Chapter 1

Computer Organisation & Architecture

- Computer:- An electronic device for storing and processing data, typically in binary form, according to the instruction given it in a variable program.
- Computer system: Hardware and Software.
- Computer Network:

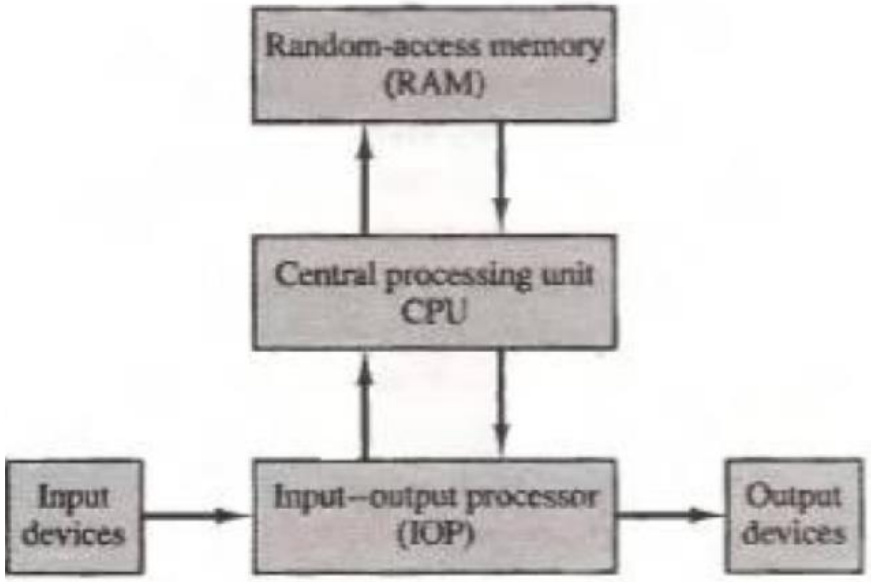


Figure 1-1 Block diagram of a digital computer.

- Computer Organisation: H/w component operate + connected together to form computer system.
- Computer Design: H/W design of the computer.
- Computer Architecture: structure + behaviour of the computer.

Digital system

- Digital System: Interconnection of digital h/w module.

Module:- register, decoder, control logic units etc.

Micro-operation:- The operation executed on the data store in register.

Eg: shift, load etc.

The internal h/w organization of a digital computer is define by:

1. Set of register and its function,
2. The sequence of micro-operation, and
3. The control.

Register Transfer Language

- Symbolic notation to represent the sequence of micro-operation and control function.
- Register:- Computer register are designated by capital letter

Eg:- MAR (Memory Address Register)

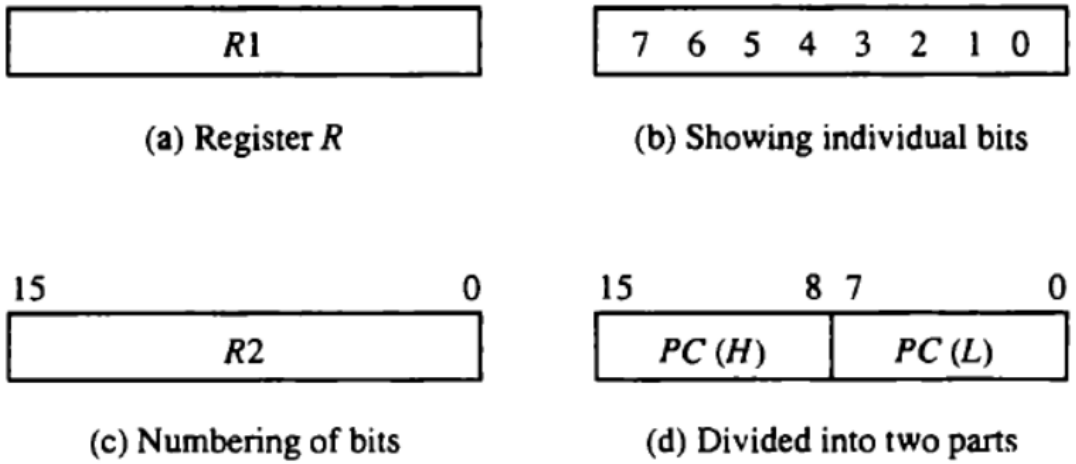
PC (program counter)

IR(instruction register)

R1(processor register)

Register Transfer Language:

Figure 4-1 Block diagram of register.



(a) Register *R*

(b) Showing individual bits

(c) Numbering of bits

(d) Divided into two parts

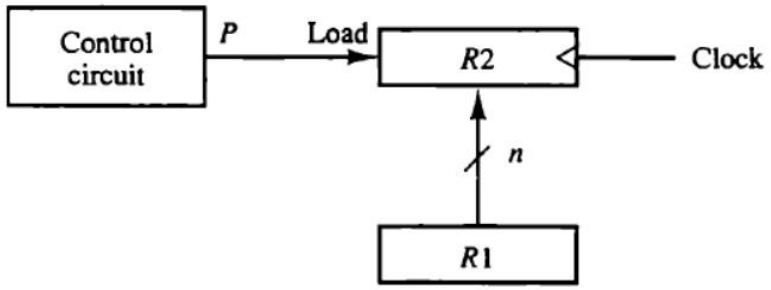
Register Transfer Language

- R2 ← R1; (transfer of the content of register R1 into register R2).
- If (P=1) then (R2 ← R1)

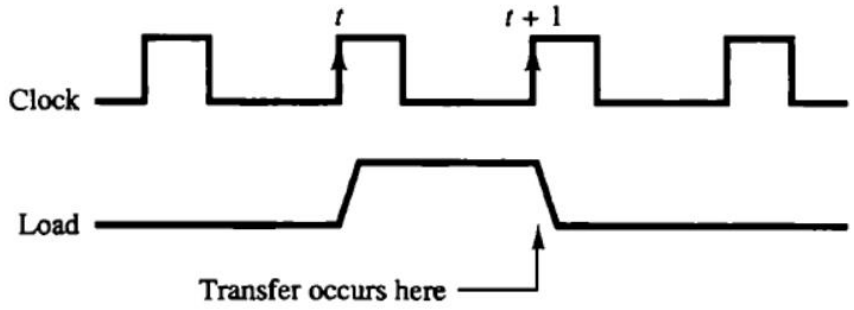
P is the Boolean variable = 0 or 1.

It is written as: P : R2←R1.

Figure 4-2 Transfer from R1 to R2 when $P = 1$.



(a) Block diagram



(b) Timing diagram

Register Transfer Language:

- Denotes the common operation in common clock pulse provided T=1.

$$T: \quad R2 \leftarrow R1, \quad R1 \leftarrow R2$$

TABLE 4-1 Basic Symbols for Register Transfers

Symbol	Description	Examples
Letters (and numerals)	Denotes a register	*MAR*, *R2*
Parentheses ( )	Denotes a part of a register	*R2*(0–7), *R2*(*L*)
Arrow ←	Denotes transfer of information	*R2* ← *R1*
Comma ,	Separates two microoperations	*R2* ← *R1*, *R1* ← *R2*

Bus system

- Provide path to transfer the information b/w the register in a multiple-register configuration.
- It consist of: Common line (one for each bit),

  Control signal(to select the register)

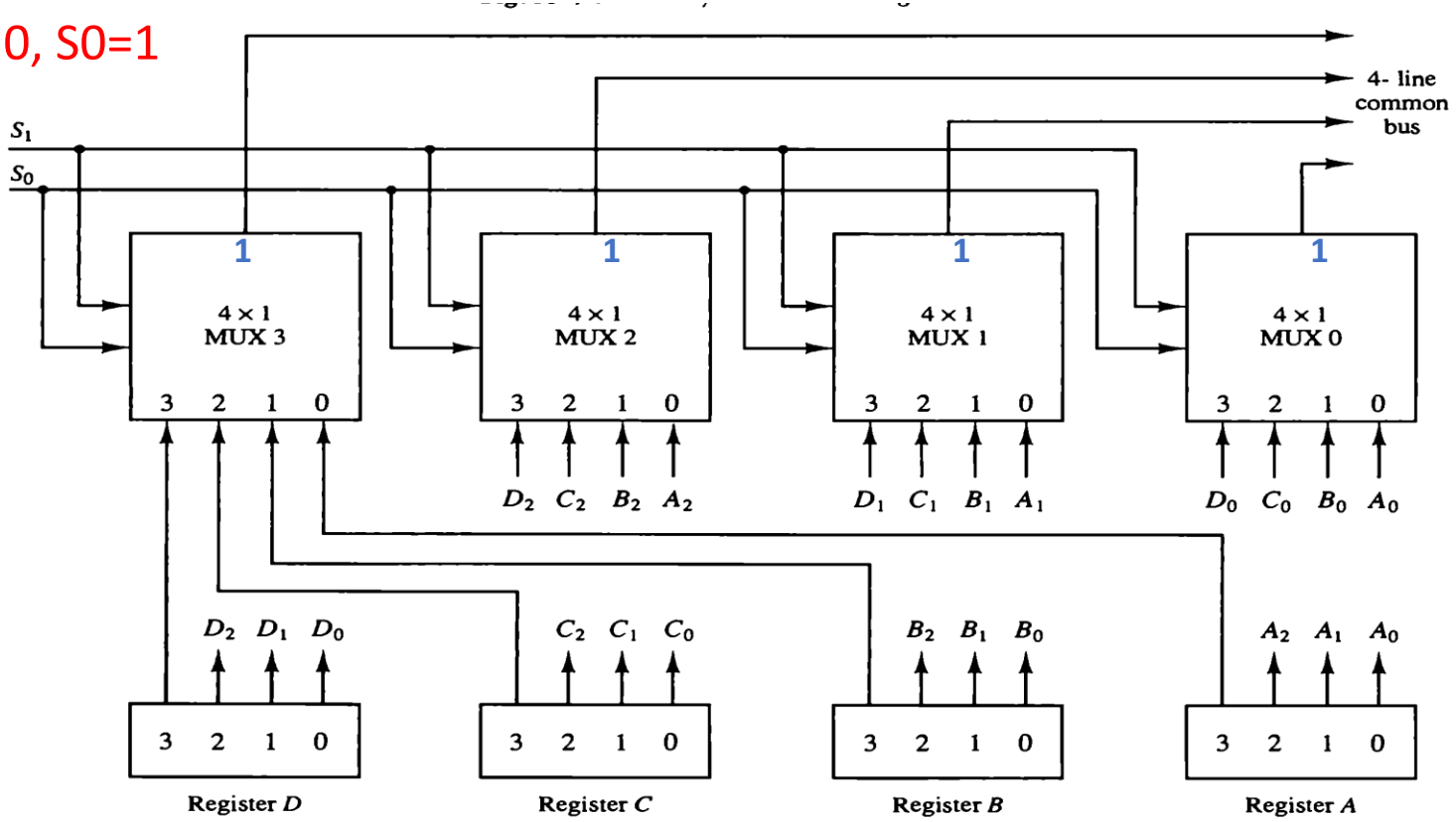
Two ways of constructing a common bus system:

(A) Multiplexer, and

(B) Buffer

Bus system using MUX

S1= 0, S0=1



B3 B2 B1 B0

TABLE 4-2 Function Table for Bus of Fig. 4-3

S_1	S_0	Register selected
0	0	A
0	1	B
1	0	C
1	1	D

For, K no. register n bit each,
Mux= n , (K x 1) size each.
Select line = $\log_2 K$
Eg: 8 Register 16 bit each,
MUX= 16, (8x1) , 3 select line.

- The symbolic statement for a bus transfer can be written as

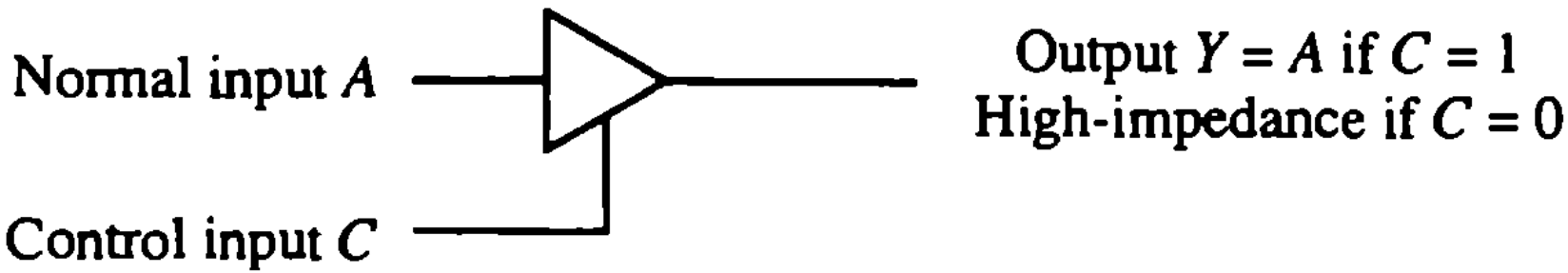
BUS←C, R1←BUS.

Three state Bus buffer

- 3 state: 2 state (logic 0 and logic 1)

  3rd state (high impedance state, i.e. open ckt (not connected)

Figure 4-4 Graphic symbols for three-state buffer.



Three state buffer

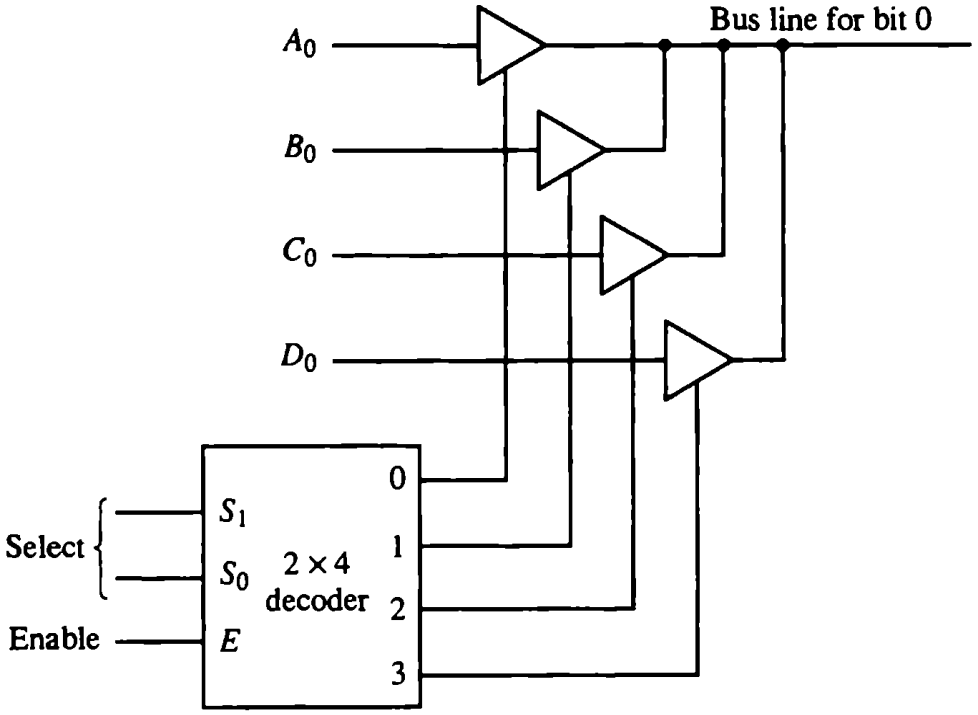


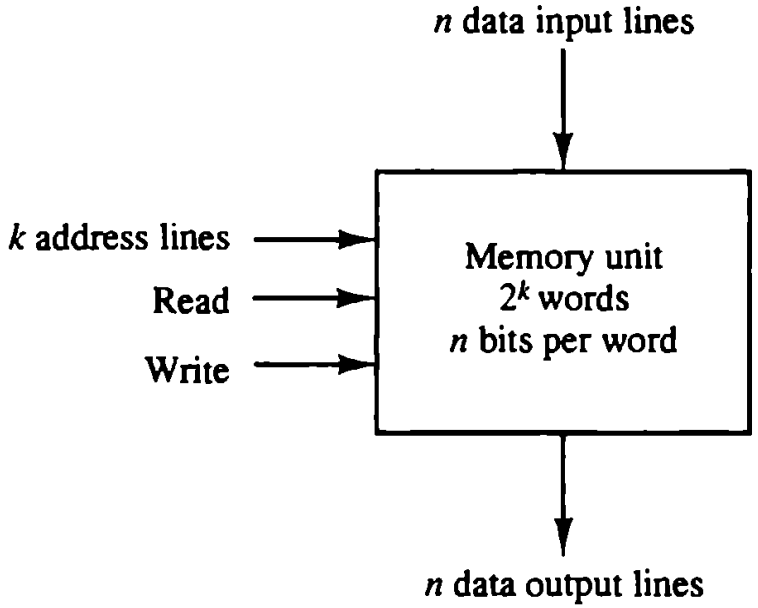
Figure 4-5 Bus line with three state-buffers.

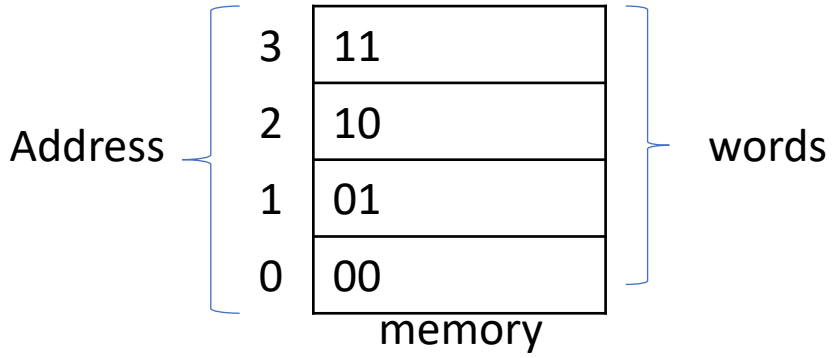
En	s1	s0	**A0**	**B0**	**C0**	**D0**
1	0	0	1	0	0	0
1	0	1	0	1	0	0
1	1	0	0	0	1	0
1	1	1	0	0	0	1
0	x	x	0	0	0	0

For K register , n bit each,
n ckt with K buffer in each is required.

Memory Unit

- A memory unit is a collection of storage cells together with associated ckt needed to transfer information in and out of storage.
- Group of bits = words.





Read : DR ← M[AR],
Write: M[AR]← R1.

Arithmetic micro-operation

TABLE 4-3 Arithmetic Microoperations

Symbolic designation	Description
$R3 \leftarrow R1 + R2$	Contents of $R1$ plus $R2$ transferred to $R3$
$R3 \leftarrow R1 - R2$	Contents of $R1$ minus $R2$ transferred to $R3$
$R2 \leftarrow \overline{R2}$	Complement the contents of $R2$ (1's complement)
$R2 \leftarrow \overline{R2} + 1$	2's complement the contents of $R2$ (negate)
$R3 \leftarrow R1 + \overline{R2} + 1$	$R1$ plus the 2's complement of $R2$ (subtraction)
$R1 \leftarrow R1 + 1$	Increment the contents of $R1$ by one
$R1 \leftarrow R1 - 1$	Decrement the contents of $R1$ by one

Binary Adder

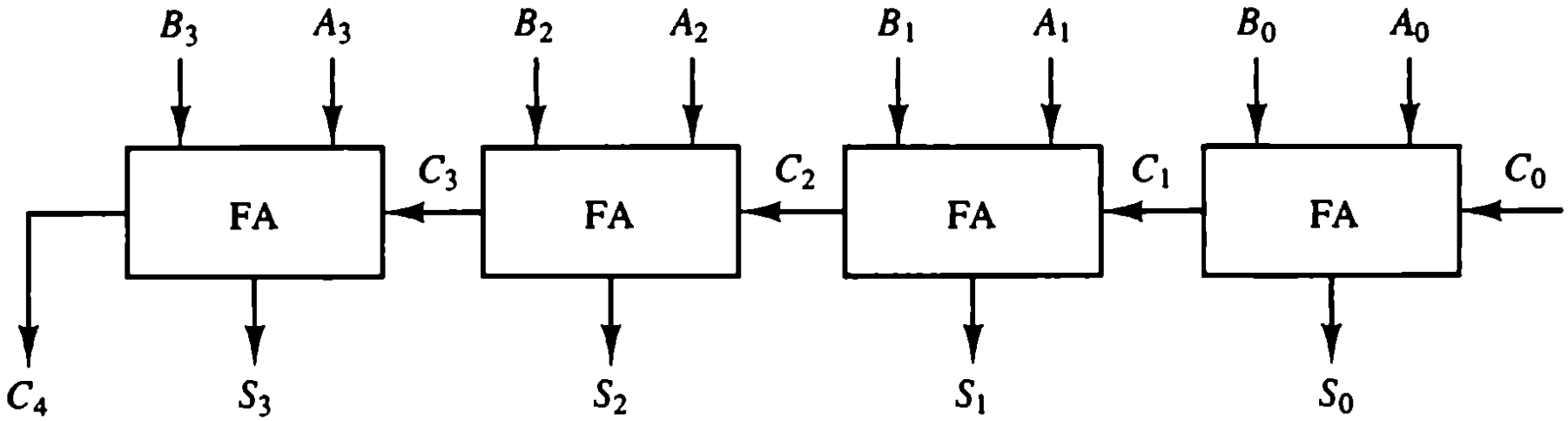


Figure 4-6 4-bit binary adder.

	3	**2**	**1**	**0**
Carry	1	1		
B	0	1	1	0
A +	0	1	1	1
S	1	1	0	1

Binary Adder- Subtractor

- M=0: circuit work as adder
- M=1: circuit work as subtractor
- A-B= A+B'+1; -B can be written as in 2's complement (B'+1)

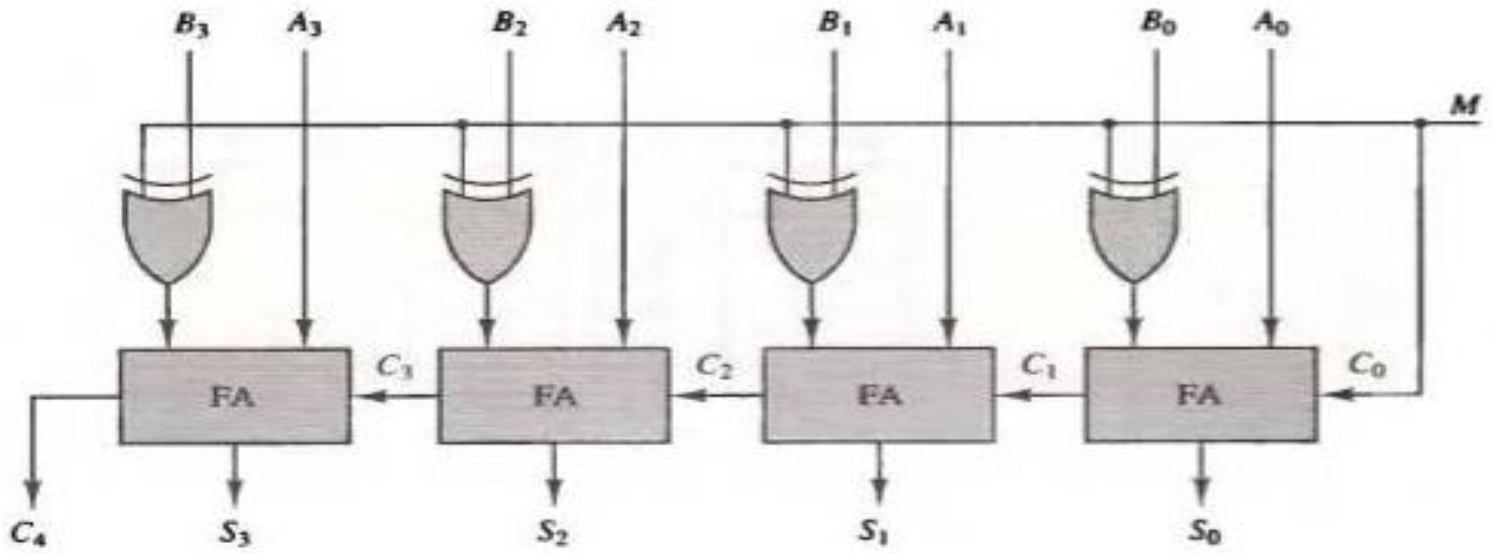


Figure 4-7 4-bit adder-subtractor.

Binary Increment

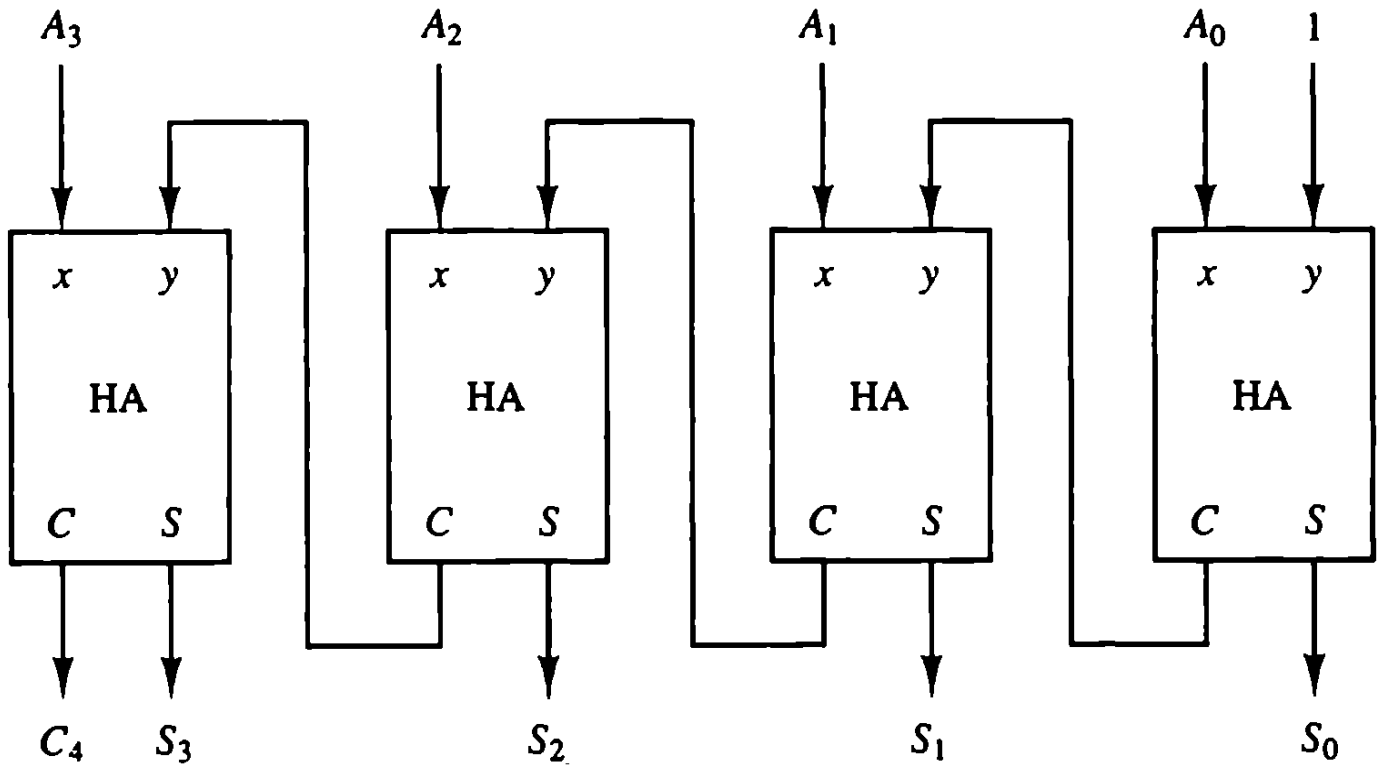


Figure 4-8 4-bit binary incrementer.

Arithmetic Circuit

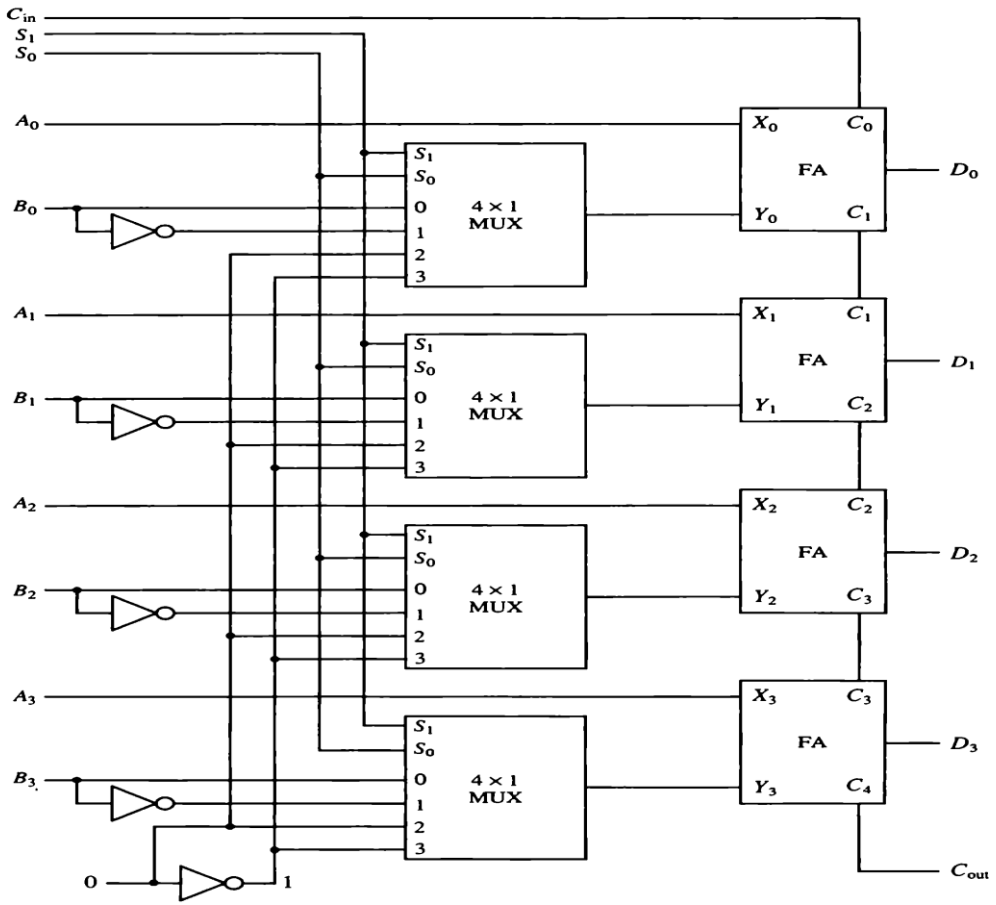


Figure 4-9 4-bit arithmetic circuit.

TABLE 4-4 Arithmetic Circuit Function Table

Select			Input	Output	
S_1	S_0	C_{in}	Y	$D = A + Y + C_{in}$	Microoperation
0	0	0	B	$D = A + B$	Add
0	0	1	B	$D = A + B + 1$	Add with carry
0	1	0	$\overline{B}$	$D = A + \overline{B}$	Subtract with borrow
0	1	1	$\overline{B}$	$D = A + \overline{B} + 1$	Subtract
1	0	0	0	$D = A$	Transfer A
1	0	1	0	$D = A + 1$	Increment A
1	1	0	1	$D = A - 1$	Decrement A
1	1	1	1	$D = A$	Transfer A

Logical micro-operation

$\vee$ → OR; $\wedge$→ AND;

$$P + Q: \quad R1 \leftarrow R2 + R3, \quad R4 \leftarrow R5 \vee R6$$

+ in micro-operation represent summation, whereas , in Control or Boolean function it is denoted as OR operation.

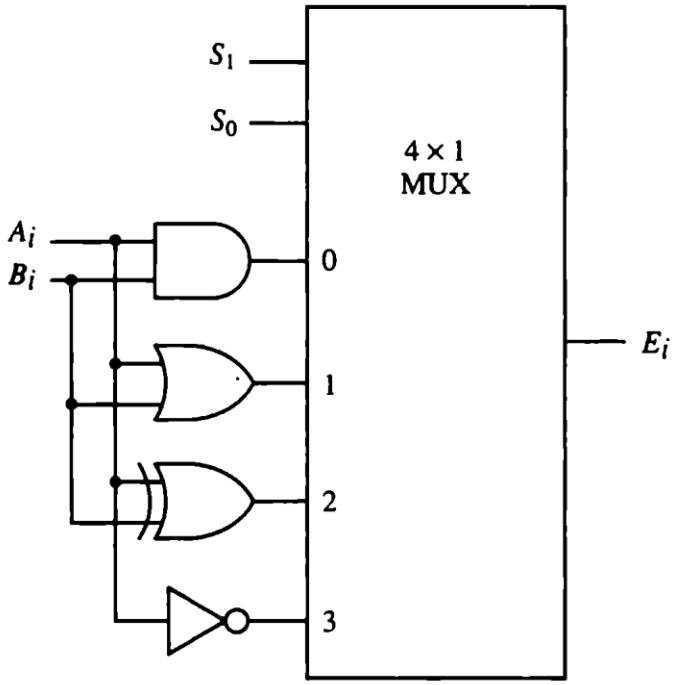
List of micro-operation using 2 binary variable.

TABLE 4-6 Sixteen Logic Microoperations

Boolean function	Microoperation	Name
$F_0 = 0$	$F \leftarrow 0$	Clear
$F_1 = xy$	$F \leftarrow A \wedge B$	AND
$F_2 = xy'$	$F \leftarrow A \wedge \overline{B}$	
$F_3 = x$	$F \leftarrow A$	Transfer A
$F_4 = x'y$	$F \leftarrow \overline{A} \wedge B$	
$F_5 = y$	$F \leftarrow B$	Transfer B
$F_6 = x \oplus y$	$F \leftarrow A \oplus B$	Exclusive-OR
$F_7 = x + y$	$F \leftarrow A \vee B$	OR
$F_8 = (x + y)'$	$F \leftarrow \overline{A \vee B}$	NOR
$F_9 = (x \oplus y)'$	$F \leftarrow \overline{A \oplus B}$	Exclusive-NOR
$F_{10} = y'$	$F \leftarrow \overline{B}$	Complement B
$F_{11} = x + y'$	$F \leftarrow A \vee \overline{B}$	
$F_{12} = x'$	$F \leftarrow \overline{A}$	Complement A
$F_{13} = x' + y$	$F \leftarrow \overline{A} \vee B$	
$F_{14} = (xy)'$	$F \leftarrow \overline{A \wedge B}$	NAND
$F_{15} = 1$	$F \leftarrow$ all 1's	Set to all 1's

H/w implementation of logical micro-operations.

Figure 4-10 One stage of logic circuit.



(a) Logic diagram

S_1	S_0	Output	Operation
0	0	$E = A \wedge B$	AND
0	1	$E = A \vee B$	OR
1	0	$E = A \oplus B$	XOR
1	1	$E = \overline{A}$	Complement

(b) Function table

- Selective set: Set bit of A=1, where B=1 else no change.

OR operation

1010	*A* before
1100	*B* (logic operand)
1110	*A* after

- Selective complement : A= A′, where B=1 else no change

X-OR operation

1010	*A* before
1100	*B* (logic operand)
0110	*A* after

- Selective Clear: A= 0; where B= 1 else no change.

A← A AND B’

1010	*A* before
1100	*B* (logic operand)
0010	*A* after

- Mask Operation: A=0; where B=0 else no change.

AND operation

1010	*A* before
1100	*B* (logic operand)
1000	*A* after masking

- Insert operation: mask + OR operation

0110 1010	*A* before
0000 1111	*B* (mask)
0000 1010	*A* after masking

0000 1010	*A* before
1001 0000	*B* (insert)
1001 1010	*A* after insertion

- Clear operation: Compared A and B, X-OR operation.

1010	*A*
1010	*B*
0000	$A \leftarrow A \oplus B$

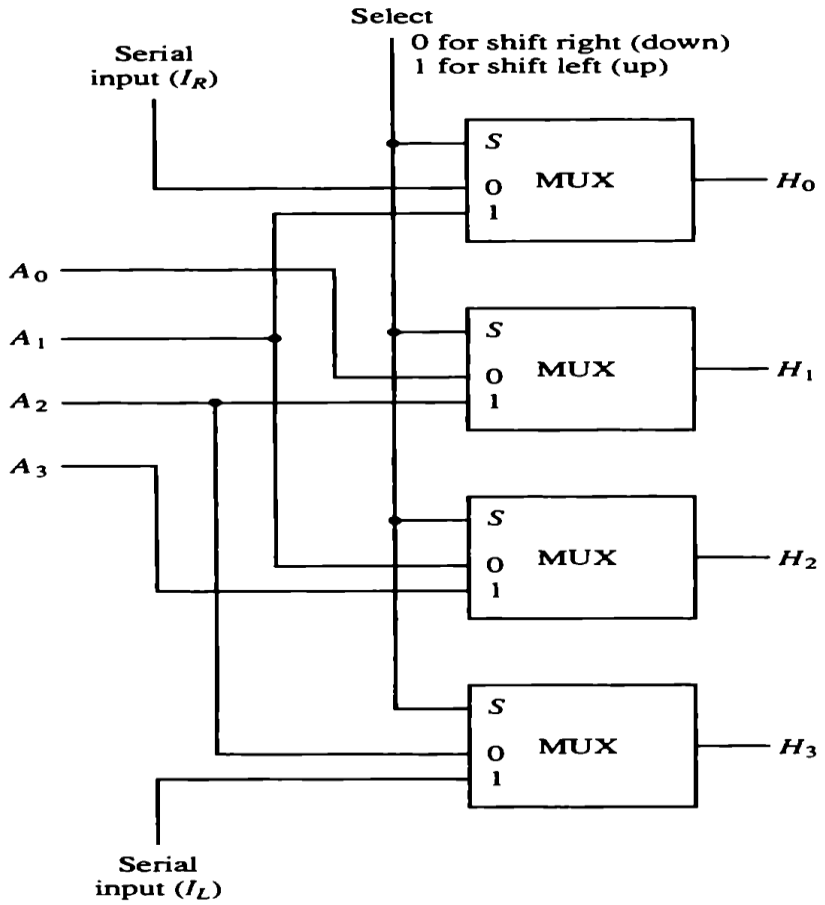
Shift-micro-operation

- Logical shift: used to shift the content left or right.
- Circular Shift: Shift the bit of register around the two end.
- Arithmetic shift: Left shift : multiply by 2 and right shift divide by 2.

TABLE 4-7 Shift Microoperations

Symbolic designation	Description
$R \leftarrow \text{shl } R$	Shift-left register R
$R \leftarrow \text{shr } R$	Shift-right register R
$R \leftarrow \text{cil } R$	Circular shift-left register R
$R \leftarrow \text{cir } R$	Circular shift-right register R
$R \leftarrow \text{ashl } R$	Arithmetic shift-left R
$R \leftarrow \text{ashr } R$	Arithmetic shift-right R

Shift micro-operation



Function table

Select	Output			
S	H_0	H_1	H_2	H_3
0	I_R	A_0	A_1	A_2
1	A_1	A_2	A_3	I_L

Figure 4-12 4-bit combinational circuit shifter.

Arithmetic logic shift unit

Figure 4-13 One stage of arithmetic logic shift unit.

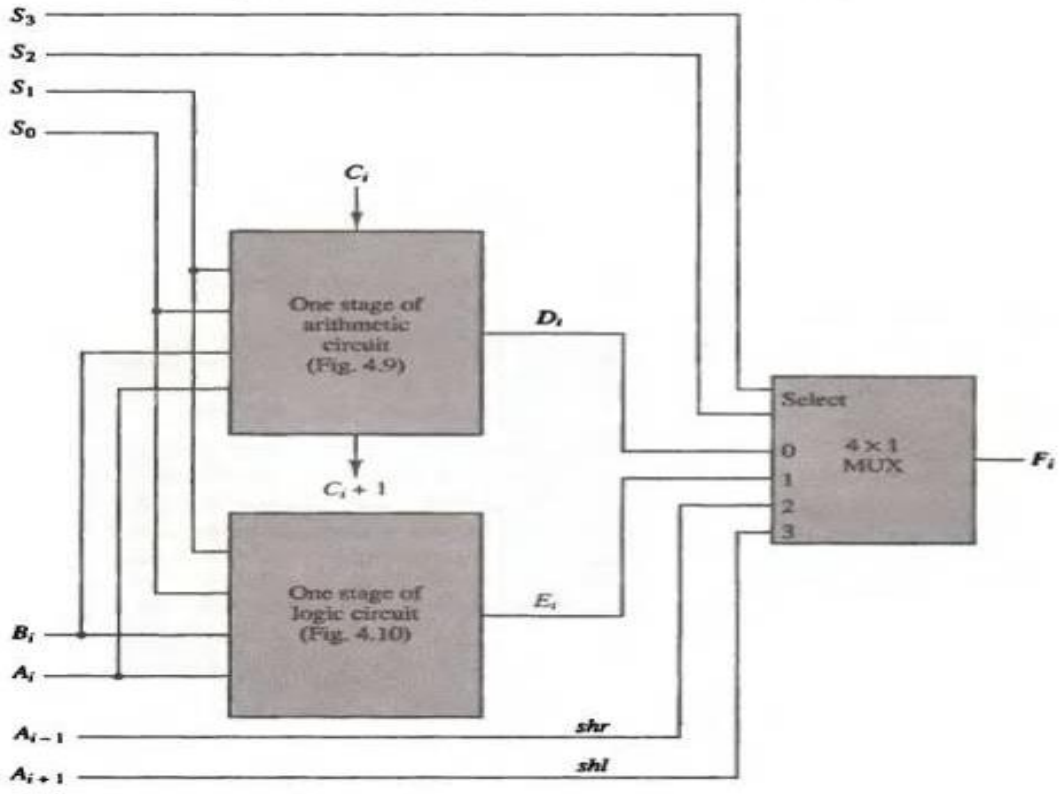


TABLE 4-8 Function Table for Arithmetic Logic Shift Unit

Operation select						
S_3	S_2	S_1	S_0	C_{in}	Operation	Function
0	0	0	0	0	$F = A$	Transfer A
0	0	0	0	1	$F = A + 1$	Increment A
0	0	0	1	0	$F = A + B$	Addition
0	0	0	1	1	$F = A + B + 1$	Add with carry
0	0	1	0	0	$F = A + \overline{B}$	Subtract with borrow
0	0	1	0	1	$F = A + \overline{B} + 1$	Subtraction
0	0	1	1	0	$F = A - 1$	Decrement A
0	0	1	1	1	$F = A$	Transfer A
0	1	0	0	×	$F = A \wedge B$	AND
0	1	0	1	×	$F = A \vee B$	OR
0	1	1	0	×	$F = A \oplus B$	XOR
0	1	1	1	×	$F = \overline{A}$	Complement A
1	0	×	×	×	$F =$ shr A	Shift right A into F
1	1	×	×	×	$F =$ shl A	Shift left A into F