Register Transfer and Micro-Operations: Register Transfer Language, Register Transfer, Bus and memory Transfers, Arithmetic Micro-operations, Logic Micro-operations, Shift Micro-operations, Arithmetic Logic Shift Unit.

Basic Computer Organization and Design: Instruction codes, Computer Registers, Computer Instructions, Timing and Control, Instruction cycle, Memory-Reference Instruction, Input-Output and Interrupt Instructions

Register Transfer and Micro-Operations

Register Transfer:

- A register transfer is indicated as

$$R2 \leftarrow R1$$

  - In this case the contents of register R2 are copied (loaded) into register R1
  - A simultaneous transfer of all bits from the source R1 to the destination register R2, during one clock pulse
  - Note that this is a non-destructive; i.e. the contents of R1 are not altered by copying (loading) them to R2

Registers:

- Computer registers are designated by capital letters (sometimes followed by numerals) to denote the function of the register.
- For example, the register that holds an address for the memory unit is usually called a memory address register and is designated by the name MAR.
- Other designations for registers are PC (for program counter), IR (for instruction register), and R1 (for processor register).
- The individual flip-flops in an n-bit register are numbered in sequence from 0 through n - 1, starting from 0 in the rightmost position and increasing the numbers toward the left.
- The following diagram shows the representation of registers.

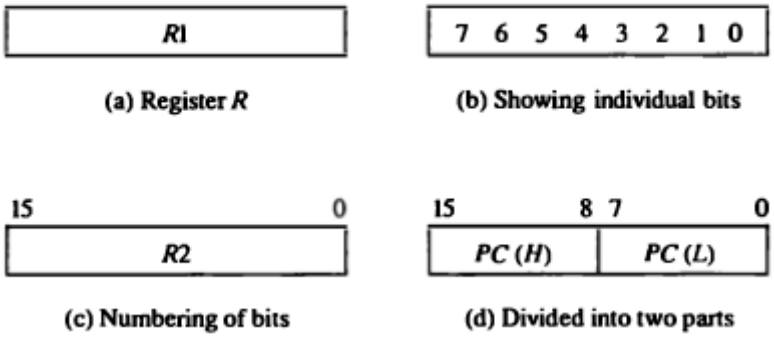


Fig :Register Transfer Language

- The symbolic notation used to describe the Micro operation transfers among registers is called a Register Transfer Language.

- Information transferred from one register to another register is designed in symbolic form by means of replacement operator (←).
- The statement R2 ← R1 denotes a transfer of the contents of register R1 into register R2.
- If we want to transfer only under a predefined condition, this can be shown by means of if-then statement.

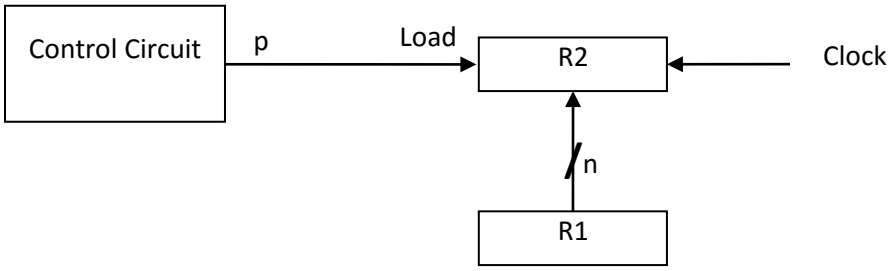
$$\text{if } (p = 1) \text{ then } R2 \leftarrow R1$$

Where p is a control signal generated in the control section.

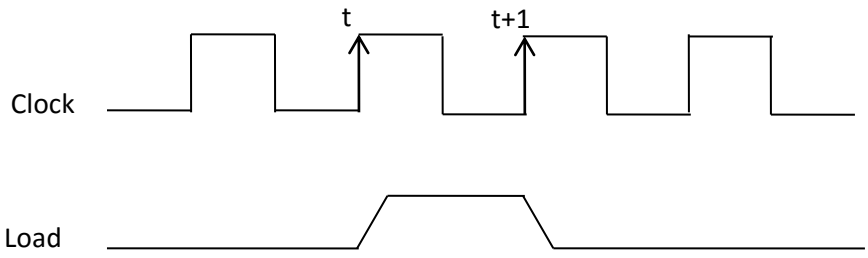
- A Control function is a Boolean variable that is equal to 0 or 1.
- The control function included in the statement is represented as follows.

$$p: R2 \leftarrow R1$$

- Here the transfer operation is performed by hardware only if p = 1.
- Every statement written in a register transfer notation implies a hardware construction for implementing the transfer.



(a) Block Diagram



(b) Timing Diagram

Figure 2: Transfer from R1 to R2 when p =1

The basic symbols of the register transfer notation are listed in the following table.

S.No	Symbol	Description	Example
1	tters (and Numbers)	notes a register	MAR, R2
2	renthesis ( )	notes a part of register	R2(0-7), R2(L)
3	row ←	notes transfer of information	R2 ← R1
4	mma ,	parates two microoperations	R2 ← R1, R1 ← R2

Table: Basic Symbols for Register Transfer

* A comma is used to separate two or more operations that are executed at the same time.
Eg: R2 ← R1, R1 ← R2

- The above statement denotes an operation that exchanges the content of two registers during one common clock pulse.

Bus Transfer:

- Digital computers have many registers, and paths must be provided to transfer information from one register to another register.
- The no. of wires will be excessive if separate lines are used between each registers and all other registers in the system.
- A Bus structure consists of a set of common lines, one for each bit of a register, through which binary information is transferred one at a time.
- Control signal determines which register is selected by the bus during each particular register transfer.

- The construction of a bus system for 4 registers is shown in the following figure.

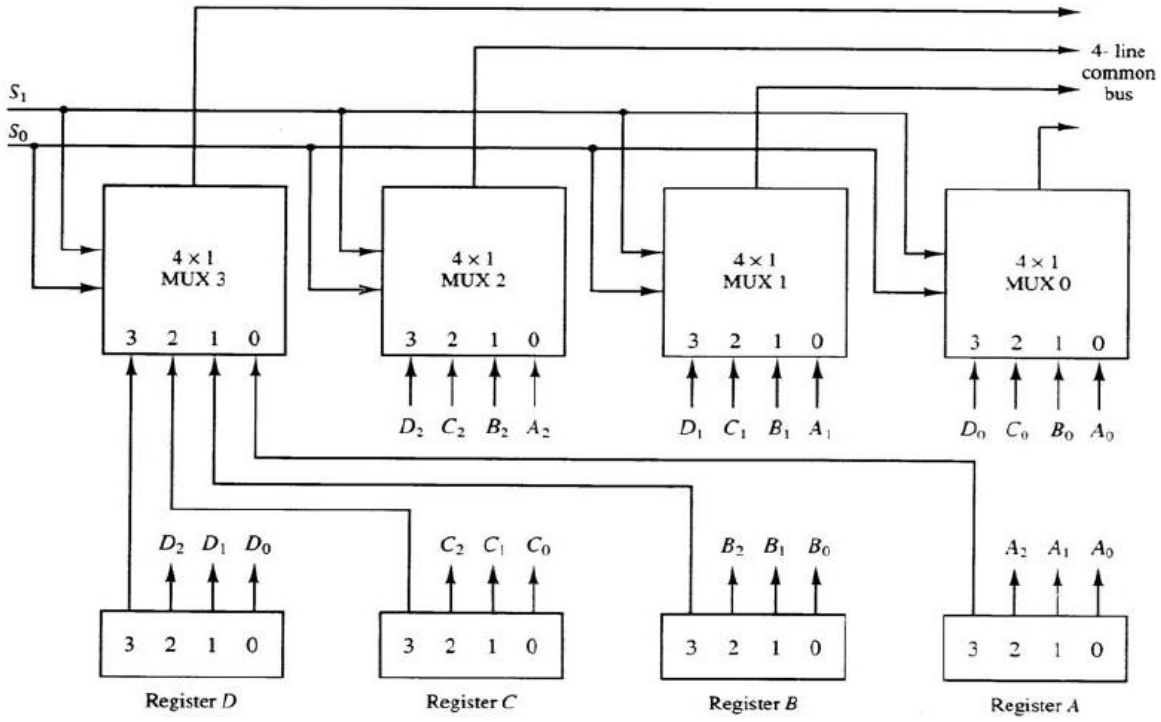


Figure : Bus system for four registers

- Here each register has 4 bits, numbered 0 through 3.
- The bus consists of four 4×1 multiplexers, each having four data inputs, 0 through 3, and two selection inputs S_1 and S_0.
- The selection lines choose the four bits of one register and transfer them into the 4-line common bus.
- When $S_1S_0 = 00$, then 0 data input of all four multiplexers are selected and applied to output that form the bus.
- The following table shows the register that is selected by the bus for each of the four possible binary values of selection lines.

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

Table : Selection of Registers

- The transfer of information from a bus into one of many registers can be accomplished by connecting the bus lines to inputs of all destination registers and activating the load control of particular destination register.

Eg: BUS ← C

- In the above statement the content of register C is placed onto BUS

Eg: R1 ← BUS

- In the above statement the content of register C, placed onto BUS is loaded into Register R1by activating the load pin.

Three-state Bus Buffer:

- A Bus system can be constructed with three-state gates instead of multiplexers.
- A three-state gate is a digital circuit that exhibits three states.
- Two of the states are signals equivalent to logic 1 or 0 as a conventional gate.
- The third state is a high-impedance state. The high-impedance behaves like an open circuit.
- The graphical symbol of a three state buffer is shown in the following figure.

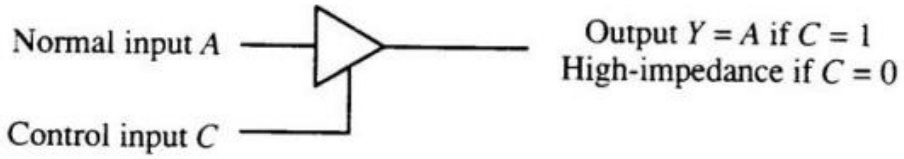


Figure : Graphic symbol of a three state buffer

- In the above diagram, control input determines the output state.
- When the control input is equal to 1, the output is enabled like as any conventional buffer.
- When the control input is equal to 0, the output is disabled and the gate goes to a high-impedance state.
- The following figure shows construction of a bus system with Three state Bus Buffer

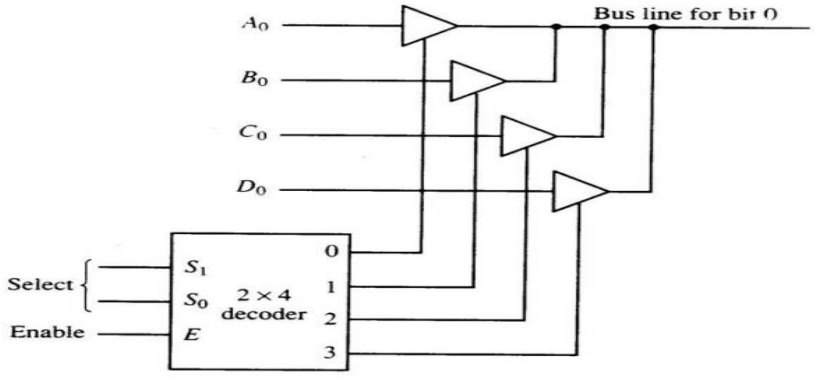


Figure : Bus Line with three state buffer

- Here the output of 4 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.

- Only one three-state buffer has access to the bus line while all other buffers are maintained in high impedance state.
- With the help of the decoder, three state buffers are controlled.
- When the enable input of the decoder is 0, all of its four outputs are 0, and the bus line is in a high-impedance state because all four buffers are disabled.
- When the enable input is active, i.e. 1, one of the three-state buffers will be active, depending on the select lines of the decoder.
- To construct a common bus for four registers of n bits each using three-state buffers, we need n circuits with four buffers in each circuit.
- Each group of four buffers receives one significant bit from the four registers.
- Each common output produces one of the lines for the common bus for a total of n lines.

E	**S_1 S_0**	**Active Three State Buffer**
0	X X	High impedance state
1	0 0	A
1	0 1	B
1	1 0	C
1	1 1	D

Table : Active Three State Buffer with respect to enable input

Memory Transfer:

- The transfer of information from a memory word symbolized by M, to the outside environment is called a *read* operation.
- The transfer of new information to be stored into the memory is called a *write* operation.
- The particular memory word among the many available is selected by the memory address during the transfer.
- Consider a memory unit that receives the address from a register, called the address register, symbolized by AR.
- The data is transferred to another register, called the data register, symbolized by DR.
- The memory read operation can be stated as follows

  Read: DR ←M [AR]
- Here data is transferred into DR from the memory location specified by AR.
- The memory write operation can be stated as follows

  Write: M [AR]← DR
- Here write operation transfers the content of a data register to a memory word M specified by the address in AR.

ARITHMETIC MICRO OPERATIONS:

The micro operations most often encountered in digital computers are classified into four categories:

1. Register transfer micro operations transfer binary information from one register to another.
2. Arithmetic micro operations perform arithmetic operation on numeric data stored in registers.
3. Logic micro operations perform bit manipulation operations on non-numeric data stored in registers
4. Shift micro operations perform shift operations on data stored in registers

Arithmetic micro operations:

- The basic arithmetic micro operations are addition, subtraction, increment and decrement.

Add Micro operation:

$$R3 \leftarrow R1 + R2$$

- The above statement states that the contents of register R1 are added to the contents of register R2 and the sum transferred to register R3.

Subtract Micro operation:

$R3 \leftarrow R1 + \overline{R2} + 1$

- In the above statement $\overline{R2}$ is the symbol for the 1's complement of R2. Adding 1 to the 1's complement produces the 2's complement. Adding the contents of R1 to the 2's complement of R2 is equivalent to R1 – R2.
- The increment and decrement micro operations are symbolized by plus-one and minus-one operations, respectively.

ymbolic 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
$R2 \leftarrow \overline{R2}$	omplement 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

Table : Arithmetic Micro operations

Binary Adder:

- To implement the add micro operation with hardware, we need the registers that hold the data and the digital component that performs the arithmetic addition.
- The binary adder is constructed with full-adder circuits connected in cascade, with the output carry from one full-adder connected to the input carry of the next full-adder.

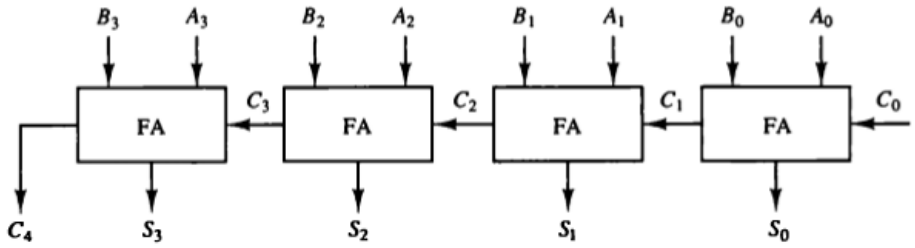


Figure : 4-bit binary adder

- The S outputs of the full-adders generate the required sum bits.
- An n-bit binary adder requires n full-adders. The output carry from each full-adder is connected to the input carry of the next-higher-order full-adder.

Binary Adder-Subtractor:

- The addition and subtraction operations can be combined into one common circuit by including an exclusive-OR gate with each full-adder.
- A 4-bit adder-subtractor circuit is shown in the following figure.

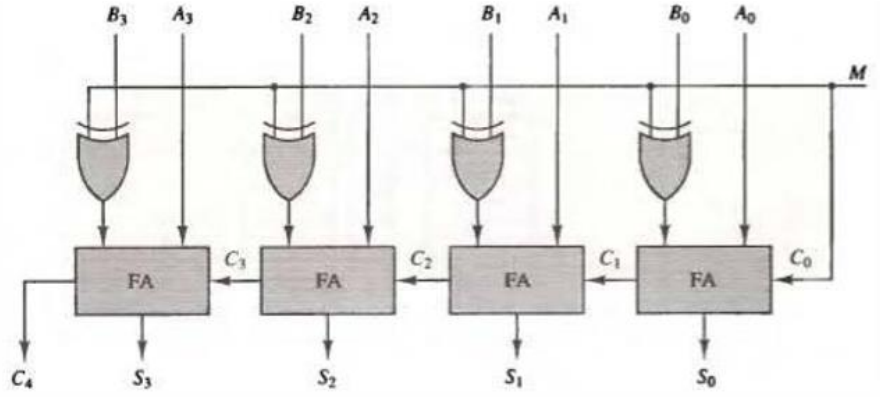


Figure : Adder-subtractor

- In the above figure, mode input M controls the operation.
- When M = 0 the circuit is an adder and when M = 1 the circuit becomes a subtractor.
- Each exclusive-OR gate receives input M and one of the inputs B. When M = 0, we have B⊕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 B⊕1 = B' and C_0 = 1. The B inputs are all complemented and 1 is added through the input carry. The circuit performs the operation A+2's complement of B.

Binary Incrementer:

- The increment micro operation adds one to a number in the register.
- The diagram of a 4-bit incrementer circuit is shown below. One of the inputs of the least significant half-adder (HA) circuit is connected to logic-1 and the other input is connected to the least significant bit of the number to be incremented.

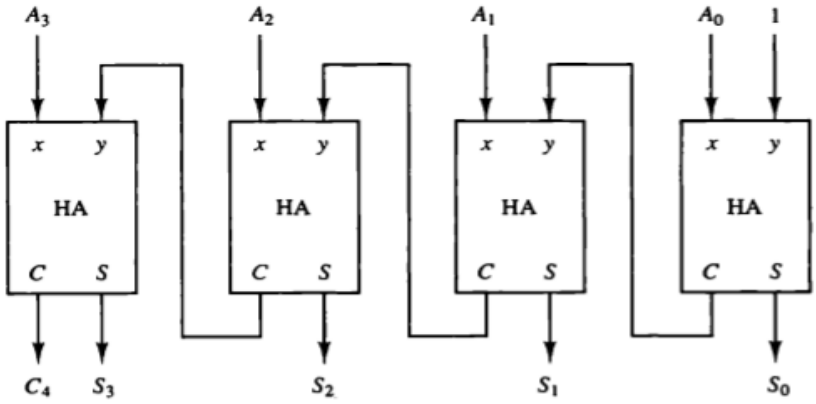


Figure : 4-Bit Binary Incrementer

- 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 A_0 through A_3, adds one to it, and generates the incremented output S_0 through S_3.

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.
- The diagram of a 4-bit arithmetic circuit is as shown in the following figure. It has four full-adder circuits that constitute the 4-bit adder and four multiplexers for choosing different operations.
- There are two 4-bit inputs A and B and a 4-bit output D. The four inputs from A go directly to the X inputs of the full adder. Each of the four inputs from B are connected to the data inputs of the multiplexers.
- The four multiplexers are controlled by two selection inputs, S_1 and S_0.

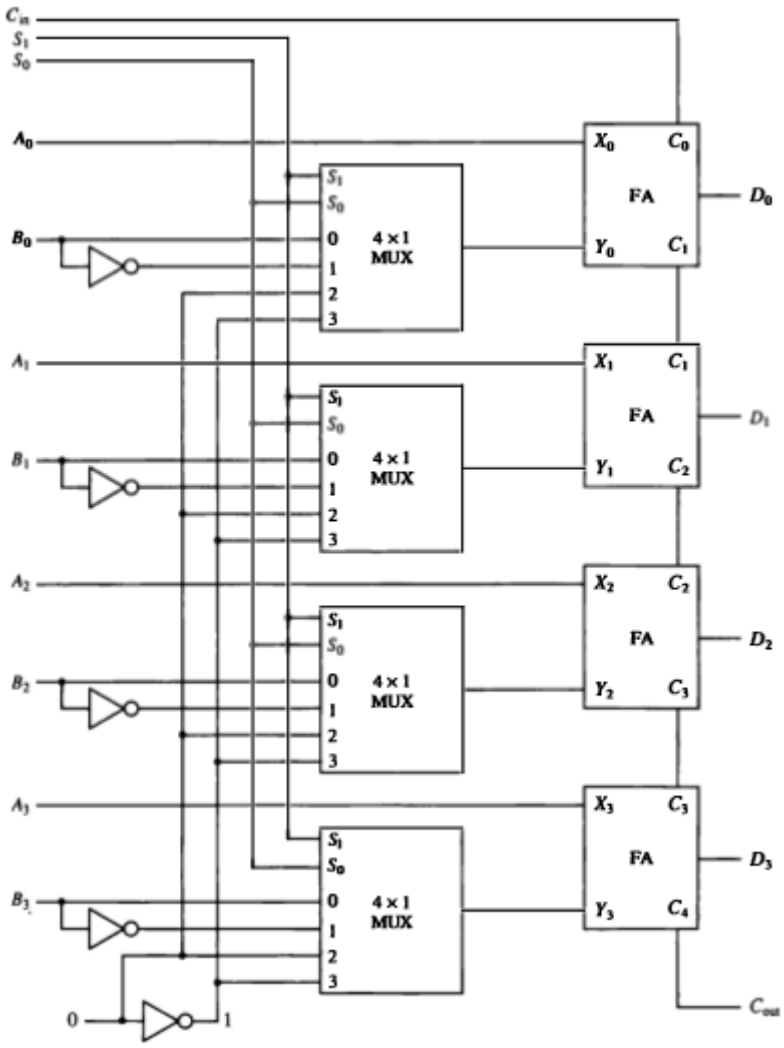


Figure : 4-bit arithmetic circuit

- The input carry C_{in} goes to the carry input of the FA in the least significant position. The other carries are connected from one stage to the next.
- 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 and Y is the 4-bit binary number at the Y inputs of the binary adder. C_{in} is the input carry, which can be equal to 0 or 1.
- By controlling the value of Y with the two selection inputs S_1 and S_0 and making C_{in} equal to 0 or 1, it is possible to generate the eight arithmetic micro operations listed in the following 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

Table : Arithmetic Circuit Function Table

Logic Micro operations:

- Logic micro operations 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.
- For example, the exclusive-OR micro operation on the contents of two registers R 1 and R2 is symbolized by the statement.

  P: R1 ← R1 ⊕R2
- The above specifies a logic micro operation to be executed on the individual bits of the registers provided that the control variable P = 1.
- As a numerical example, assume that each register has four bits. Let the content of R1 be 1010 and the content of R2 be 1100. The exclusive-OR micro operation stated above symbolizes the following logic computation:

  1010 Content of R1
  1100 Content of R2
  0110 Content of R1 if P = 1
- The symbol **V** will be used to denote an **OR** micro operation and the symbol **Λ** to denote an **AND** micro operation. The complement micro operation is the same as the 1's complement and uses a bar on top of the symbol that denotes the register name.

List of Logic Microoperations

- There are 16 different logic operations that can be performed with two binary variables. They can be determined from all possible truth tables obtained with two binary 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
$F_{10} = y'$	$F \leftarrow \overline{B}$	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

Table : Sixteen Logic Microoperations

- In the following table, each of the 16 columns F_0 through F_{15} represents a truth table of one possible Boolean function for the two variables x and y. Note that the functions are determined from the 16 binary combinations that can be assigned to F.

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_{12}	F_{13}	F_{14}	F_{15}
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

Hardware Implementation for Logic Microoperations:

- 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.
- The following figure shows one stage of a circuit that generates the four basic logic mnicrooperations. It 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 S_1 and S_0 choose one of the data inputs of the multiplexer and direct its value to the output.

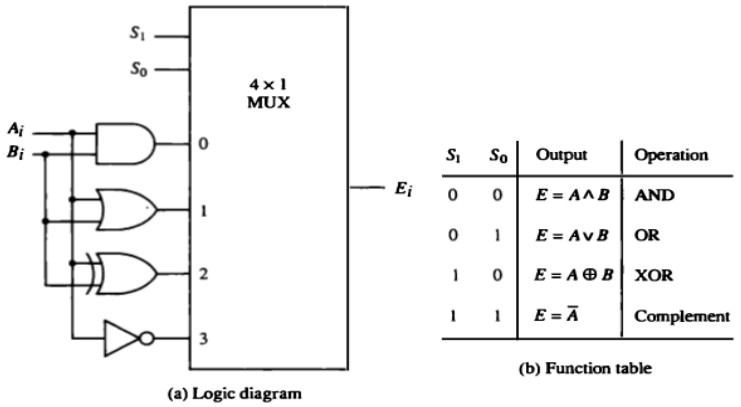


Figure : One stage of logic circuit

Applications of logic microoperations

1. selective-set

- The selective-set operation sets to 1 the bits in register A where there are corresponding 1's in register B. It does not affect bit positions that have 0's in B. The following numerical example clarifies this operation:

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

2. selective-complement

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

For example:
1010 A before
1100 B (logic operand)
0110 A after

3. Selective-clear

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

For example:
1010 A before
1100 B (logic operand)
0010 A after

4. mask

- The mask operation is similar to the selective-clear operation except that the bits of A are cleared only where there are corresponding 0'sin B. The mask operation is an AND micro operation as seen from the following numerical example:

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

5. Insert

- The insert operation inserts a new value into a group of bits.
- This is done by first masking the bits and then ORing them with the required value.
- For example, an A register contains eight bits, 0110 1010. To replace the four leftmost bits by the value 1001.

  First mask the four unwanted bits.

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

  Now insert the new value.

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

**The mask operation is an AND microoperation and the insert operation is an OR microoperation.

6. clear

- The clear operation compares the words in A and B and produces an all 0's result, if the two numbers are equal.
- This operation is achieved by an exclusive-OR microoperation as shown by the following example.

1010	A
1010	B
00	A ←A⊕B
01	

Shift Microoperations:

- Shift rnicrooperations are used for serial transfer of data.
- They are also used in conjunction with arithmetic, logic, and other data-processing operations.
- The contents of a register can be shifted to the left or the right.
- There are three types of shifts:
  1. Logical Shift
  2. Circular Shift
  3. Arithmetic Shift

1. Logical Shift

- A logical shift is one that transfers 0 through the serial input to fill the vacancy created by shift operation.
- The symbols *shl* is used for logical shift-left rnicro operation.

Shr is used for shift-right rnicro operation.

shl: Example: R1 ←shl R1

the above statement left shifts the content of register R1 by 1-bit.

Example: R1 = 0110

After performing shift left, R1 has the content 1100

shr: Example: R1 ←shr R1

the above statement right shifts the content of register R1 by 1-bit.

Example: R1 = 1100

After performing shift left, R1 has the content 0110

2. Circular Shift

- The circular shift also known as a rotate operation.
- It circulates the bits of the register around the two ends without loss of information.
- The symbols *cil* used for logical circular shift-left rnicrooperation.

cir used for circular shift-right rnicrooperation.

cil: Example: R1 ←cil R1

the above statement specifies circular shift left of the content of register R1.

Here all the bits are shifted one bit position to LEFT, the Left most bit (MSB) was circulated to Right most bit (LSB).

Example: R1 = 1011

After performing circular shift left, R1 has the content 0111

cir: Example: R1 ←cir R1

the above statement specifies circular shift right of the content of register R1.

Here all the bits are shifted one bit position to RIGHT, the Right most bit (LSB) was circulated to Left most bit (MSB).

Example: R1 = 1011

After performing circular shift right, R1 has the content 1101

3. Arithmetic Shift

- An arithmetic shift is a microoperation that shifts a signed binary number to the left or right.
- After the shift microoperation the sign of the number is to be restored.
- The symbols ash*l* used for logical arithmetic shift-left rnicrooperation.

ashr used forarithmetic shift-right rnicrooperation

ashl: Example: R1 ←ashl R1

the above statement specifies arithmetic shift left of the content of register R1

Here all the bits except the MSB, shift one bit position to LEFT.The second Left most bit was discarded and Right most bit (LSB) was loaded by 0.

Example: R1 = 1011

After performing arithmetic shift left, R1 has the content 1110

ashr: Example: R1 ←ashr R1

the above statement specifies arithmetic shift right of the content of register R1

Here all the bits shifted one bit position to RIGHT. The Left most bit (MSB) remains same.

Example: R1 = 1011

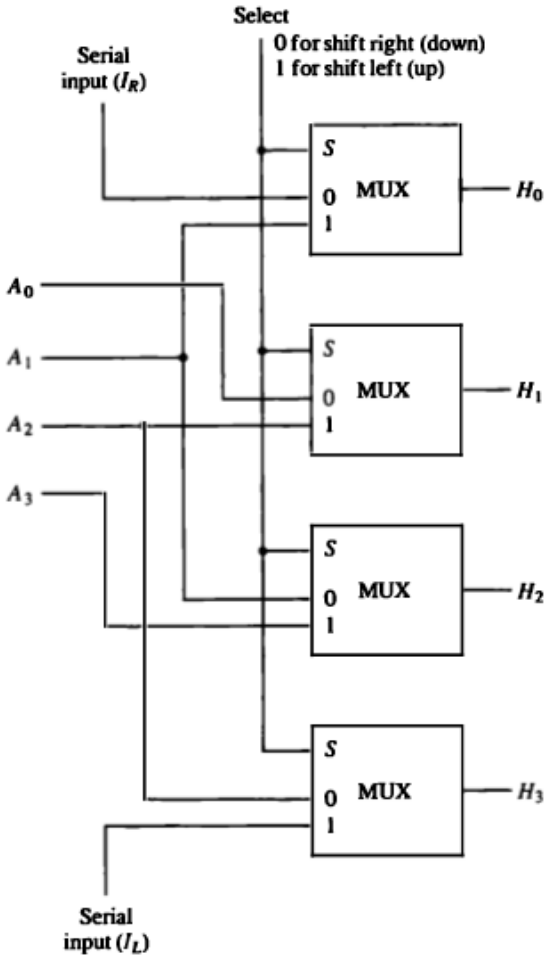
After performing arithmetic shift right, R1 has the content 1101

Symbolic designation	Description
R ← shl R	Shift-left register R
R ← shr R	Shift-right register R
R ← cil R	Circular shift-left register R
R ← cir R	Circular shift-right register R
R ← ashl R	Arithmetic shift-left R
R ← ashr R	Arithmetic shift-right R

Table : Shift Micro operations

Hardware Implementation for Shift Microoperations

- A combinational circuit shifter can be constructed with multiplexers as shown in figure.



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-bit combinational circuit shifter

** Here H_0 bit is considered as MSB and H_3 bit is considered as LSB.

- The 4-bit combinational circuit shifter uses four multiplexers, each of 2X1.
- The 4-bit shifter has four data inputs, A_0 through A_3, and four data outputs H_0 through H_3.
- There are two serial inputs, one for shift left (I_L) and the other for shift right (I_R).
- When the selection input S = 0, the input data are shifted right (down in the diagram).
- When S = 1, the input data are shifted left (up in the diagram).
- The above function table shows which input goes to each output after the shift.
- A shifter with n data inputs and n data outputs requires n multiplexers each of 2X1.

Arithmetic Logic Shift Unit

- Instead of having individual registers performing the microoperations directly, computer systems employ a number of storage registers connected to a common operational unit called an arithmetic logic unit, abbreviated ALU.

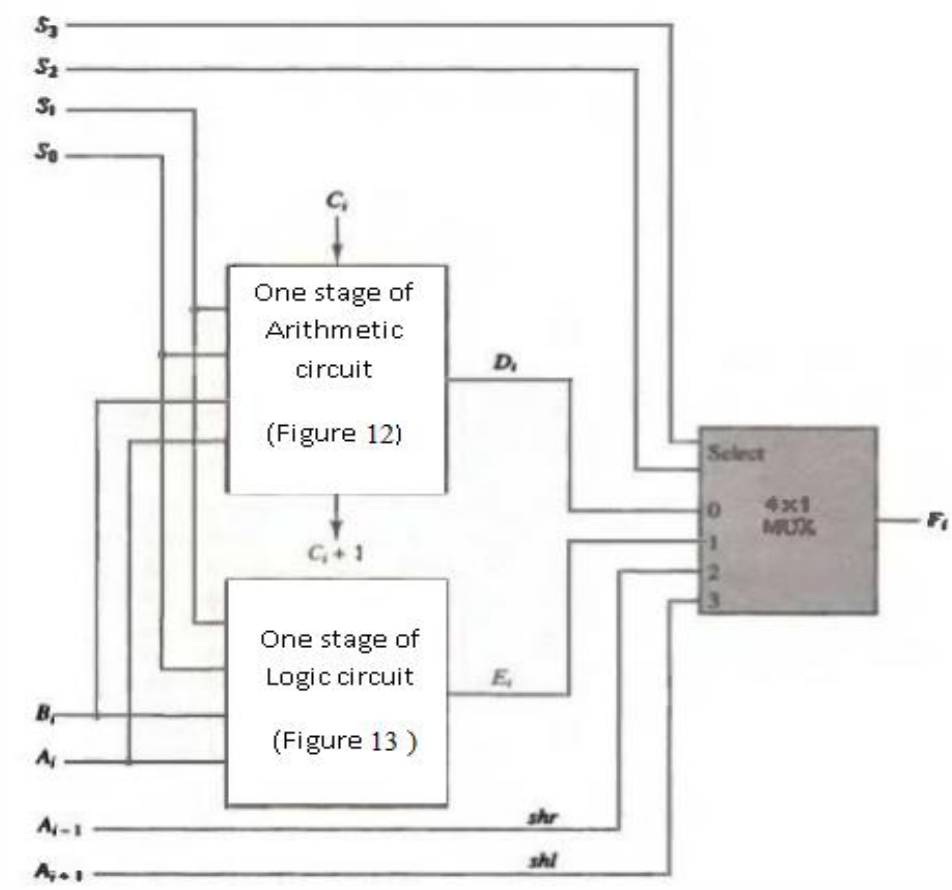


Figure : One stage of Arithmetic Logic Shift Unit

- To perform a microoperation, the contents of specified registers are placed in the inputs of the 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.
- The arithmetic, logic, and shift circuits can be combined into one ALU with common selection variables.
- One stage of an arithmetic logic shift unit is shown in the following figure.
- Inputs A_i and B_i are applied to both the arithmetic and logic units.
- A particular microoperation is selected with inputs S_1 and S_0.
- A 4 x 1 multiplexer at the output chooses between an arithmetic output in D_i and a logic output in E_i.
- The data in the multiplexer are selected with inputs S_3 and S_2.The other two data inputs to the multiplexer receive inputs A_{i-1} for the shift-right operation and A_{i+1} for the shift-left operation.
- The circuit whose one stage is specified in the above figure provides eight arithmetic microoperation, four logic microoperations, and two shift microoperations.
- Each operation is selected with the five variables S_3, S_2, S_1, S_0 and C_{in} The input carry C_{in} is used for arithmetic operations only.
- The first eight are arithmetic microoperations, which are selected with $S_3S_2 = 00$.
- The next four are logic microoperations, which are selected with $S_3S_2 = 01$.
- The input carry has no effect during the logic microoperations and is marked with don't-care x.
- The last two operations are shift microoperations and are selected with $S_3S_2 = 10$ for shift right microoperation and S_3S_2=and 11 for shift left microoperation. The other three inputs have no effect on the shift.

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

Table: Function Table for Arithmetic Logic Shift Unit