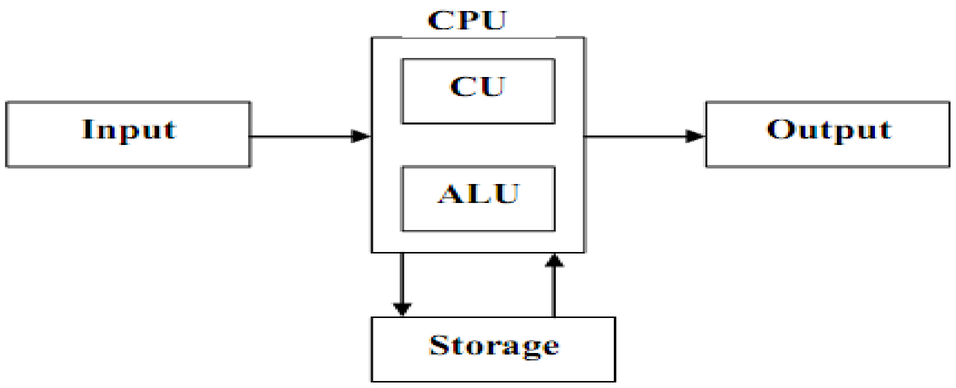
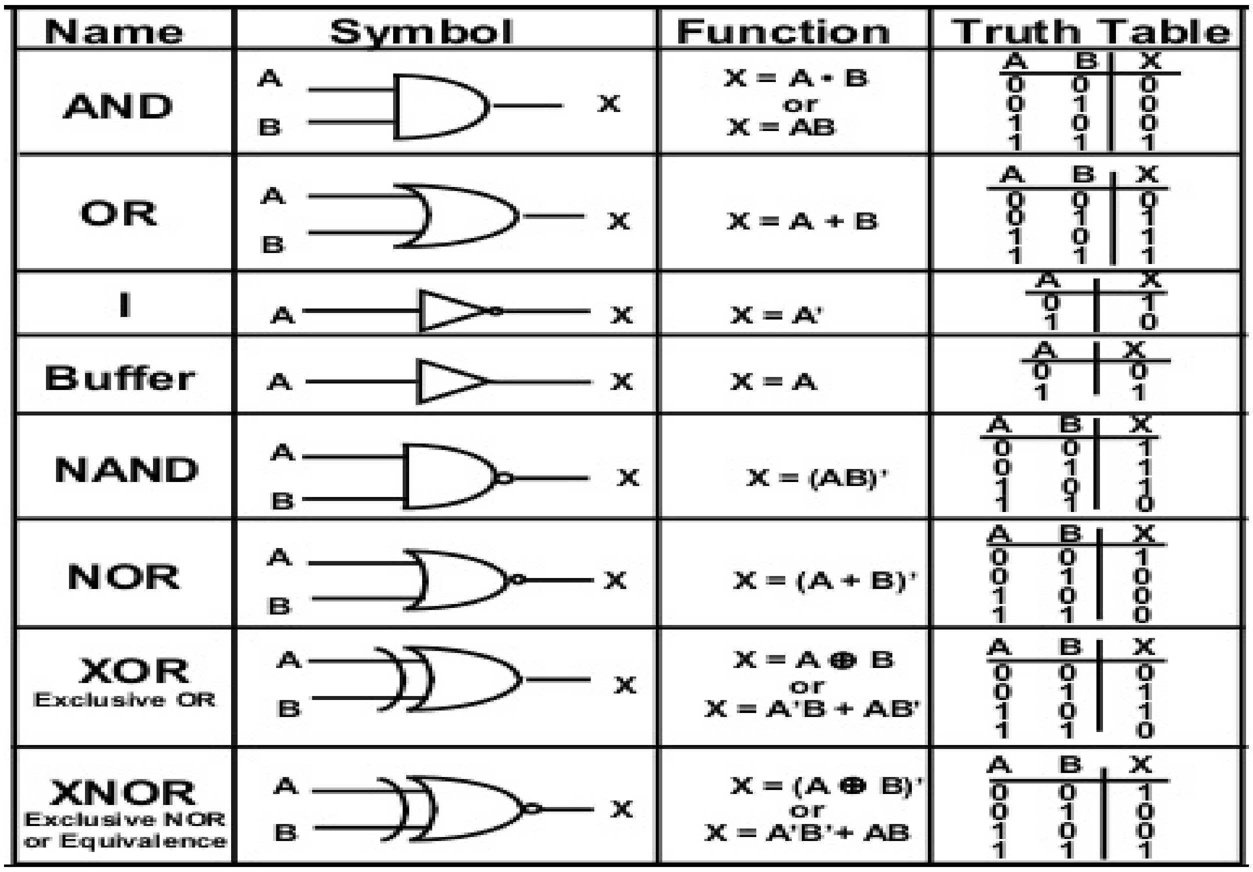
1. Draw and explain neat sketch block diagram of Digital Computer

- A digital computer consists of ALU (Arithmetic and Logical Unit), control unit, input unit, output unit and memory unit.
- **Input unit –** accept digital information from user with the help of input devices such as keyboard, mouse
- **ALU –** perform arithmetic operations like addition, subtraction, multiplication, division and logical operations as defined by a program
- **Control Unit-** controls the flow of information between the various units.
- **Output unit-** receives the result and printed on output device like printer, monitor.



2. Explain in detail logic gates in detail.

- Basic elements that design a digital system



Name	Symbol	Function	Truth Table
AND	A B X	X = A • B or X = AB	A B | X: 0 0 | 0; 0 1 | 0; 1 0 | 0; 1 1 | 1
OR	A B X	X = A + B	A B | X: 0 0 | 0; 0 1 | 1; 1 0 | 1; 1 1 | 1
I	A X	X = A'	A | X: 0 | 1; 1 | 0
Buffer	A X	X = A	A | X: 0 | 0; 1 | 1
NAND	A B X	X = (AB)'	A B | X: 0 0 | 1; 0 1 | 1; 1 0 | 1; 1 1 | 0
NOR	A B X	X = (A + B)'	A B | X: 0 0 | 1; 0 1 | 0; 1 0 | 0; 1 1 | 0
XOR Exclusive OR	A B X	X = A ⊕ B or X = A'B + AB'	A B | X: 0 0 | 0; 0 1 | 1; 1 0 | 1; 1 1 | 0
XNOR Exclusive NOR or Equivalence	A B X	X = (A ⊕ B)' or X = A'B'+ AB	A B | X: 0 0 | 1; 0 1 | 0; 1 0 | 0; 1 1 | 1

3. What is Flip-Flop? Explain types of Flip-Flops.

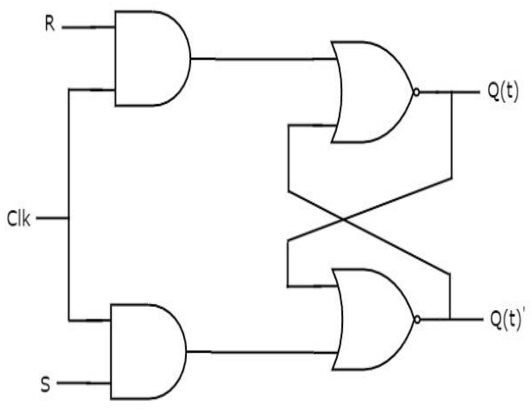
- A Flip Flop is a **sequential circuit** which is capable of **storing one bit of information**.
- It is also known as a basic digital memory circuit.
- It has two stable states: logic 1 and logic 0.
- Flip Flop can be designed using **NOR gates or NAND gates**.

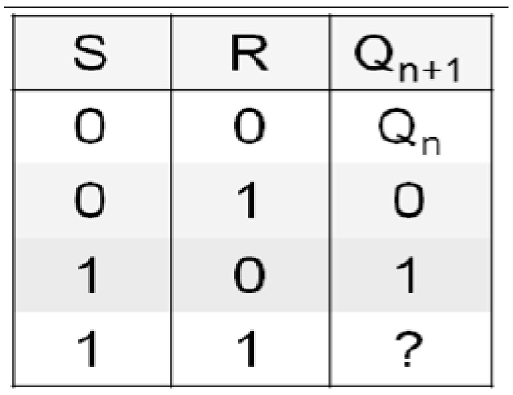
Applications of flip flop	**Types of Flip-Flops**
• Used as a memory element. • It can use as a shift register and counter. • Used as a delay element	• RS Flip Flop • D Flip Flop • T Flip Flop • JK Flip Flop • Master Slave JK Flip Flop

Types of Flip-Flops

i. **SR Flip-Flop**:

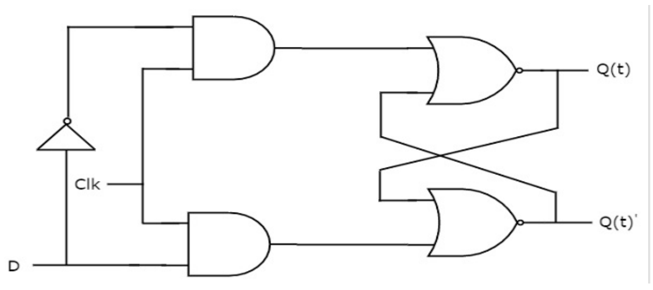
Truth table:

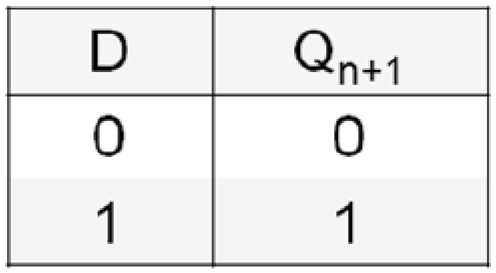




S	R	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	?

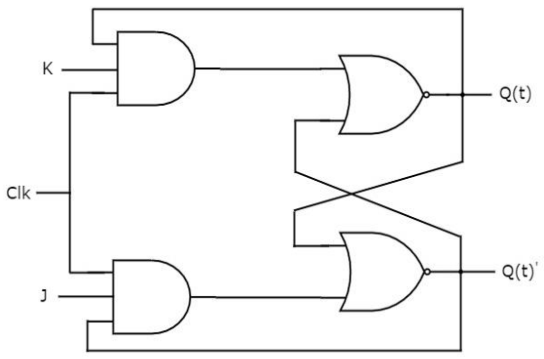
ii. **D Flip-Flop:**





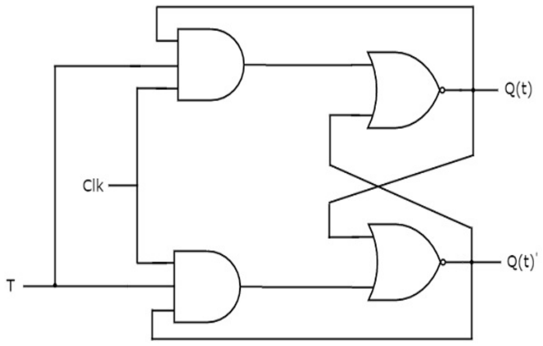
D	Q_{n+1}
0	0
1	1

iii. **JK Flip-Flop:**



J	K	Q_{n+1}
0	0	Q_n
0	1	0
1	0	1
1	1	Qn’

iv. **T Flip-Flop:**



T	Q_{t+1}
0	Q_t
1	$\overline{Q}_t$

4. Explain Shift registers.

- It is a group of flip-flops, which are used to hold (store) the binary data is known as **register**.
- If the register is capable of shifting bits either right hand side or left hand side is known as **shift register**. It is a chain of flip flop in which output of one register connected to input of another register.
- An ‘N’ bits shift register contains ‘N’ flip-flops.

Applications of shift register:

- Design special types of counter
- It is use for data storage, data transfer and arithmetic and logical operation.

Following are the four types of shift registers

- Serial In - Serial Out shift register
- Serial In - Parallel Out shift register
- Parallel In - Serial Out shift register
- Parallel In - Parallel Out shift register

5. Define the following term.

- **Micro operation, register transfer, registers transfer language.**

Micro operation

- The operations performed **on data stored in registers**, are called Microoperation. Example:- shift, load, clear, increment

Register Transfer

- The term "register transfer'' means the availability of **hardware logic circuits** that can perform a **microoperation** and **transfer the result** of the operation to the **same or another register**.

Register Transfer Language

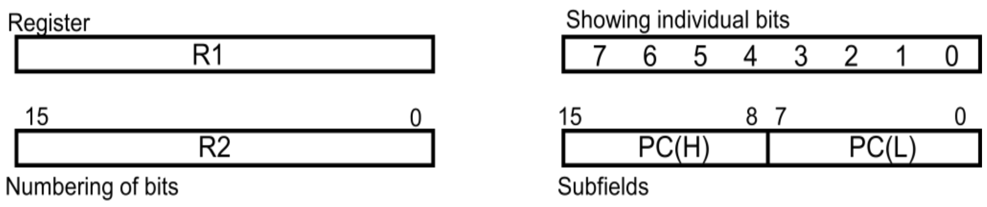
- **Symbolic notation used** to describe the **micro operation transfer** among the **register** is called register transfer language.

6. Explain in detail block diagram of register. OR Explain any three symbols used in register transfer language with example.

- The term register transfer means availability of hardware logic circuits that can perform micro operation and transfer the result of the operation to the same or another register.
- **A Register Transfer Language is** Symbolic notation used to describe the micro operation transfer among the registers is called register transfer language.
- Tool for describing internal structure of digital computer

Register Transfer

- Information transferred from one register to another register is represented in symbolic form like
  R2 ← R1
- The above operation copy content of R1 into R2 register.
- A register that hold memory address is called Memory Address register and denoted by **MAR**.
- Other notation for registers are **PC(program counter), IR(instruction Register) and R1(processor register)**



Block diagram of registers.

Control Function

- Normally we want that transfer occur only in predetermined control condition and that can be shown by if-then statement like **if (P=1) then (R2 ← R1)**. Here **P is a control signal** generated in the control section.
- **A control function** is a Boolean variable that is equal to 1 or 0.
- **The control function is shown as: P: R2 ← R1**
- The above function shows that transfer operation can be performed only if P=1. The control condition is terminated with a colon.
- Every statement written in a register transfer notation implies a hardware construction for implementing the transfer.
- Below figure shows the block diagram that represents the transfer from R1 to R2.

Transfer from R1 to R2 when P = 1

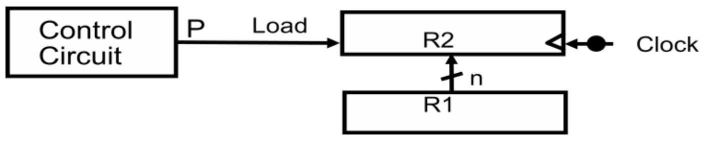


Figure: Register transfer operation

- The n outputs of register R1 are connected to the n inputs of register R2. The letter n will be used to indicate any number of bits for the register.

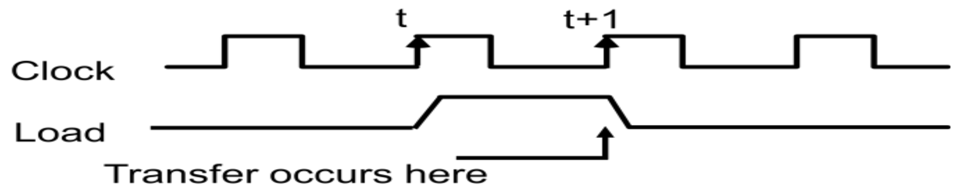


Figure: Timing diagram

- In the timing diagram, P is activated at the time t in the control section represent the rising edge of a clock pulse.
- At the time t+1 load input is activated and information transfer from register R1 into R2.
- P may go back to 0 at time t + 1; otherwise, the transfer will occur with every clock pulse transition while P remains active.
- The statement below denotes that exchanges the contents of two registers during one common clock pulse provided that T = 1.
- T: R2← R1, R1← R2

- The basic symbols of the register transfer notation are listed in Table below:

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 micro operations	R2← R1, R1← R2

7. Explain Bus Transfer and Memory Transfer in brief.

Bus Transfer

- A digital computer has many registers and paths, these registers and paths are used to transfer information from one register to another.
- A large number of wires required if separate lines are used between each register. For example: - n no of registers require n*(n-1) no of wires and that is impractical solution.
- A more efficient method for transferring information between registers in a multiple - register configuration is a **common bus system**.
- A bus structure contains a set of **common lines and a single line transfer one bit of information** at a time.
- **Control signals select a register whose binary information is transfer to the common bus**.
- There are two ways of constructing a common bus system
  i) **Multiplexers**.
  ii) **Three state buffer and decoder**

Memory Transfer

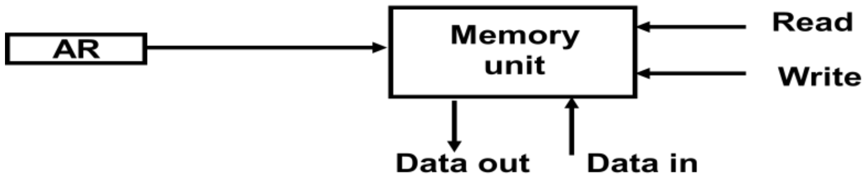
- A memory word represents by the letter M.
- It is necessary to specify the **address of M (Memory)** when writing memory transfer operations.
- A memory unit receives the address from a register called the **address register**, symbolized by **AR**.
- The data are transferred to other register called the **data register, symbolized by DR.**
- There are two memory transfer operations: 1. **Read Operation and 2. Write Operation**
  - **Read Operation:** Information transfer **from a memory word** to the outside environment is called a read operation.
    - The read operation can be stated as follows:

      Read: DR ←M[AR]

    - Above operation transfer information into DR from the memory word M selected by the address in AR.
  - **Write Operation:** The information **to be stored into the memory** is called a write operation.
    - The write operation transfers the content of a data register to a memory word M selected by the address. Assume that the input data are in register R1 and the address is in AR.
    - Write operation can be stated symbolically as follows:

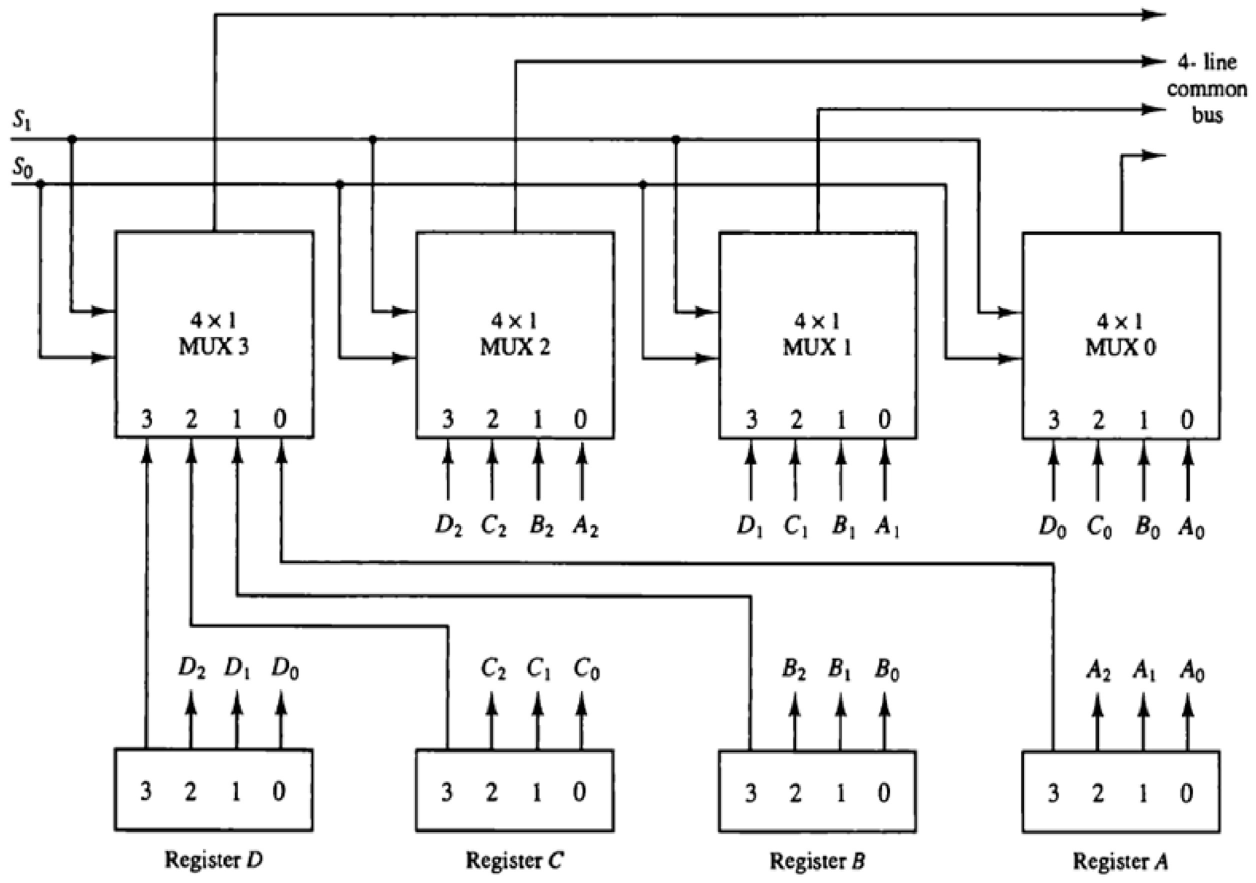
      Write: M[AR] ←R1

    This causes a transfer of information from R1 into memory word M selected by address AR.



8. Explain register bus system data transfer OR Explain Bus system Data transfer for four Registers using Multiplexers. OR Explain BUS Transfer using Multiplexer (4 Register).

- **Multiplexer** is used to design a common bus system.
- The multiplexer select the source **register whose binary information** is then placed on the bus.
- The construction of a bus system for four registers is shown in figure below.
- Each register has four bits, **numbered 0 through 3**.
- The bus consists of **four 4 x 1 multiplexers** each having **four data inputs, 0 through 3**, and **two selection inputs, S1 and S0**.
- The diagram shows that the bits in the same significant position in each register are connected to the data inputs of one multiplexer to form one line of the bus.
- The two selection **lines S1 and S0** are connected to the **selection inputs of all four multiplexers**.
- The selection lines choose **the four bits of one register** and transfer them into the **four- line common bus.**



Truth Table for source registers selection

S0	S1	Register selected
0	0	A
0	1	B
1	0	C
1	1	D

- **When S1S0 = 00,** then 0 data inputs of all four multiplexers are selected and applied to the outputs that form the bus.
- Here, the bus lines receive the **content of register A** with the help of multiplexers.
- Similarly, **register B is selected if S1S0 = 01,** and so on.
- Table shows the register that is selected by the bus for each of the four possible binary values of the selection lines.
- In general, a bus system will multiplex k registers of n bits, each to produce an n-line common bus.
- The number of multiplexers needed to construct the bus is equal to n, and n is the number of bits in each register.
- The size of each multiplexer must be k x 1 since it multiplexes k data lines.
- For example, a common bus for eight registers of 16 bits each requires 16 multiplexers, one for each line in the bus. Each multiplexer must have eight data input lines and three selection lines to multiplex one significant bit in the eight registers.

9. Explain three-state bus buffer. OR Explain the operation of three state bus buffers and show its use in design of common bus.

- A bus system can be constructed with **three-state gates** instead of multiplexers.
- A three-state gate is a digital circuit that has three states.
  State 1: Signal equivalent to Logic 1
  State 2: Signal equivalent to Logic 0
  State 3: High Impedance State (behaves as open circuit)
- The high-impedance state behaves like an **open circuit**, which means that the **output is disconnected and does not have logic significance.**
- The graphic symbol of a three-state buffer gate is shown in figure below:

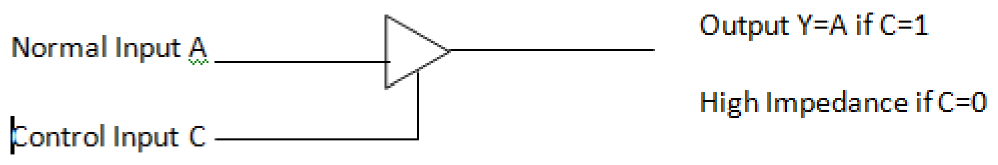
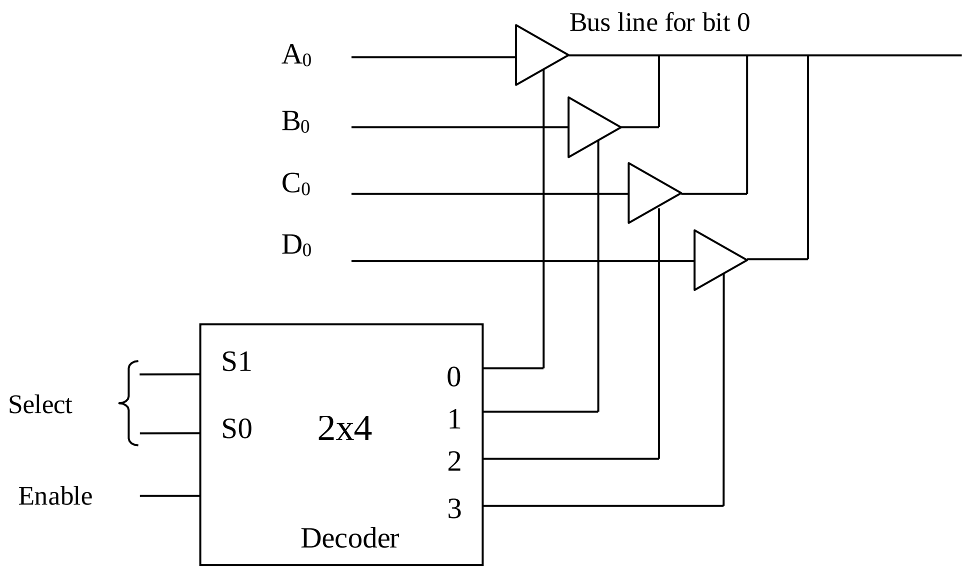


Figure: Graphic symbols for three-state buffer

- When the control input C is equal to 1, the output is enabled and the gate behaves like buffer and the output equal to the normal input.
- When the control input C is 0, the output is disabled and the gate goes to a high- impedance state, regardless of the value in the normal input.
- The construction of a bus system with three-state buffers and decoder is demonstrated below:



Bus line with three state-buffers

- The outputs of four buffers are connected together to form a **single bus line**.
- The control inputs to the buffers determine which of the four normal inputs will communicate with the bus line.
- No more than one buffer may be in the active state at any given time.

- The connected buffers must be controlled so that only **one three-state buffer** has access to the bus line while all other buffers are maintained in a high impedance state.
- One way to ensure that **no more than one control input is active** at any given time is to use a **decoder**
- When the enable input of the **decoder is 0**, all of its **four outputs are 0**, and all **four buffers are disabled.**
- When the **enable input is active**, one of the three-state buffers will be active, depending on the binary value in the select inputs of the decoder.
- To construct a common bus for **four registers of n bits**, we need n circuits with four buffers in each as shown in figure: Bus line with three state-buffers,
- Only one decoder is necessary to select between the four registers.

Truth Table for source registers selection

S0	S1	Register selected
0	0	A
0	1	B
1	0	C
1	1	D

10. Explain Arithmetic Micro-operation.

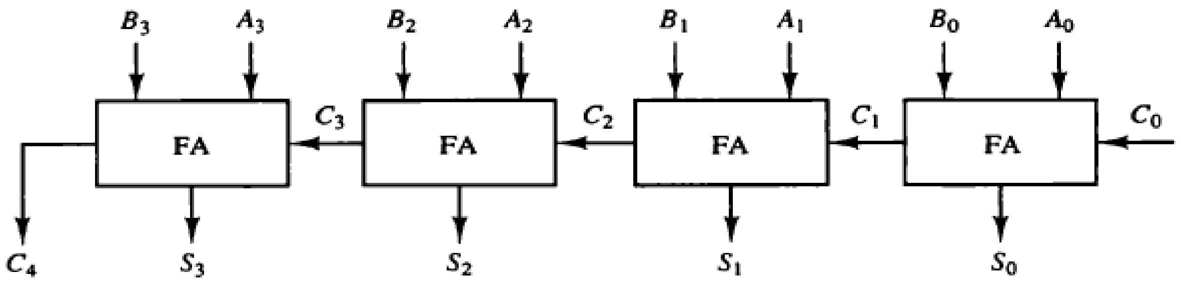
- The basic arithmetic micro-operations are:
  1. Addition
  2. Subtraction
  3. Increment
  4. Decrement
  5. Shift
- The additional arithmetic micro operations are:
  1. Add with carry
  2. Subtract with borrow
- Summary of Typical Arithmetic Micro-Operations:

R3 ←R1 + R2	Contents of R1 plus R2 transferred to R3
R3 ← R1 - R2	Contents of R1 minus R2 transferred to R3
R2 ← R2’	Complement the contents of R2
R2 ← R2’+ 1	2's complement the contents of R2(negate)
R3 ←R1 + R2’+1	subtraction
R1 ←R1 + 1	Increment
R1 ←R1 – 1	Decrement

11. Draw and explain 4 bit binary adder circuit diagram.

- A binary parallel adder is digital circuit that produces the **arithmetic sum of two binary numbers in parallel.**
- It contains **full-adders connected** in cascade, with output carry from one full-adder connected to input carry of the next full adder.
- Following figure shows the interconnections of **four full adders to provide a four bit adder**.

- **Bits of register A and B** are input bits applied to each Full adder. The input carry to the binary adder is **C0 and the output carry is C4**.
- The carries are connected in a chain through the full adder.



- The S outputs of the full adder generate the required sum bits. Below figure is for n- bit binary adder.
- The micro operation can be written in RTL as P: A ← A + B where A & B are two registers.

12. Explain Binary Adder-Subtractor in detail.

- The subtraction of binary numbers can be done with the help of complements.
- The subtraction A - B can be done by taking the 2's complement of B and adding it to A.
- The 2's complement can be obtained by taking the 1's complement of a number and adding one to to it. The 1's complement can be obtained by converting 1 to 0 and 0 to 1.
- The addition and subtraction operations can be combined into one common circuit by including an exclusive-OR gate with each full-adder.
- The mode input M controls the operation.

  When **M = 0** the circuit is an **Adder**

  When **M = 1** the circuit becomes a **Subtractor**
- Each exclusive-OR gate receives input M and one of the inputs of B.

When M = 0,

We have $C_0=0$

$B \oplus 0 = B$

- The full-adders receive the value of B, the input carry is 0, and the circuit performs A + B.

When M = 1,

We have $C_0=1$

$B \oplus 1 = B`$

- The B inputs are all complemented and 1 is added through the input carry. The circuit performs the operation A plus the 2's complement of B.

  A + 2's compliment of B
- A 4-bit adder-subtractor circuit is shown as follows:

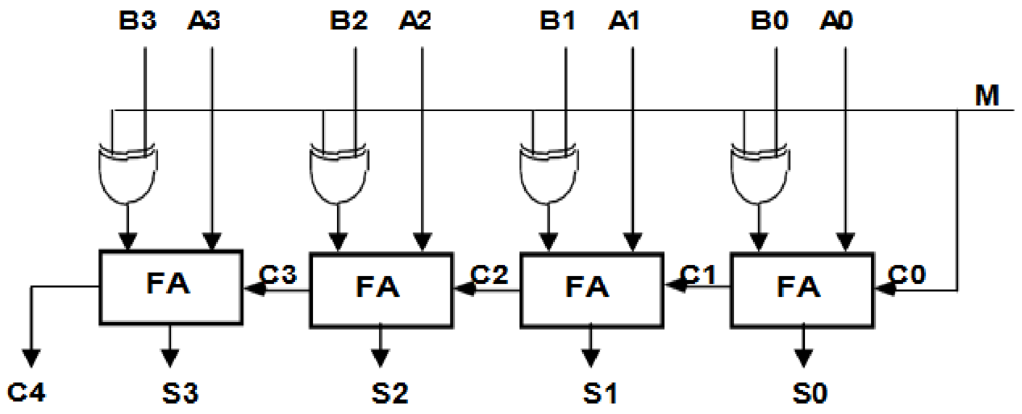


Figure: 4-bit Adder-Subtractor

- For unsigned numbers,
  If A>=B, then A-B If A<B, then B-A
- For signed numbers,
  Result is A-B, provided that there is no overflow.

13. Explain Binary Incrementer.

- The increment micro operation adds one to a number in a register.
- For example, if a 4-bit register has a binary value 0110, it will go to 0111 after it is incremented.

```
0 1 1 0
+     1
--------
0 1 1 1
```

- The diagram of a 4-bit combinational incrementer circuit is shown below. There are four Half adder used in the circuit. Each HA has two inputs. One input to First HA is logic-1 and the other input is bit of the number.
- The output carry from one half-adder is connected to one of the inputs of the next- higher-order half-adder.
- The circuit receives the four bits from A0 through A3, adds one to it, and generates the incremented output in S0 through S3.
- The output carry C4 will be 1 only after incrementing binary 1111. This also causes outputs S0 through S3 to go to 0.

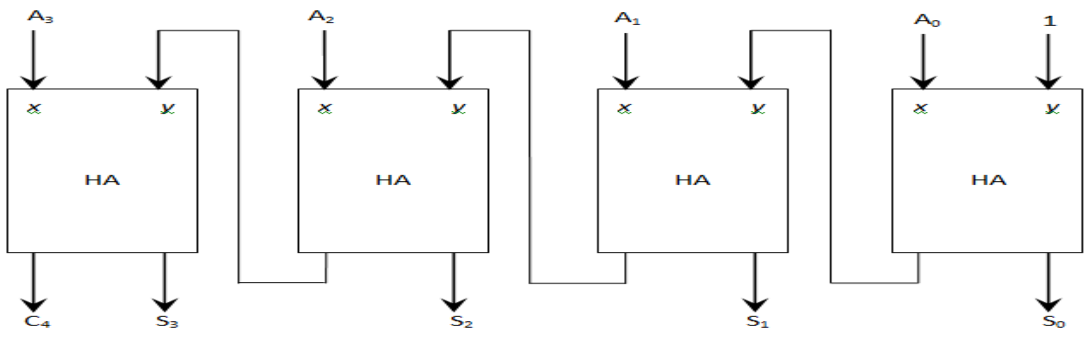


Figure: 4-bit binary incrementer

14. Draw block diagram of 4-bit arithmetic circuit and explain it in detail.

- The arithmetic micro operations can be implemented in one composite arithmetic circuit.
- The basic component of an arithmetic circuit is the parallel adder.
- Hardware implementation consists of:

1) Circuit contains the four 4-bit adder and four multiplexers for choosing different operations.
2) There are two 4-bit inputs A and B
3) The four inputs from A go directly to the X inputs of the binary adder. Each of the four inputs from B is connected to the data inputs of the multiplexers. The multiplexer's data inputs also receive the complement of B.
4) The other two data inputs are connected to logic-0 and logic-1.
5) The four multiplexers are controlled by two selection inputs, S1 and S0.
6) The input carry Cin goes to the carry input of the FA in the least significant position. The other carries are connected from one stage to the next.
7) 4-bit output D0…D3

- The diagram of a 4-bit arithmetic circuit is shown below figure :

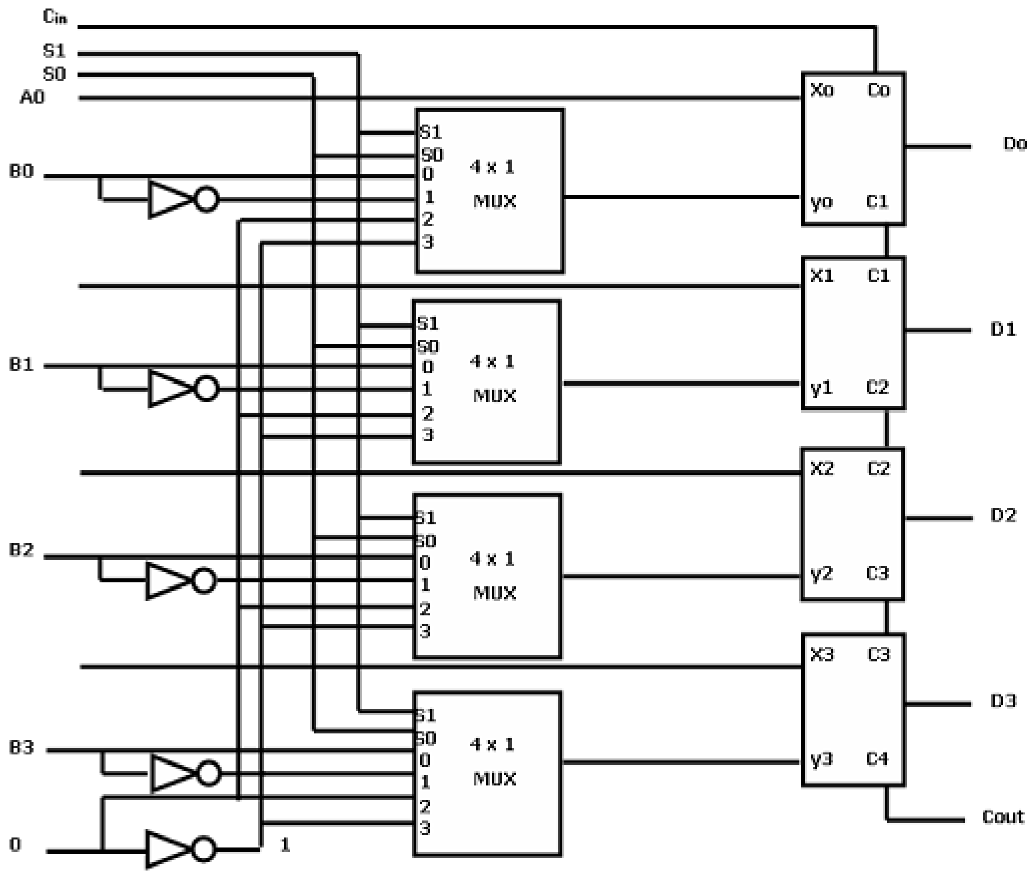


Figure: 4-bit arithmetic circuit

- The output of binary adder is calculated from arithmetic sum.

$$D=A+Y+C_{in}$$

Select			**Input**	**Output**	**Microoperation**
S_1	**S_0**	**C_{in}**	**Y**	**D = A + Y + Cin**	
0	0	0	B	D = A + B	Add
0	0	1	B	D = A + B + 1	Add with Carry
0	1	0	B′	D = A + B′	Subtract with Borrow
0	1	1	B′	D = A + 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

TABLE: Arithmetic Circuit Function Table

- When S1 S0= 0 0

If Cin=0, D=A+B;	Add
If Cin=1, D=A+B+1;	Add with carry

- When S1 S0= 0 1

If Cin=0, D=A+$\overline{B}$;	Subtract with borrow
If Cin=1, D=A+$\overline{B}$+1;	A+2's compliment of B i.e. A-B

- When S1 S0= 1 0
  Input B is neglected and Y=> logic '0'
  D=A+0+ Cin

If Cin=0, D=A;	Transfer A
If Cin=1, D=A+1;	Increment A

- When S1 S0= 1 1
  Input B is neglected and Y=> logic '1'
  D=A-1+ Cin

If Cin=0, D=A-1;	Decrement A
If Cin=1, D=A;	Transfer A

Note that the micro-operation D = A is generated twice, so there are only seven distinct micro-operations in the arithmetic circuit.

15.Define Micro Operations & explain Logic Micro Operations.

- Micro Operation: - Micro operations are elementary operations performed on data stored in registers or in memory.
- There are four types of micro operations that are described below:

1. Transfer micro operations: This is an operation where data is transferred from one register to another. Data may also be transferred from a register to memory or from memory to a register.
2. Arithmetic micro operations: This is an operation that performs arithmetic on data in one or two registers.
3. Logic micro operations: This operation performs bit manipulation on data in one or two

registers. Example: - complement, AND, OR

4. Shift micro operations: This operation shifts data in a register. Example: - logical shift left

Logic micro operation:

- For example, the exclusive-OR micro-operation with the contents of two registers R1 and R2 is symbolized by the statement:
  `P: R1 ←R1 ⊕ R2`

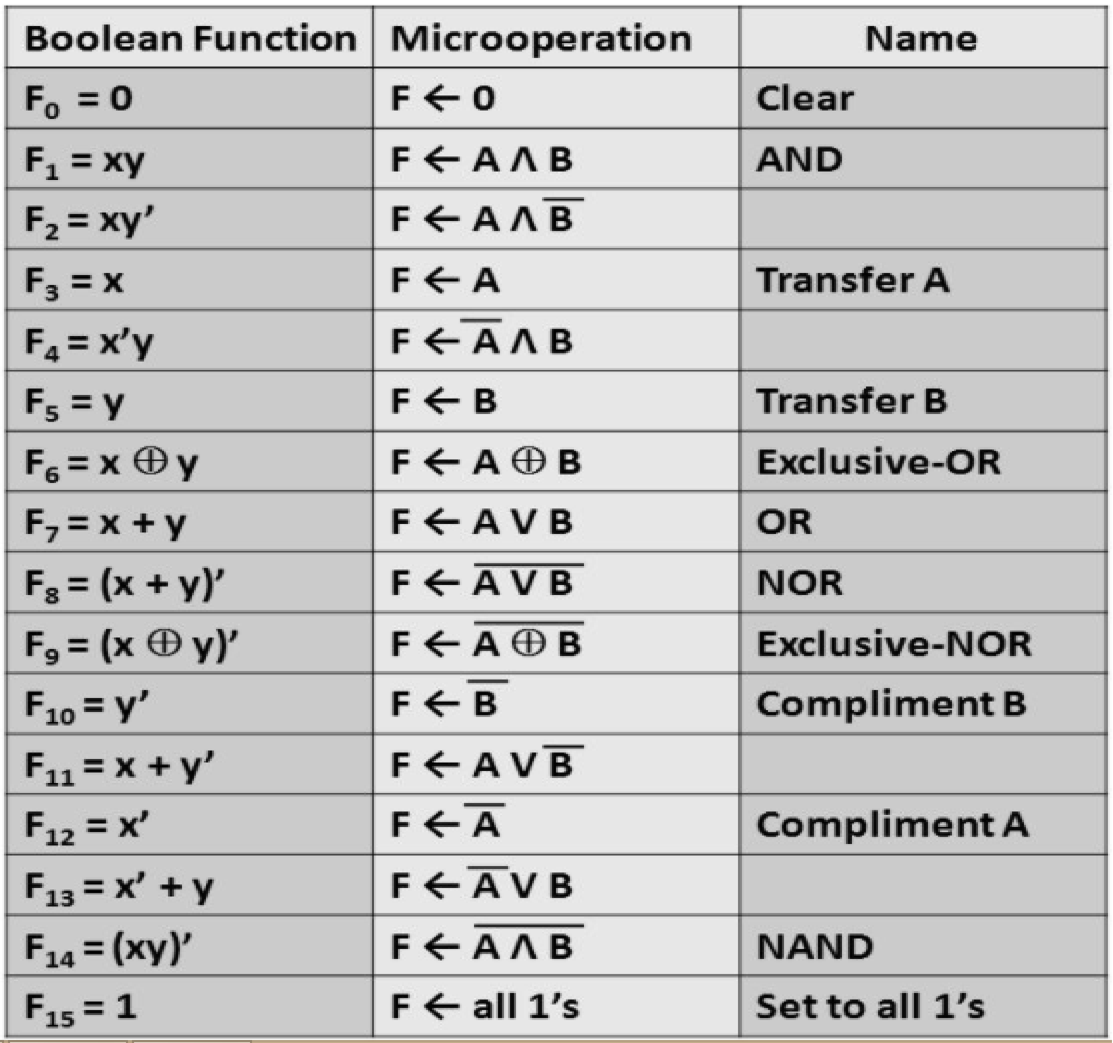
	1 0 1 0	Content of R1
$\oplus$	1 1 0 0	Content of R2
	0 1 1 0	Content of R1 after P = 1

List of Logic Micro operations

- There are 16 different logic operations that can be performed with two binary variables.

A	**B**	**F_0** Clear	**F_1** A Bitwise And B (AB)	**F_2** AB’	**F_3** transfer A	**F_4** A`^B	**F_5** transfer B	**F_6** A ex-or B	**F_7** A or B	**F_8** A nor B	**F_9** A ex-nor B	**F_{10}** Compl- B	**F_{11}** A or B`	**F_{12}** Compl- A	**F_{13}** A`or B	**F_{14}** A nand B	**F_{15}** Set 1
0	**0**	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	**1**	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	**0**	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	**1**	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

TABLE: Truth Tables for 16 Functions of Two Variables



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
$F_{10} = y'$	$F \leftarrow \overline{B}$	Compliment B
$F_{11} = x + y'$	$F \leftarrow A \vee \overline{B}$	
$F_{12} = x'$	$F \leftarrow \overline{A}$	Compliment 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

TABLE: Sixteen Logic Microoperation

Hardware Implementation

- Although there are 16 logic micro operation, most computers use only four—AND, OR, XOR (exclusive-OR), and complement from which all others can be derived.
- Below figure shows one stage of a circuit that generates the four basic logic micro operations.

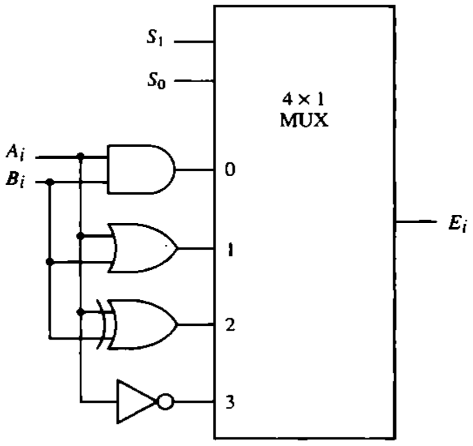


Figure : One stage of logic circuit

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	$E = \bar{A}$	Compliment

Table: Function table

- Hardware implementation consists of four gates and a multiplexer.
- The outputs of the gates are applied to the data inputs of the multiplexer.
- The two selection inputs S1 and S0 choose one of the data inputs of the multiplexer and direct its value to the output.
- The diagram show single stage called i. For a logic circuit with n bits, the diagram must be repeated n times for i = 0, 1, 2 . . . n - 1. The selection variables are applied to all stages.

16. Applications of logical microoperation

1) Selective-Set operation:

- The selective-set operation sets bits equal to 1 in register A where there are corresponding 1's in register B.
- It does not affect bit positions that have 0's in B.
- Example:

1010	A before
$\underline{1100}$	B (logical operand)
1110	A after

2) Selective-Complement operation:

- The selective-complement operation complements bits in A where there are corresponding 1's in B.
- It does not affect bit positions that have 0's in B.
- For example:

1010	A before
$\underline{1100}$	B (logical operand)
0110	A after

- The exclusive-OR microoperation can be used to selectively complement bits of a register.

3) Selective-Clear operation:

- The selective-clear operation clears the bits to 0 in A only where there are corresponding 1's in B. For example:

1010	A before
$\underline{1100}$	B (logical operand)
0010	A after

- The corresponding logic microoperation is $A \leftarrow A \wedge B$.

17. Explain shift micro operations and draw 4-bit combinational circuit shifter.

- Shift micro operations are used for serial transfer of Data.
- The contents of a register can be shifted to the left or right.

There are 3 types of shift micro-operations:

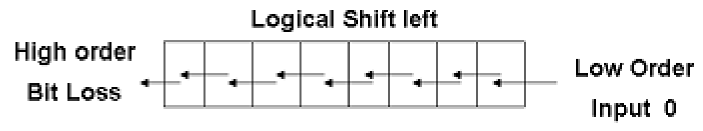
1. Logical Shift:

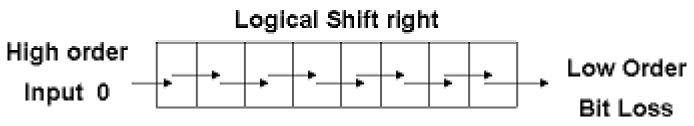
- A logical shift is one that transfers 0 through the serial input.
- We will adopt the symbols shl for logical shift-left and shr shift-right micro-operations.
- For example:

$$R1 \leftarrow \text{shl } R1$$
$$R2 \leftarrow \text{shr } R2$$

are two micro-operations that specify a 1-bit shift to the left of the content of register R1 and a 1-bit shift to the right of the content of register R2.

- The register symbol must be the same on both sides of the arrow.
- The bit transferred to the end position through the serial input is assumed to be 0 during a logical shift.

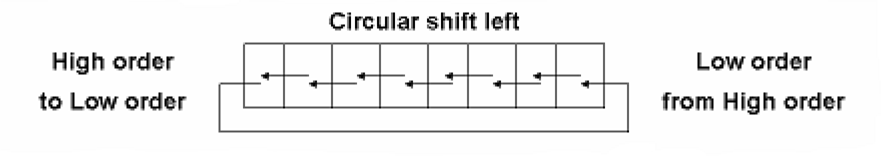


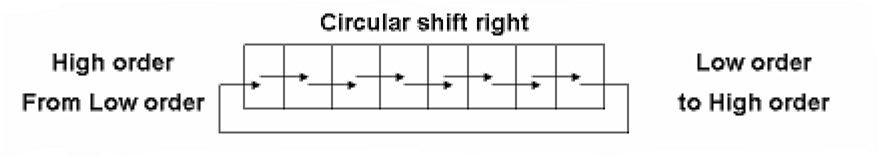


2. Circular Shift:

- The circular shift (also known as a rotate operation) circulates the bits of the register around the two ends without loss of information.
- This is obtained by connecting the serial output of the shift register to its serial input
- We will use the symbols cil the circular shift left and cir for circular shift right, respectively.

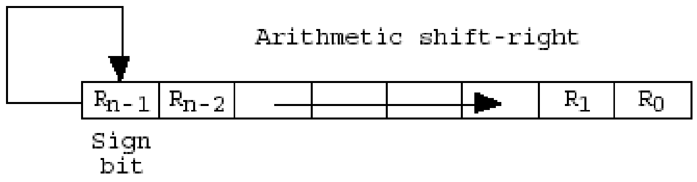
$$R1 \leftarrow \text{Cil } R1$$
$$R2 \leftarrow \text{Cir } R2$$

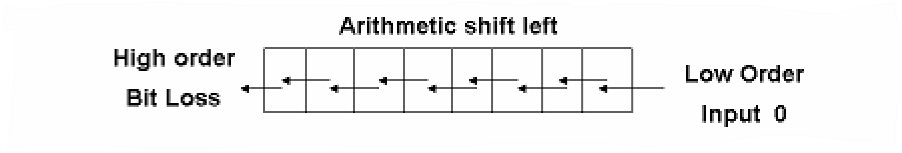


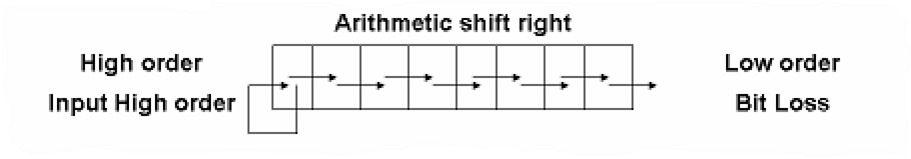


3. Arithmetic Shift:

- An arithmetic shift is a micro-operation that shifts a signed binary number to the left or right.
- An arithmetic shift-left multiplies a signed binary number by 2.
- An arithmetic shift-right divides the number by 2.
- Arithmetic shifts must leave the sign bit unchanged because the sign of the number remains the same when it is multiplied or divided by 2.
- The leftmost bit in a register holds the sign bit, and the remaining bits hold the number.
- The sign bit is 0 for positive and 1 for negative.
- Negative numbers are in 2's complement form.







18. Draw and explain one stage of arithmetic logic shift unit.

- To perform a microoperation, the contents of specified registers are placed in the inputs of the common ALU.
- The ALU performs an operation and the result of the operation is then transferred to a destination register.
- The ALU is a combinational circuit so that the entire register transfer operation from the source registers through the ALU and into the destination register can be performed during one clock pulse period.
- One stage of an arithmetic logic shift unit is shown in figure below:

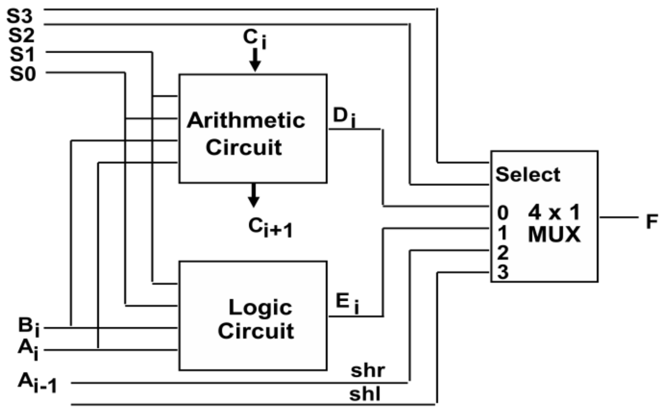


Figure: One stage of arithmetic logic shift unit

- Inputs Ai and Bi are applied to both the arithmetic and logic units.
- A particular micro operation is selected with inputs S1 and S0.
- A 4 x 1 multiplexer at the output chooses between an arithmetic output in Di , a logic output in Ei, shr and shl.
- The data in the multiplexer are selected with inputs S3 and S2.
- Inputs Ai-1 for the shift-right operation and Ai+1 for the shift-left operation.
- Note that the diagram shows just one typical stage. The circuit shown in figure must be repeated n times for an n-bit ALU.
- The outputs carry Ci+1 of a given arithmetic stage must be connected to the input carry Cin of the next stage in sequence.

- The circuit whose one stage is specified in figure provides:- 8 arithmetic operation, 4 logic operations, 2 shift operations
- Each operation is selected with the five variables S3, S2, S1, S0, and Cin. The input carry Cin is used for selecting an arithmetic operation only.
- The first eight are arithmetic operations and are selected with S3S2 = 00.
- The next four are logic operations are selected with S3S2 = 01. The input carry has no effect during the logic operations and is marked with don't-care X's.
- The last two operations are shift operations and are selected with S3S2 = 10 and 11.
- The other three selection inputs have no effect on the shift.

Table below lists the 14 operations of the ALU.

Operation Select					**Operation**	**Function**
S_3	**S_2**	**S_1**	**S_0**	**C_{in}**		
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**	**X**	**$F = A \wedge B$**	**AND**
0	**1**	**0**	**1**	**X**	**$F = A \vee B$**	**OR**
0	**1**	**1**	**0**	**X**	**$F = A \oplus B$**	**XOR**
0	**1**	**1**	**1**	**X**	**$F = \overline{A}$**	**Complement A**
1	0	X	X	X	F = shr A	Shift right A into F
1	1	X	X	X	F = shl A	Shift left A into F

Table: Function Table for Arithmetic Logic Shift Unit