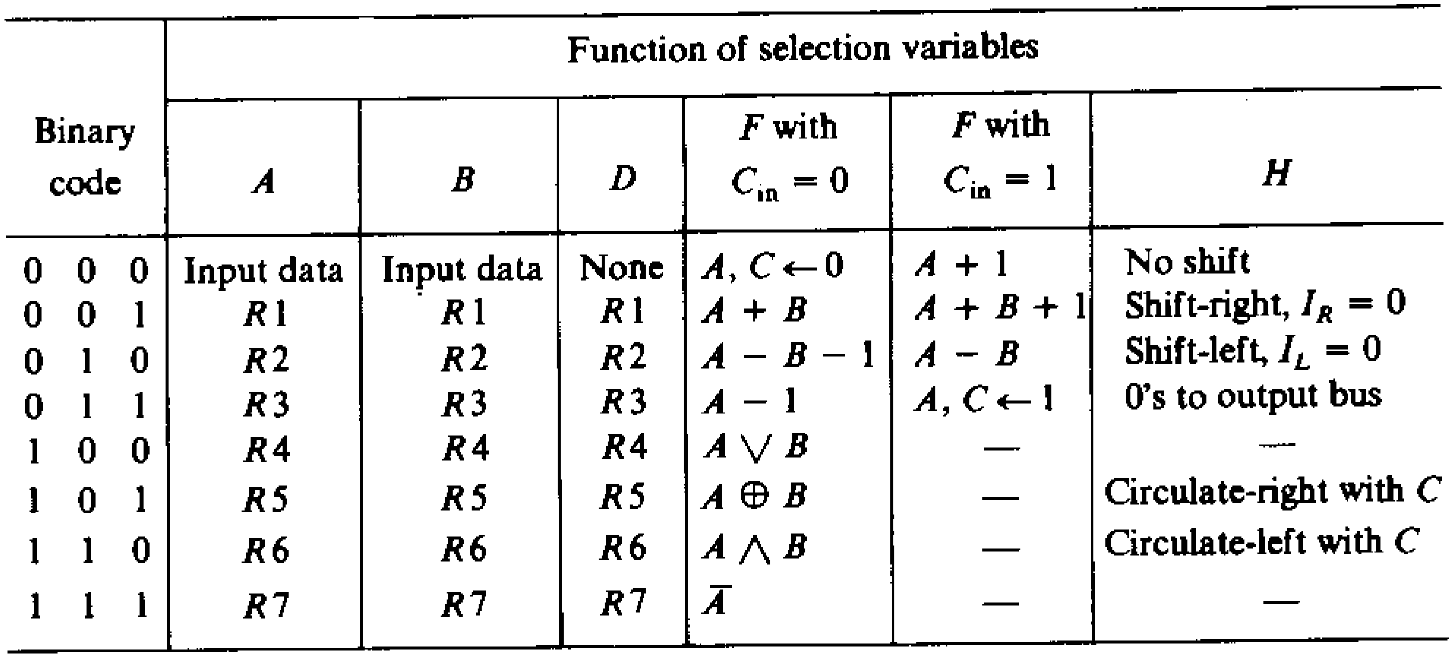




Processor Unit

- Block Diagram with functions

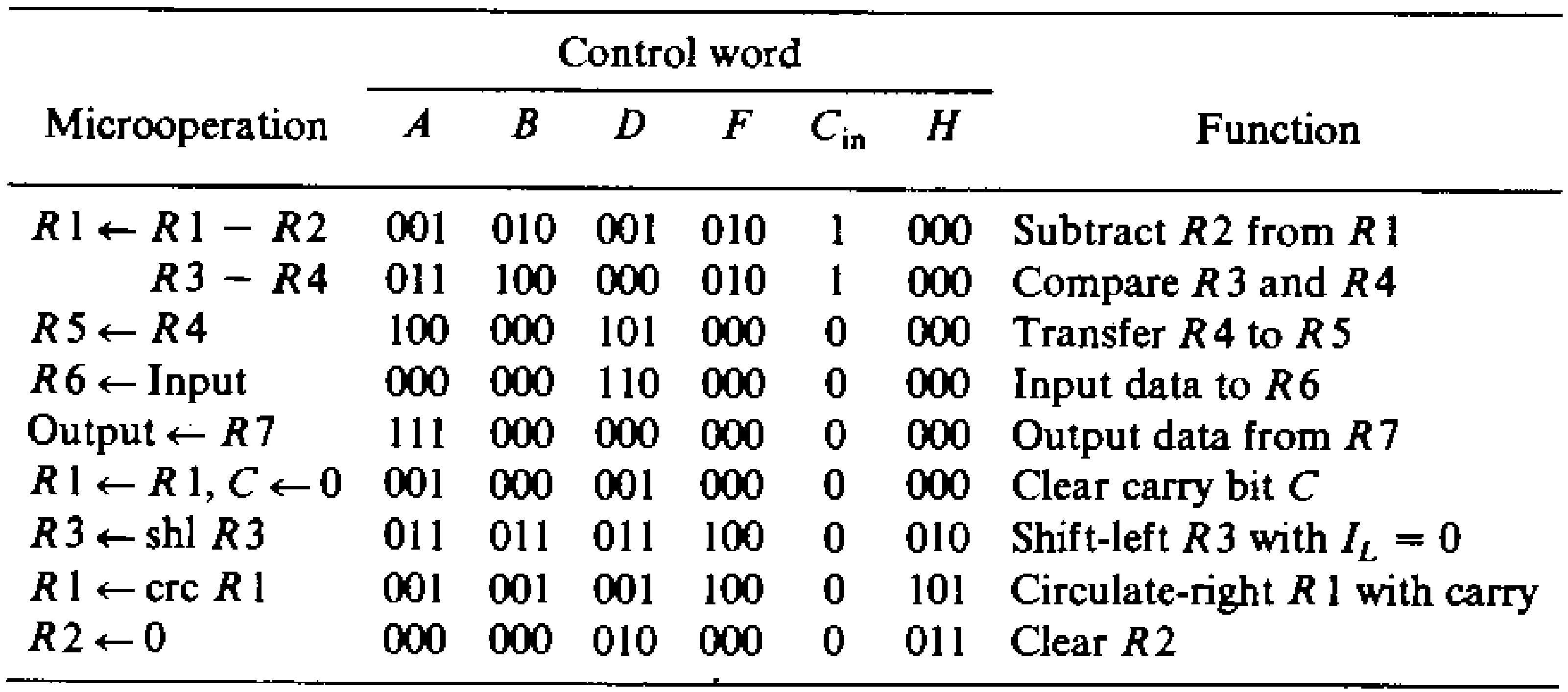




Binary code	Function of selection variables					
	A	B	D	F with $C_{in} = 0$	F with $C_{in} = 1$	H
0 0 0	Input data	Input data	None	$A, C \leftarrow 0$	$A + 1$	No shift
0 0 1	$R1$	$R1$	$R1$	$A + B$	$A + B + 1$	Shift-right, $I_R = 0$
0 1 0	$R2$	$R2$	$R2$	$A - B - 1$	$A - B$	Shift-left, $I_L = 0$
0 1 1	$R3$	$R3$	$R3$	$A - 1$	$A, C \leftarrow 1$	0's to output bus
1 0 0	$R4$	$R4$	$R4$	$A \vee B$	—	—
1 0 1	$R5$	$R5$	$R5$	$A \oplus B$	—	Circulate-right with C
1 1 0	$R6$	$R6$	$R6$	$A \wedge B$	—	Circulate-left with C
1 1 1	$R7$	$R7$	$R7$	$\bar{A}$	—	—

Processor Unit

- Examples of Micro-operations



Microoperation	Control word A	B	D	F	C_{in}	H	Function
$R1 \leftarrow R1 - R2$	001	010	001	010	1	000	Subtract $R2$ from $R1$
$R3 - R4$	011	100	000	010	1	000	Compare $R3$ and $R4$
$R5 \leftarrow R4$	100	000	101	000	0	000	Transfer $R4$ to $R5$
$R6 \leftarrow$ Input	000	000	110	000	0	000	Input data to $R6$
Output $\leftarrow R7$	111	000	000	000	0	000	Output data from $R7$
$R1 \leftarrow R1, C \leftarrow 0$	001	000	001	000	0	000	Clear carry bit C
$R3 \leftarrow$ shl $R3$	011	011	011	100	0	010	Shift-left $R3$ with $I_L = 0$
$R1 \leftarrow$ crc $R1$	001	001	001	100	0	101	Circulate-right $R1$ with carry
$R2 \leftarrow 0$	000	000	010	000	0	011	Clear $R2$

Processor Unit

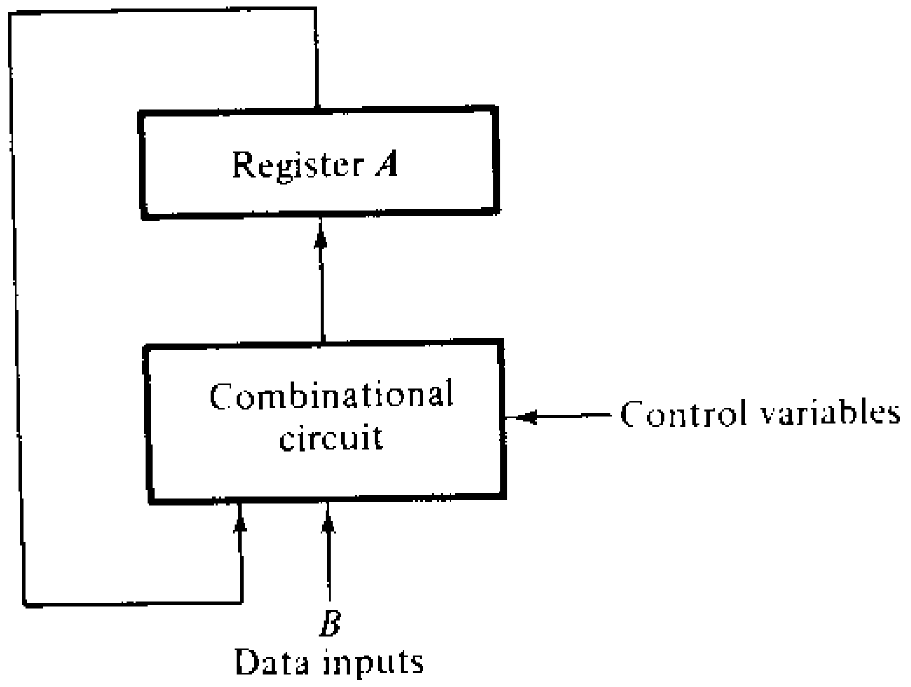
- Other operations
  - How to shift the contents of a specific register?
  - **How to clear a specific register?**
    - **There are several ways to perform these operations**
    - **We need to find the simplest one.**

Lesson-16:

Topic	Lesson Learning Outcomes	Teaching-Learning Methodology	Assessment Method
Design of Accumulator	• To identify the requirements of accumulator design • To design an accumulator using JK flipflop.	Class Lecture Question and answer	Test, exams, quiz, etc

Design of Accumulator

- Accumulator
  - Not only the ‘register A’ but the combinational circuit is also a part of the accumulator.

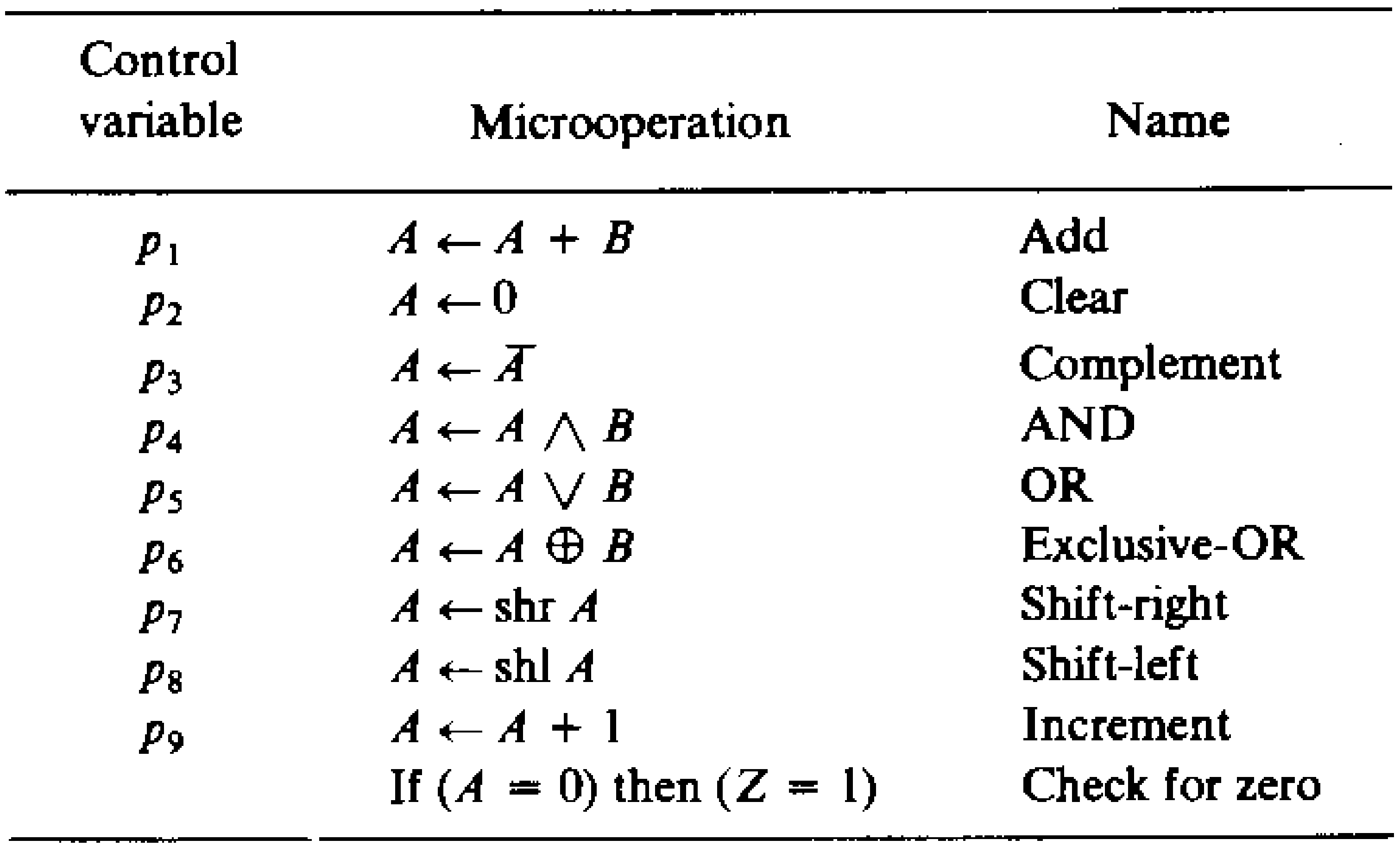


Design of Accumulator

- ❑ Accumulator
  - – A multi-functional register
    - **Not only perform the operations of commonly used register.**
    - **It should be perform all the micro-operations in a processor unit.**
    - **Also termed as multipurpose operational register.**
    - **The design of accumulator is dependent on the number of micro-operations it should be performed.**

Design of Accumulator

- List of micro operations for an accumulator



Control variable	Microoperation	Name
p_1	$A \leftarrow A + B$	Add
p_2	$A \leftarrow 0$	Clear
p_3	$A \leftarrow \overline{A}$	Complement
p_4	$A \leftarrow A \wedge B$	AND
p_5	$A \leftarrow A \vee B$	OR
p_6	$A \leftarrow A \oplus B$	Exclusive-OR
p_7	$A \leftarrow \text{shr } A$	Shift-right
p_8	$A \leftarrow \text{shl } A$	Shift-left
p_9	$A \leftarrow A + 1$	Increment
	If $(A = 0)$ then $(Z = 1)$	Check for zero

Design of Accumulator

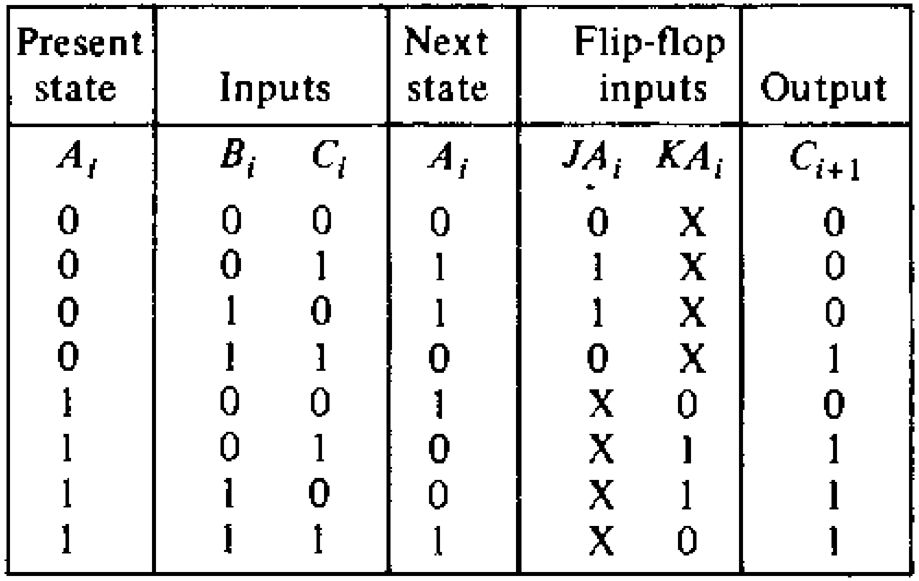
- Design procedure
  - Each stage of the accumulator should be designed separately
    - **Stages A_i should connected in sequence $i=1,\ldots,n$.**
    - **Each stage should be deigned with a flip-flop**
      - **In this example we will use JK flip-flop,**
      - **Characteristics of JK flip-flop**

$$Q(t+1) = JQ' + K'Q$$

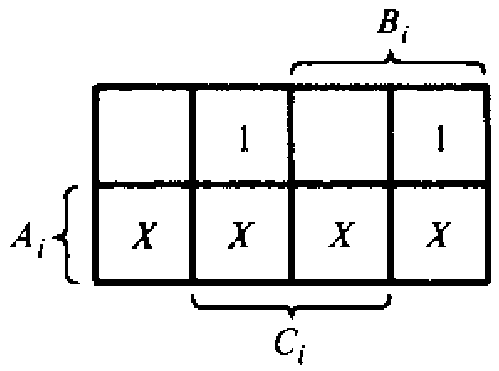
    - **Control variables should be mutually exclusive.**

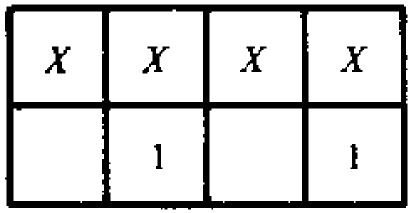
Design of Accumulator

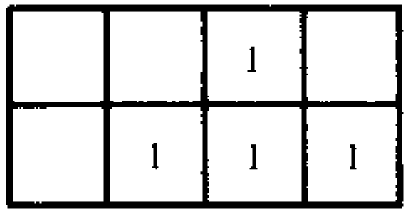
- ❑Design procedure
  - – Add **B** to **A**
    - – Control variable ***p_1***



Present state	Inputs		Next state	Flip-flop inputs		Output
A_i	B_i	C_i	A_i	JA_i	KA_i	C_{i+1}
0	0	0	0	0	X	0
0	0	1	1	1	X	0
0	1	0	1	1	X	0
0	1	1	0	0	X	1
1	0	0	1	X	0	0
1	0	1	0	X	1	1
1	1	0	0	X	1	1
1	1	1	1	X	0	1







$JA_i = B_iC_i' + B_i'C_i$

$KA_i = B_iC_i' + B_i'C_i$

$C_{i+1} = A_iB_i + A_iC_i + B_iC_i$

Design of Accumulator

- Design procedure
  - Add **B** to **A** (control variable ***p**₁*)
    - **The Boolean function with control variable**

$$JA_i = B_iC_i'p_1 + B_i'C_ip_1$$

$$KA_i = B_iC_i'p_1 + B_i'C_ip_1$$

$$C_{i+1} = A_iB_i + A_iC_i + B_iC_i$$

Design of Accumulator

❑Design procedure

- Clear **A** (***p₂***)
  - **We need only apply control variable to K.**

$$Q(t+1) = JQ' + K'Q$$

  - **The Boolean function**

$$JA_i = 0$$

$$KA_i = p_2$$

- Complement (***p₃***)
  - **The Boolean function**

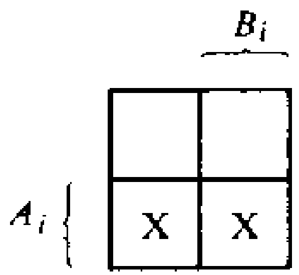
$$JA_i = p_3$$

$$KA_i = p_3$$

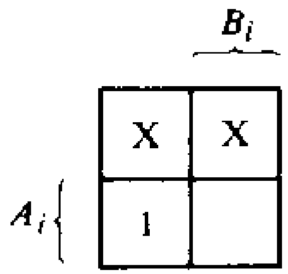
Design of Accumulator

- Design procedure
  - AND (p_4)

Present state	Input	Next state	Flip-flop inputs	
A_i	B_i	A_i	JA_i	KA_i
0	0	0	0	X
0	1	0	0	X
1	0	0	X	1
1	1	1	X	0



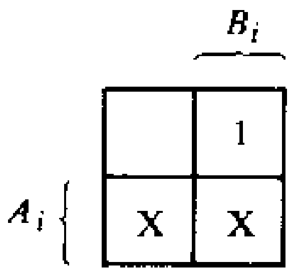
$JA_i = 0$



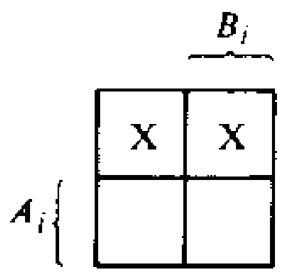
$KA_i = B'_i$

  - OR (p_5)

Present state	Input	Next state	Flip-flop inputs	
A_i	B_i	A_i	JA_i	KA_i
0	0	0	0	X
0	1	1	1	X
1	0	1	X	0
1	1	1	X	0



$JA_i = B_i$

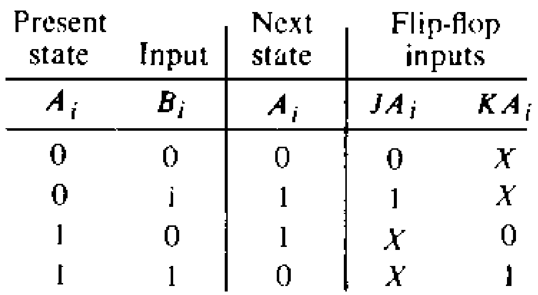


$KA_i = 0$

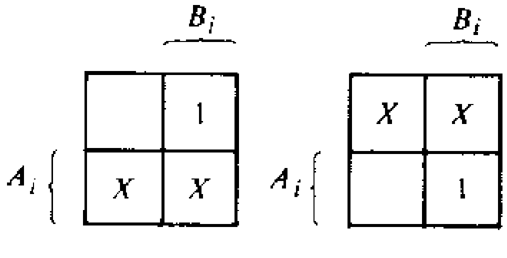
Design of Accumulator

❑Design procedure

– XOR (***p$_6$***)



Present state	Input	Next state	Flip-flop inputs	
A_i	B_i	A_i	JA_i	KA_i
0	0	0	0	X
0	1	1	1	X
1	0	1	X	0
1	1	0	X	1



$JA_i = B_i$ $KA_i = B_i$

– Shift right (***p$_7$***)

$$JA_i = A_{i+1}p_7$$
$$KA_i = A'_{i+1}p_7$$

– Shift left (***p$_8$***)

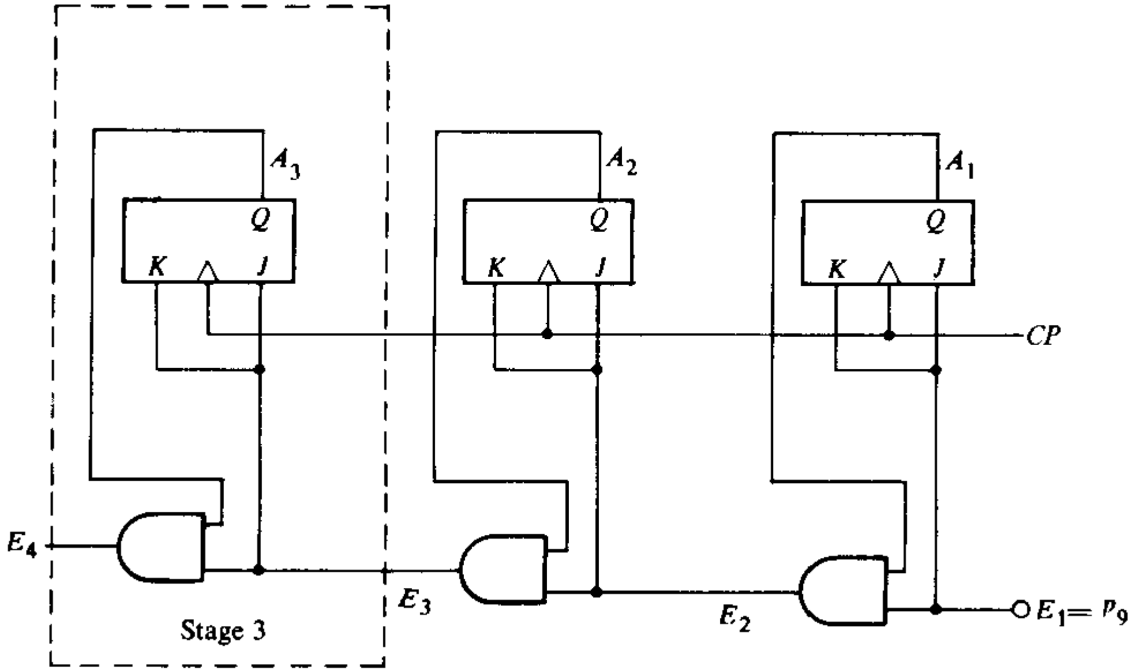
$$JA_i = A_{i-1}p_8$$
$$KA_i = A'_{i-1}p_8$$

Design of Accumulator

❑Design procedure

– Increment (***p*₉**)

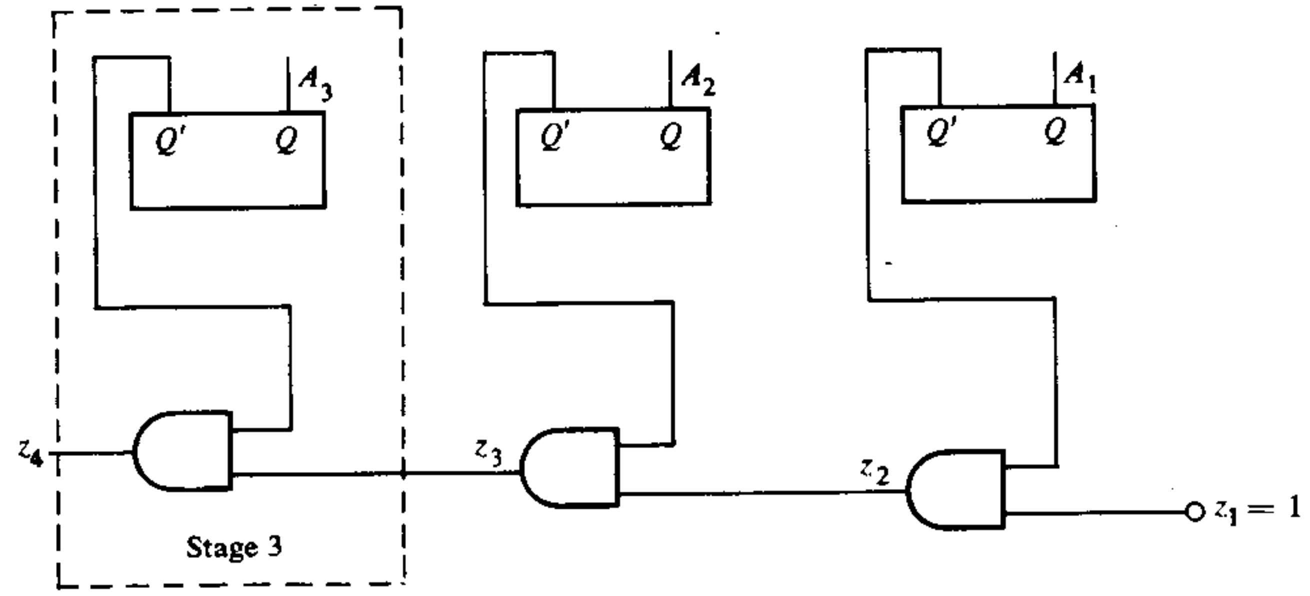
- **Similar with synchronous counter of fig. 7-17**



$$JA_i = E_i$$
$$KA_i = E_i$$
$$E_{i+1} = E_i A_i \qquad i = 1, 2, \ldots, n$$
$$E_1 = p_9$$

Design of Accumulator

- Design procedure
  - Check for zero (***Z***)



$$z_{i+1} = z_i A_i' \qquad i = 1, 2, \ldots, n$$

$$z_1 = 1$$

$$z_{n+1} = Z$$

Design of Accumulator

- Design procedure
  - One stage of accumulator

$$JA_i = B_iC_i'p_1 + B_i'C_ip_1 + p_3 + B_ip_5 + B_ip_6 + A_{i+1}p_7 + A_{i-1}p_8 + E_i$$

$$KA_i = B_iC_i'p_1 + B_i'C_ip_1 + p_2 + p_3 + B_i'p_4 + B_ip_6 + A_{i+1}'p_7$$
$$+ A_{i-1}'p_8 + E_i$$

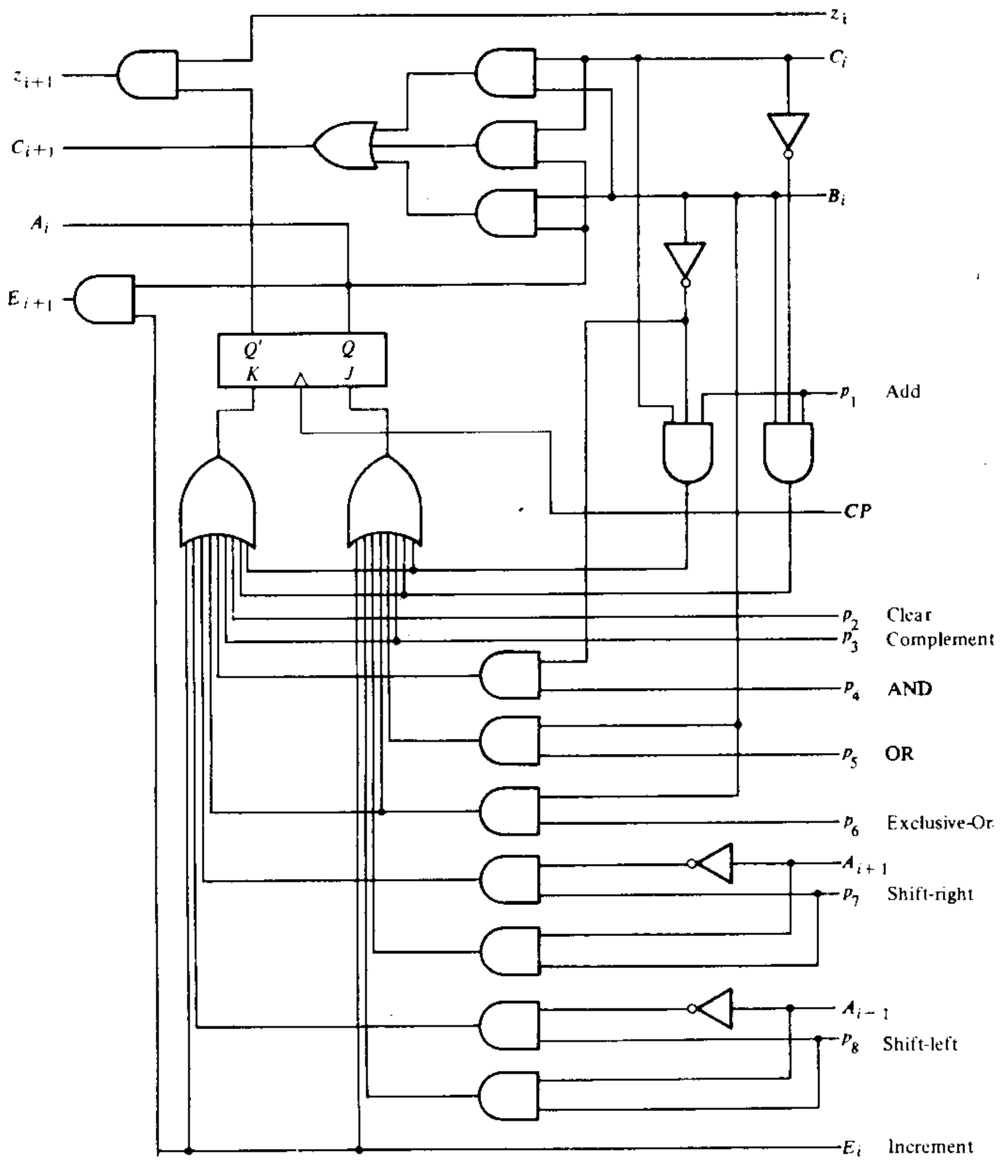
$$C_{i+1} = A_iB_i + A_iC_i + B_iC_i$$

$$E_{i+1} = E_iA_i$$

$$z_{i+1} = z_iA_i'$$

Design of Accumulator

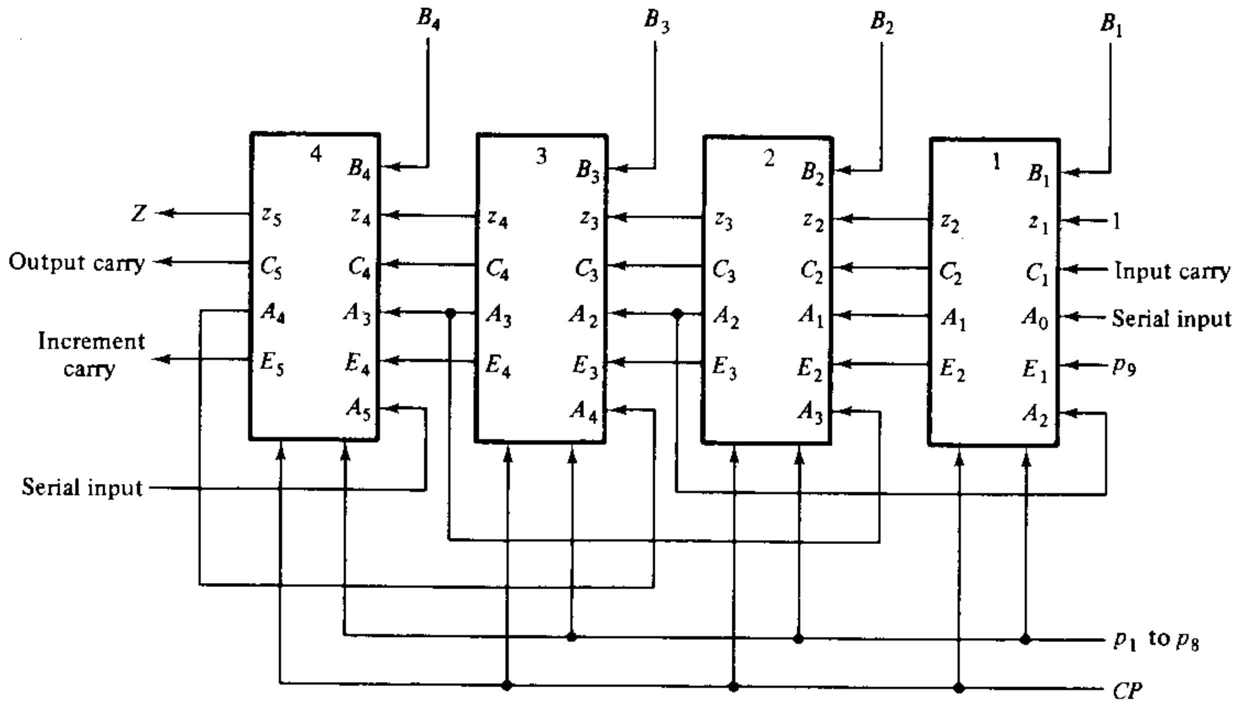
- Design procedure
  - One stage of accumulator



Design of Accumulator

❑Design procedure

- Complete accumulator
- 4-bit accumulator with 4 stages



Lesson-17:

Topic	Lesson Learning Outcomes	Teaching-Learning Methodology	Assessment Method
Solving problems of Chapter-9	• To improve the problem solving skills	Class Lecture Homework	Test, exams, quiz, assignment, etc