Register Transfer:-

The Availability of hardware logic circuit that can Perform a stated microoperation & the result of o/p to the same (or) other register is called as "Register Transfer".

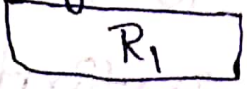
Eg:- Info from 1 reg to another reg is designated in symbolic form by means of replacement operator

$$R_2 \leftarrow R_1$$

Computer registers are designated by capital letters (or) sq PC (Program Counter), IR (Instruction Register). The Numbering in register will be starting from 0 in the right most position and increasing towards left.

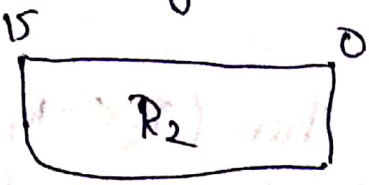
The way of representing is as follows:

(1) A Register is represented by a rectangular box with name inside register



Register R

(2) The Numbering of bits in a 16 bit register can be marked on top of the box.



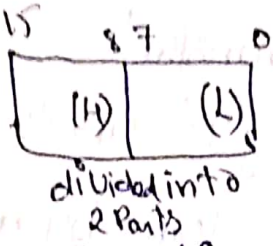
Numbering of bits

(3) The individual bits can be distinguished.

7	6	5	4	3	2	1	0

showing individual bits

(4) The bits from 0 to 7 are represented with L (lower bits) & bits from 8 to 15 are represented with H (higher bits)



The information transfer from one register to another is represented as $R_2 \leftarrow R_1$

→ In the above statement, it denotes transfer of contents from R_1 to R_2. The contents of R_2 are replaced by contents of R_1

→ The source Register R_1 does not change its contents after the transfer.

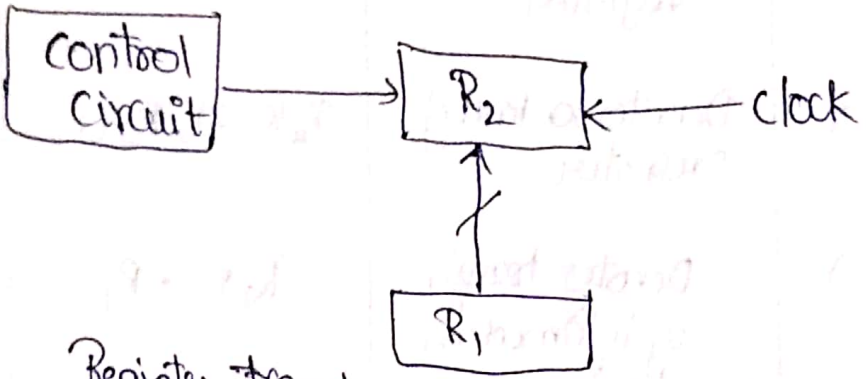
→ To perform a register transfer:

* output lines of source register are connected to i/p lines of destination register
* Destination register has Parallel load capacity
* A Predetermined controlled condition can control the transfer from one register to another register.

If (P=1) then ($R_2 \leftarrow R_1$)

P is a control variable, when P=1 then contents of R_1 & R_2 takes place & represented as $P: R_2 \leftarrow R_1$

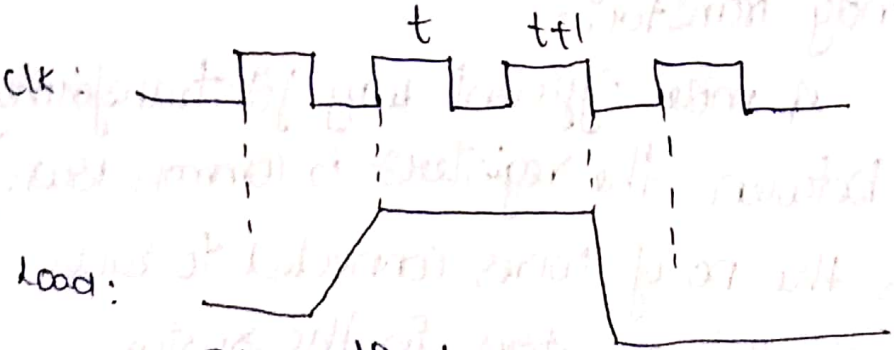
The control condition P is terminated with a colon & the transfer operation is specified after colon.



Register transfer from R_1 to R_2 when P=1

The n outputs of register R_1 are connected to n inputs of register R_2. n indicates no. of bits in the register. R_2 has a load input i.e activated by control variable "P". The control variable must be synchronize with clocks applied to register R_2

fig: Timming diagram



P is activated by raising Edge of clock pls at time t. At time t+1 the load i/p is active and data i/p of R_2 are loaded into Register.

Basic Symbols for register Transfer:

Symbol	Description	Examples
(1) Letters (and Numbers)	Denotes a register	$M[AR], R_2$
(2) Parenthesis ( )	Denotes a part of register	$R_2(0-7), R_2(L)$
(3) Arrow ($\leftarrow$)	Denotes transfer of information	$R_2 \leftarrow R_1$
(4) Comma (,)	Seperates 2 micro operations	$R_2 \leftarrow R_1, R_1 \leftarrow R_2$

Register Transfer Language:-

The Symbolic notation used to describe the micro operation transfer among registers is called as Register transfer language.

Bus and Memory Transfer:-

A more Efficient way for transferring information between the registers is common Bus system. This reduces the no. of wires connected to each register & all other registers in the System.

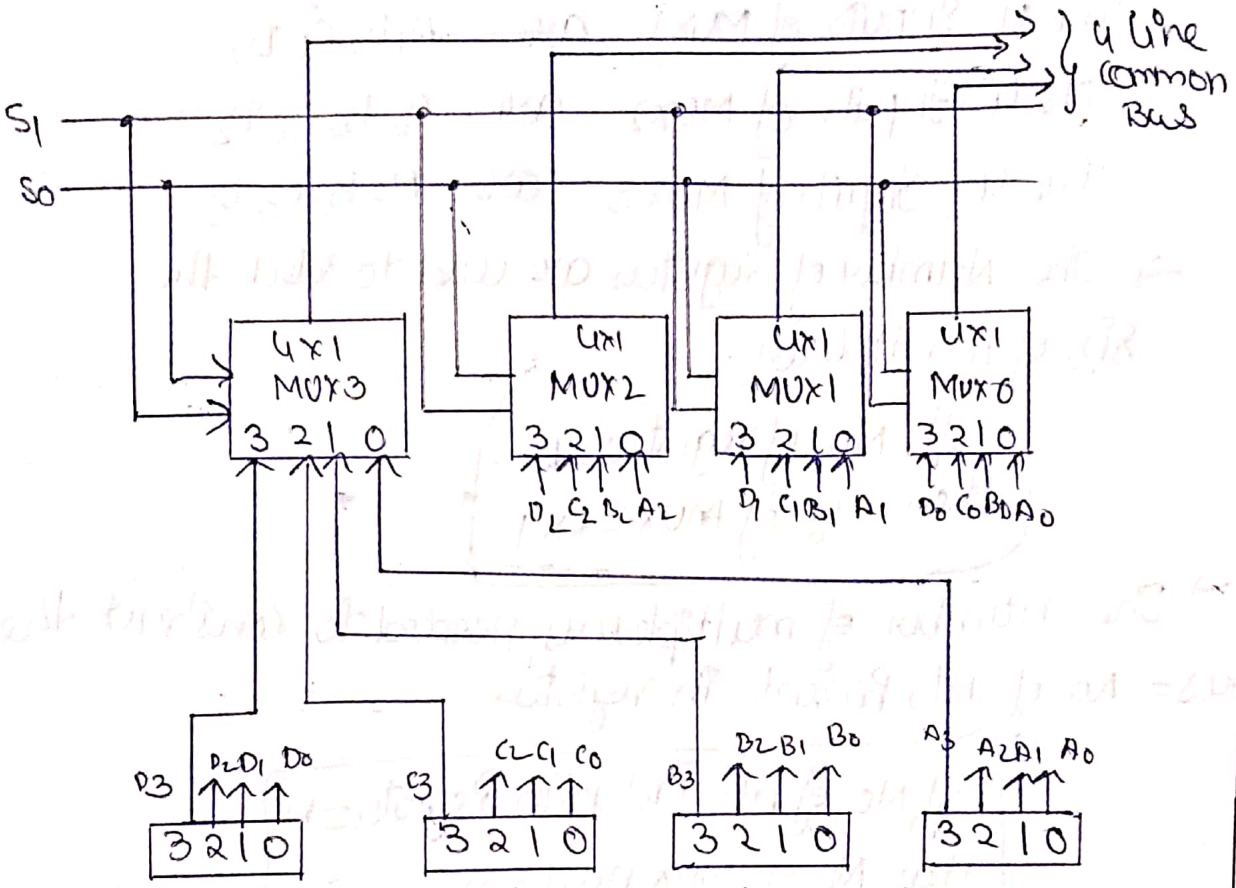
A Bus structure consists of common lines one for each bit of register through which binary information is transfered one at a time.

The Common Bus System is constructed using

(1) Multiplexers

(2) Three state Buffers.

Using multiplexers:-



The Multiplexer selects the source register whose contents are to be placed on bus.

In the fig; there are 4 registers each containing 4bits from 0 to 3.

The Bus consists of 4x1 Multiplexers each having 4 data inputs & 2 selection inputs.

The 2 selection lines S_1 & S_0 are used to select register as follows.

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

The 4 Inputs of MUX 0 are $A_0 B_0 C_0 D_0$
The 4 Inputs of MUX 1 are $A_1 B_1 C_1 D_1$
The 4 Inputs of MUX 2 are $A_2 B_2 C_2 D_2$
The 4 Inputs of MUX 3 are $A_3 B_3 C_3 D_3$

→ The Number of registers are used to select the size of multiplexer.

> If No. of registers = k
> then Size of MUX = k x 1

→ The number of multiplexers needed to construct the bus = No. of bits Present in register

> If, No. of bits Present in Register = n
> then, No. of MUX need = n

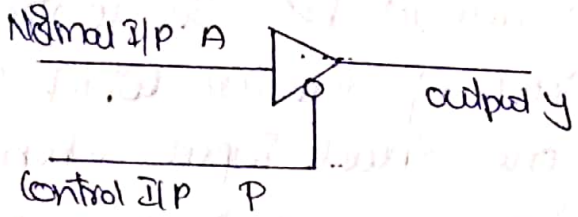
The Symbolic Statement for a Bus transfer is as follows:

> Bus ← C → register
> R_1 ← Bus

The Content of register C is placed on bus & the Content of bus is transfered to register R_1. The direct transfer of these statements are

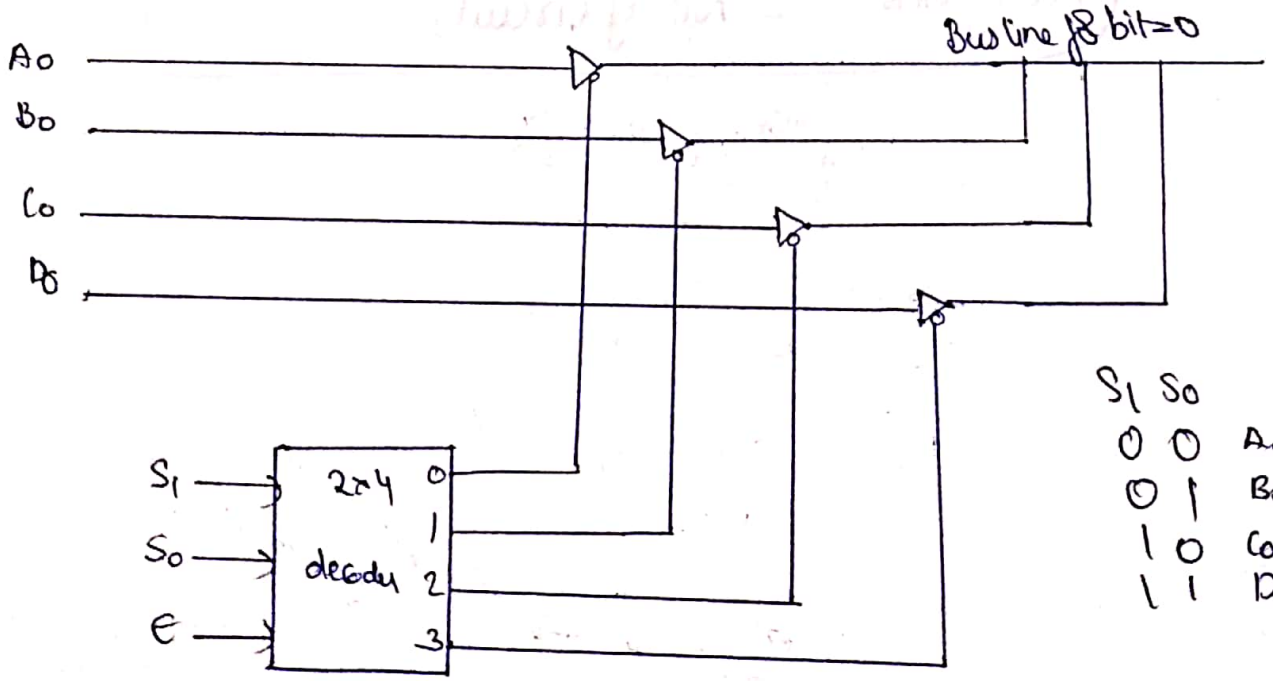
$R_1 \leftarrow C$

Using three state Buffer:-



A three state Buffer is a digital circuit that exhibit 3 states, two of signals & equivalent to 1 and 0. The third state is high impedence state

It is distinguished from a normal buffer by having two inputs - Normal input and control input. The control input determines output state. when P=1 then output is enable & equal to normal input when P=0, the output is disableles and goes to impedence state



S_1	S_0	
0	0	A_0
0	1	B_0
1	0	C_0
1	1	D_0

Bus line with three state Buffer

The construction of a Bus system with 3-State Buffers we have outputs of 4 buffers connected together to form a simple bus line for 1 bit. The control inputs to buffers are the output of decoder which have two selection inputs and one Enable input. When e=1 the decoder is Enable & corresponding output is selected based on selection inputs. If $S_1S_0=00$ then the buffers with input A_0 is activated and bit A_0 is placed on bus line for bit 0.

To construct a common bus for 4 registers of n bit each using 3 state buffer we need n circuits with 4 buffers in each circuit.

No. of Registers = No. of Buffers in Each circuit

No. of Bits = No. of circuits

ARITHMETIC MICRO OPERATIONS

A micro operation is an elementary operation performed with the data stored in registers. Arithmetic micro operations perform arithmetic operations on numeric data stored in registers. The basic arithmetic micro operations are addition, subtraction, increment, decrement and shift. The descriptions of the micro operations described in the table below.

Symbolic Designation	Description
$R_3 \leftarrow R_1 + R_2$	Contents of R1 plus R2 transferred to R3
$R3 \leftarrow R1 - R2$	Contents of R1 minus R2 transferred to R3.
$R2 \leftarrow \bar{R2}$	Complement the contents of R2 (1's comp)
$R2 \leftarrow \bar{R_2} + 1$	2's complement the contents of R2 (negate)
$R3 \leftarrow R1 + \bar{R2} + 1$	R1 plus the 2's comp 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.

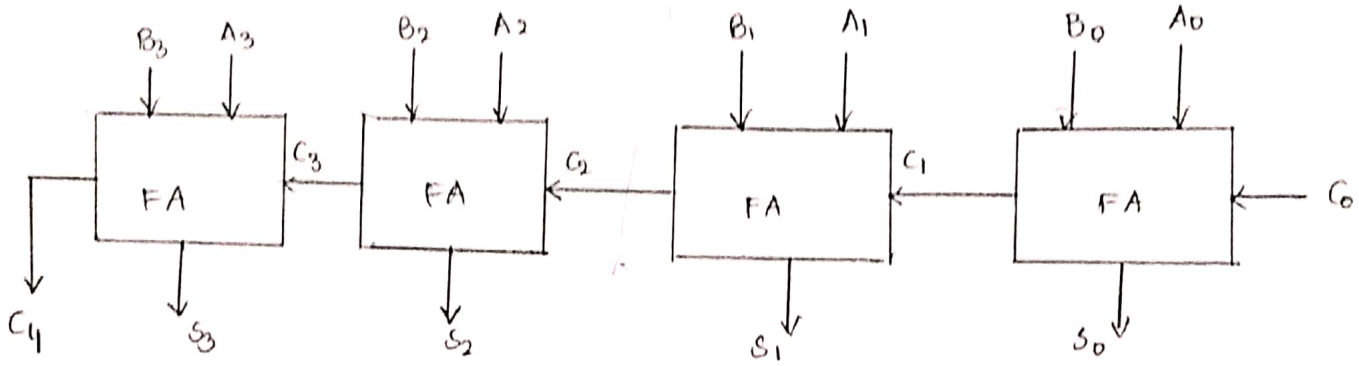
Binary Adder:

The digital circuit that forms the arithmetic sum of two bits and a previous carry is called a full adder. The digital circuit that generates the arithmetic sum of two binary numbers of any length is called a binary adder.

An n-bit binary adder requires n full address. The output carry from

Binary Adder:

The digital circuit that forms the arithmetic sum of two bits and a previous carry is called a full adder. The digital circuit that generates the arithmetic sum of two binary numbers of any length is called a binary adder.



4- bit binary adder

An n-bit binary adder requires n full address. The output carry from each full-adder is connected to the input carry of the next-high-order full adder. The n data for bits for the A inputs come from one register (such as R1) and then n data bits for B inputs come from another register (such as R2). Then sum is can be transferred to a third register or to one of the source registers (R1 or R2), replacing its previous content.

Ex: A + B

$R_1 + R_2$

4 bits ↓ 4 bits ↓

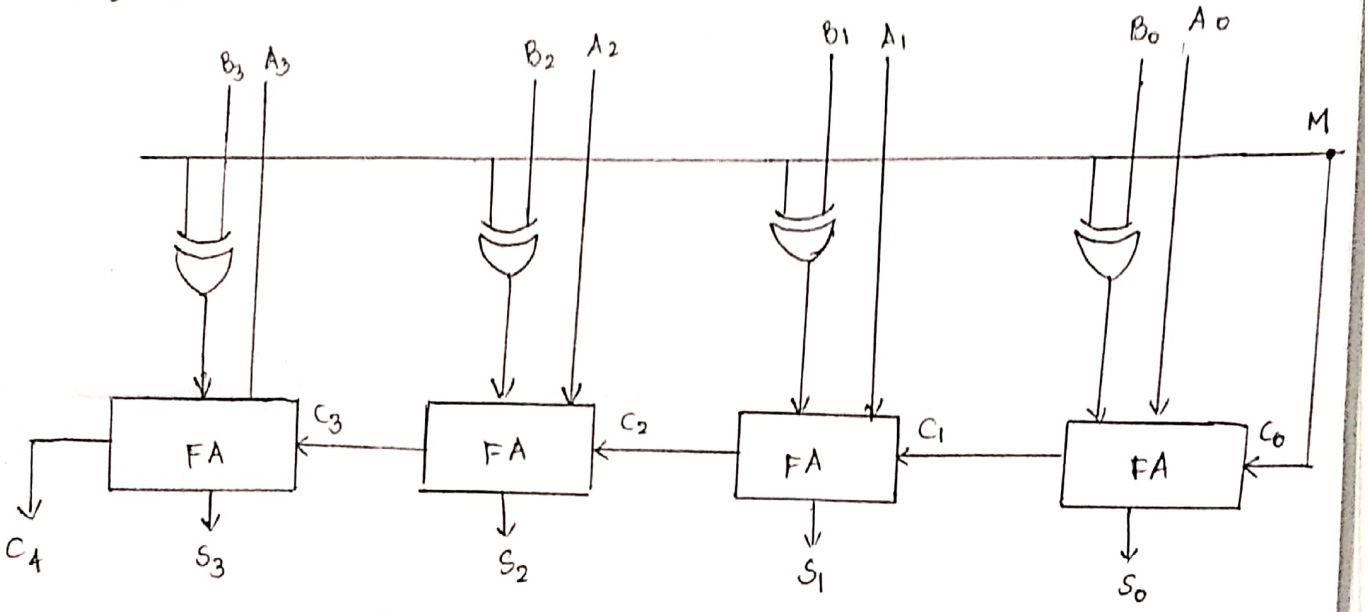
$R_1 = 0\ 0\ 1\ 1 \rightarrow (1)$ (1 1 carry)

$R_2 = 0\ 0\ 0\ 1 \rightarrow (2)$

$0\ 1\ 0\ 0$

ary Adder-Subtractor:

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 and when M=1, then circuit becomes a subtraction.



$$O/P = A + B + 0$$

$$= A + B$$

$$= A + \bar{B} + (1) \quad (M \Downarrow) \quad = A - B$$

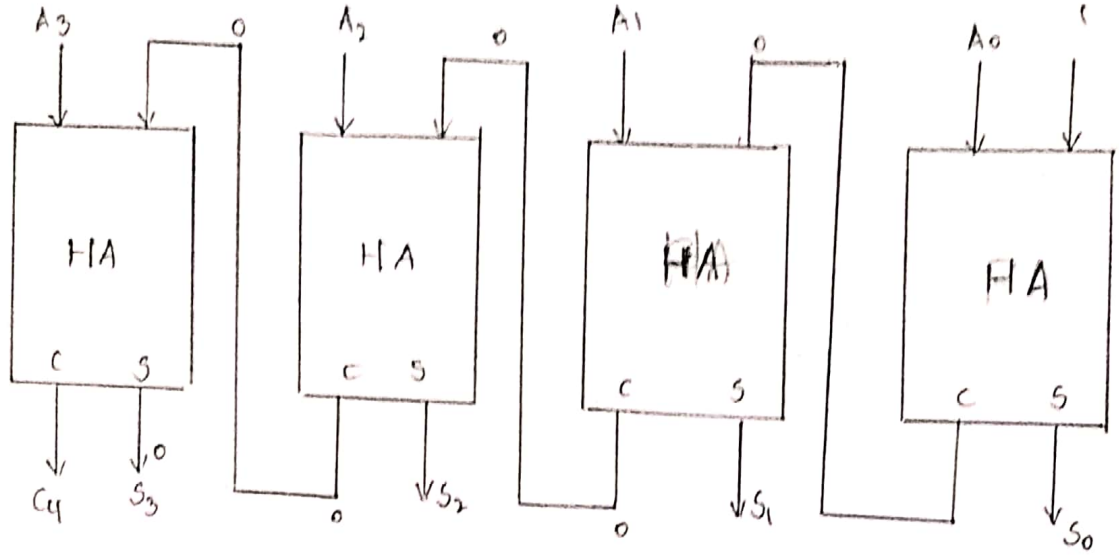
M = 0 ⇒ Addition

M = 1 ⇒ Subtraction

Binary Increment:

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. Everytime the count

ble is active, the clock pulse transition increments the contents of the register by one.



Arithmetic circuit:

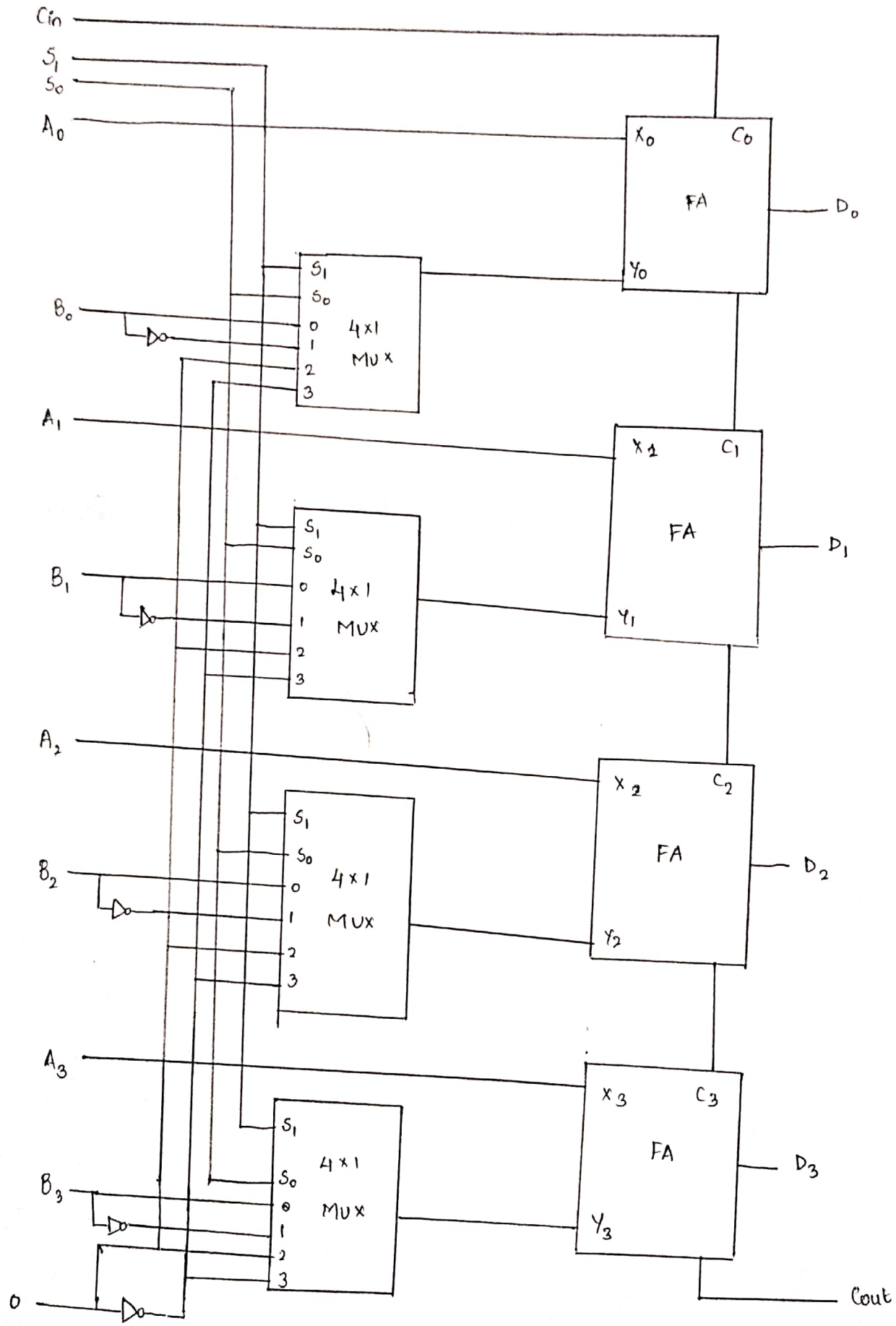
The arithmetic circuit micro operations can be implemented in one composite arithmetic circuit. The basic component of an arithmetic circuit is the parallel adder. By controlling the data inputs to the adder, it is possible to obtain different types of arithmetic operations. It has 4 full-adder circuits that constitute the 4-bit adder or four multiplexers for choosing different operations.

The output of the binary adder is calculated from the following arithmetic sum:

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

where A is the 4-bit binary number at the X-inputs &

is the 4-bit binary number at the Y-inputs of the binary adder, Cin is the input carry.



Arithmetic Circuit Function Table

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

K. Avinash kiran
B5 22

1. (4)

Logic Microoperation

Logic microoperations specify binary operations for strings of bits stored in registers. These operations consider each bit of the register separately and treat them as binary variables.

Special symbols are there for logic microoperations. The symbol $\vee$ is used to denote an OR operation, $\wedge$ to denote an AND microoperation. The complement microoperation is denoted by putting a bar on top of the symbol.

For example,

$R_1 \leftarrow R_2 + R_3$

$R_4 \leftarrow R_5 \vee R_6$.

List of Logic Microoperations :-

Truth table for 16 functions of 2 variables

x	y	F_0	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{13}	F_{14}	F_{15}
0	0	0	0	0	0	0	0	0	0	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	0	0	0	1	1	0	0	1	1	0	0	1	1	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	1	0	1

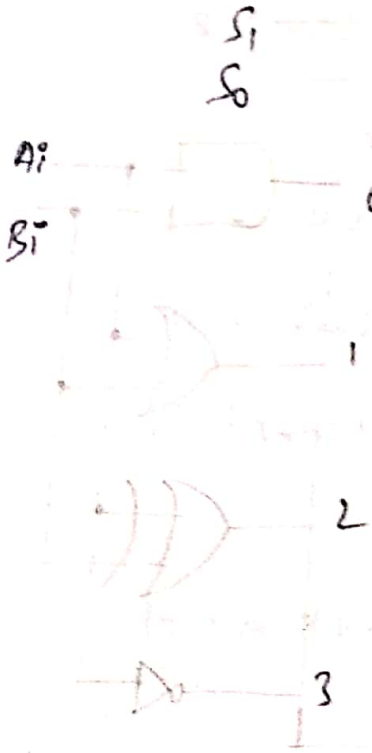
In the above table there are 16 different logic microoperations that can be performed by 2 variables. The columns F_0 through F_{15} represents a truth table of one possible boolean function. Note that the functions are determined from the 16 binary combinations that can be assigned to F. Each bit of the register is treated as a binary variable and the microoperation is performed on the string of bits stored in the register.

Sixteen Logic Microoperations

Boolean Function	Microoperation	Name
$F_0 = 0$	$F \leftarrow 0$	Clear
$F_1 = xy$	$F \leftarrow A \wedge B$	AND
$F_2 = xy'$	$F \leftarrow A \wedge \bar{B}$	
$F_3 = x$	$F \leftarrow A$	Transfer A
$F_4 = x'y$	$F \leftarrow \bar{A} \wedge B$	
$F_5 = y$	$F \leftarrow B$	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
$F_{10} = y'$	$F \leftarrow \bar{B}$	complement B
$F_{11} = x + y'$	$F \leftarrow A \vee \bar{B}$	
$F_{12} = x'$	$F \leftarrow \bar{A}$	complement A
$F_{13} = x' + y$	$F \leftarrow \bar{A} \vee B$	
$F_{14} = (xy)'$	$F \leftarrow \overline{A \wedge B}$	NAND
$F_{15} = 1$	$F \leftarrow$ all 1's	Set to all 1's

Hardware Implementation:

The hardware implementation of logic microoperations requires that logic gates be inserted for each bit or pair of bits in the registers to perform the required logic function. Although there are 16 microoperations most use only four - AND, OR, XOR and complement from which all can be derived.



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	$E = \bar{A}$	complement

b) Function table

Applications:-

i) Selective-Set :- The selective-set operation sets to 1 the bits in register A where there are corresponding 1's in register B.

For example,

```
1010    A before
1100    B (logic operand)
----
1110    A after.
```

→ We can implement a selective set operation using OR gate.

→ If two leftmost are 1's the corresponding of A are set to 1. If two leftmost are 0's the B remains unchanged.

ii) Selective compliment :-

It compliments the values of register A where the corresponding bit position are complemented.

```
A=  011011   A (before)
B=  110110   B (logic operand)
----------
    101101   (A after)
```

→ If the left most bits are 0 then transfer the same bit. If it is 1 then compliment the bit position.

→ We can compliment using X-OR gate.

iii) Selective clear:-

→ It clears the content of register A with the corresponding 1's in the respective position.

A = 0 1 1 0 1 1 0 1
0 1 1 0 1 0 1 0
---
0 0 0 0 0 1 0 1

→ If the leftmost is 0 then transfer the same content.

→ If the leftmost is 1 then put 0 / Reset / clear..

iv) Masking Operation:-

It is similar to the selective clear operation whereas the significance is given to zeros in place of 1's in the 2nd register.

A = 0 0 1 1 0
B = 0 0 1 1 1
---
0 0 1 1 0

→ If the leftmost is 1 then transfer the same bit or else clear / 0 / Reset.

(V) Insert operation :-

This operation inserts a new value into the group of bits. This has 2 stages

i, Masking of bits

ii, OR operation.

Ex :-

1st stage

A = 0110 0111

0000 1111

After Masking = 0000 0111

OR gate = 1001 0000

1001 0111

Here we mask the values which are to be changed and insert the values which we need.

(4)

1. (5)

Write about Shift Microoperations ?

Definition :-

Shift microoperations are used for serial transfer of data. They are also used in conjunction with arithemetic, logic and data processing operations. The contents of register are shifted to left or right

During the shift left operation the serial input transfers a bit into the rightmost position.

During the shift right operation the serial input transfers a bit into the leftmost position.

Types of Shift :-

1) Logical Shift

2) Circular Shift

3) Arithemetic Shift.

Logical Shift :-

A logical shift is one that transfers '0' through a serial input. We will adopt the

Symbols 'Shl' and 'Shr' for logical shift left & Ashr logical shift right micro-operations. A = 0 1 1 0.

		Shift left	Shift right.
$R_1 \leftarrow$ Shl R_1		0 1 1 0 ↓ 0	↓ 0 1 1 0
$R_2 \leftarrow$ Shr R_2		1 1 0 0	0 0 1 1

The above two microoperations specify a 1-bit shift to the left of R_1 content and 1 bit shift right of R_2 content registers. The register symbol must be on the both sides of arrow. The bit transferred to end position through a serial input is assumed to be '0' during logical shift

Circular Shift

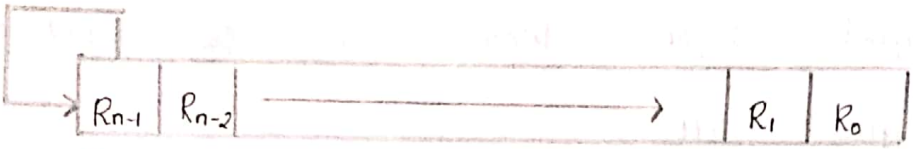
The Circular shift (also known as rotate Operation) circulates the bits of the register around the two ends without the loss of information. This is accomplished by connecting serial output of shift register to its serial input. We will use symbols Cir and Cil for circular shift right and Circular shift left respectively.

$R \leftarrow$ Cil R

$R \leftarrow$ Cir R

Arithemetic Shift :-

An arithemetic shift is a microoperation that shifts a signed binary number to left or right. An arithemetic shift - right divides the number by 2. An arithemetic shift - left multiplies a signed binary number by 2. Arithemetic shifts must leave the sign bit unchanged because the sign of number remains the same when it multiplied and divided by 2.



Sign bit

Arithemetic Shift left right.

Sign bit

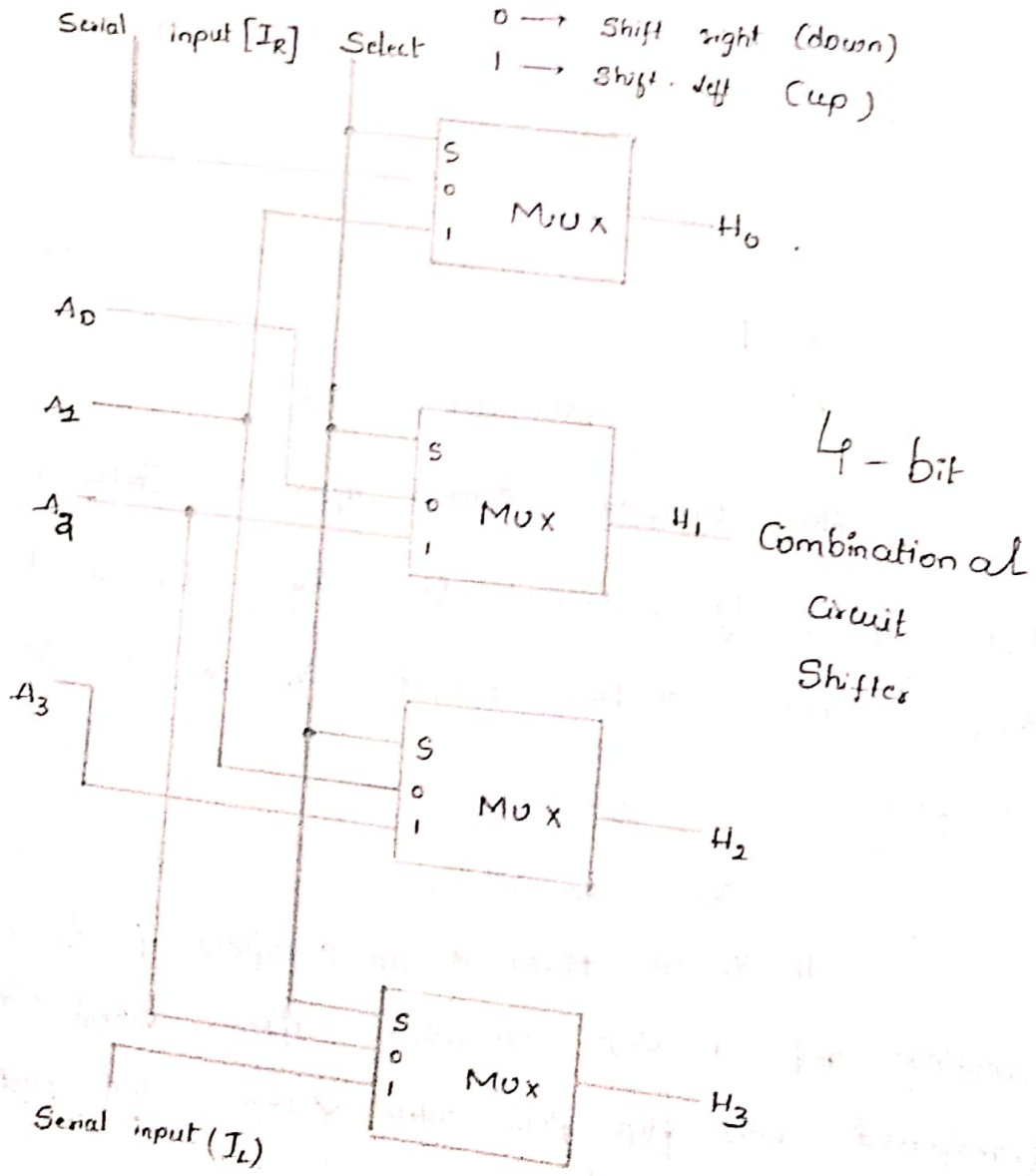
Arithemetic Shift left.

An overflow occurs after an Arithemetic Shift left if initially, before the shift, R_{n-1} is not equal to R_{n-2}. An overflow flipflop V_s can be used to detect overflow.

$$V_S = R_{n-1} \oplus R_{n-2}$$

If $V_S = 0$ there is no overflow, if $V_S = 1$ there is overflow and a sign reversal after shift. V_S must be transferred into flip-flop with same clock pulse that shifts the register.

The combitional circuit Shifter can be Constructed using multiplexers. The 4-bit Shifter has four data inputs A_0 to A_3 and four data outputs H_0 to H_3. There are two serial inputs, one for Shift left $[I_L]$ and other for Shift right $[I_R]$. When Selection input $S=0$, the input data are Shifted right. When $S=1$, the input data are Shifted left.



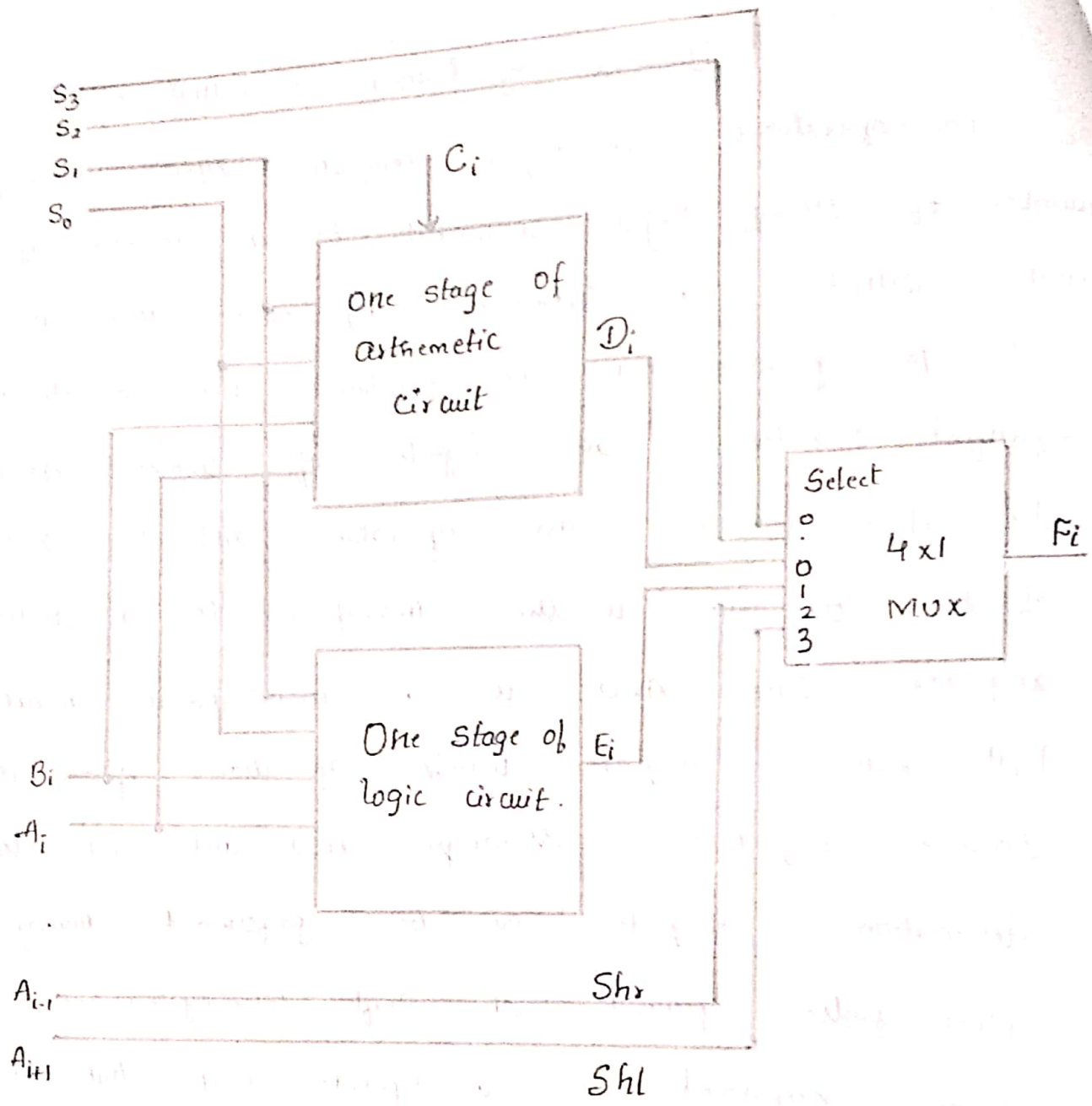
4-bit Combinational Circuit Shifter

Write about Asthemetic Logic Shift Unit (ALSU)? Explain

Instead of having individual registers the microoperations directly, computer systems employ a number of storage registers connected to a common operational unit called a Asthemetic logic unit, abbreviated ALU. To perform a microoperation, the contents of specified registers are inputs of common ALU. The ALU performs an operation and the result of the operation is then transfered to a destination register. The ALU is a combinational circuit so that entire register transfer operation from the source registers through ALU and into the destination register can be peformed during one clock pulse period. The Shift microoperations are often performed in a separate unit, but sometimes Shift unit is made part of overall ALU.

The arthemetic, logic and shift circuits can be introduced in previous sections as one ALU with common selection variables. One stage of Asthemetic logic Shift unit is show in figure. The Subscript 'i' designates a typical stage.

One Stage of Arithmetic logic Shift unit



The Circuit provides eight arithmetic Operation, four logic operations and 2 shift operations. Each operation is selected with five variables S_3, S_2, S_1, S_0 and and C_{in}. The input carry C_{in} is used for Arithmetic Operation only.

Function Table for Arithmetic Logic Shift Unit.

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	$F = A+\bar{B}$	Subtract with borrow
0	0	1	0	1	$F = A+\bar{B}+1$	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	$F = \bar{A}$	Complement A
1	0	x	x	x	Shr A	Shift right A into F
1	1	x	x	x	Shl A	Shift left A into F

The above table lists the operations of ALU.

The first eight are arithmetic operations and are selected with $S_3S_2 = 00$.

The next four has no effect during logic operations and are selected with $S_3S_2 = 01$.

The carry input has no effect during logic operations and is marked with 'X's.

The last two operations are shift operations and are selected with $S_3S_2 = 10$ and 11.

The other three selection inputs have no effect on shift.