

EXPERIMENT NO. 1

DATE:

COMPONENTS OF A COMPUTER

AIM : To assemble a computer system.

1. POWER SUPPLY



A power supply converts Ac power from an outlet into Dc power for the computer.

2. HEAT SINK FAN



Forces air over the heat sink.

3. VIDEO ADAPTER CARD



Provides video capabilities.

4. NIC



Connects the computer to a network.

5. WIRELESS NIC



Connects the computer to a wireless network.

6. OPTICAL DRIVE



Reads digital video disks.

7. FLOPPY DRIVE



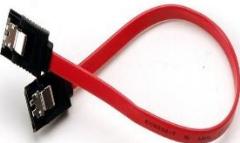
Reads and writes data to a floppy disk.

8. HARD DISK DRIVE



Reads and writes data to a fixed disk.

9. SATA CABLE



Connects SATA drive to the motherboard.

10. MOTHERBOARD



The main printed circuit board in a computer.

11. PATA CABLE



Connects drives to the motherboard.

12. RAM



Temporary memory used by the CPU.

EXPERIMENT NO. 2

DATE:

ASSEMBLING OF A COMPUTER SYSTEM

AIM : To assemble a computer system.

PROCEDURE :

- The assembling of the computer system is exactly the opposite of disassembling operation. Before starting assembling the computer system, make sure you have the screws and a screwdriver for those.
- The first step for assembling the computer system starts with mounting the processor on the processor socket of the motherboard. To mount the process, you don't need to apply any force. The special ZIF (zero insertion force) sockets are usually used to prevent any damage to the processor pins. Once the processor is mounted, the heat sink will be attached on top of the processor. The CPU fan is also attached on top of the heat sink.
- Now the motherboard is to be fixed vertically in the tower case and the screws are fixed from behind of the motherboard.
- Now line up the power supply at the top back end of the cabinet and screw it. The power connectors for motherboard power supply and CPU fan power supply are to be connected. If the cabinet cooling FAN is required then it is to be screwed at the back end grill of the cabinet and its power connector is to be connected from SMPS.
- Install the CD/DVD drives at the top front end of the cabinet and screw it. Install the Hard disk drive and floppy disk drive below CD/DVD drive and screw it. Make sure once screwed there is no vibration in either of the CD/DVD, Hard disk or Floppy disk drives.
- Now select the appropriate data cable and connect one end of the cable to its drive socket and another end at its appropriate connector on the motherboard. For SATA hard disk drive or CD/DVD drives use SATA cable and its power cable, else use IDE data cable. Do the proper jumper settings as per the usage requirement.
- It is time now to mount the memory modules on the motherboard by aligning the RAM to its socket on the motherboard and press it downward. Make sure the side tab are fixed into the RAM notch. If not, you may still have to press a bit.

- Install the internal cards to its socket and attach the cables or power cable to it. The selection of right socket or slot is required as per the type of socket.
- Cover the tower by placing it and pressing towards front side and screw it.
- Connect the external devices with CPU at its appropriate socket. It includes mouse and keyboard at PS2 or USB connectors. Monitor at the video output socket. Connect the power cable to the back of tower in SMPS. Plug in the power cable to the electric board.

RESULT : The computer system was successfully assembled.

EXPERIMENT NO. 3

DATE:

DISASSEMBLING OF A COMPUTER SYSTEM

AIM : To disassemble a computer system.

PROCEDURE :

- **Detach the power cable:**

The disassembling of the computer system starts with externally connected device detachment. Make sure the computer system is turned off, if not then successfully shut down the system and then start detaching the external devices from the computer system. It includes removing the power cable from electricity switchboard, then remove the cable from SMPS (switch mode power supply) from the back of the CPU Cabinet. Do not start the disassembling without detaching the power cable from the computer system. Now remove the remaining external devices like keyboard, mouse, monitor, printer or scanner from the back of CPU cabinet.

- **Remove the Cover:**

The standard way of removing tower cases used to be to undo the screws on the back of the case, slide the cover back about an inch and lift it off. The screwdrivers as per the type of screw are required to do the task.

- **Remove the adapter cards:**

Make sure if the card has any cables or wires that might be attached and decide if it would be easier to remove them before or after you remove the card. Remove the screw if any, that holds the card in place. Grab the card by its edges, front and back, and gently rock it lengthwise to release it.

- **Remove the drives:**

Removing drives is easier. There can be possibly three types of drives present in your computer system, Hard disk drive, CD/DVD/Blue-ray drives, floppy disk drives (almost absolute now a day). They usually have a power connector and a data cable attached from the device to a controller card or a connector on the motherboard. CD/DVD/Blue Ray drive may have an analog cable connected to the sound card for direct audio output.

The power may be attached using one of two connectors, a Molex connector or a Berg connector for the drive. The Molex connector may require to be wiggled

slightly from side to side and apply gentle pressure outwards. The Berg connector may just pull out or it may have a small tab which has to be lifted with a screwdriver.

Now Pull data cables off from the drive as well as motherboard connector. The hard disk drive and CD/DVD drives have two types of data cables. IDE and SATA cables. The IDE cables need better care while being removed as it may cause the damage to drive connector pins. Gently wiggle the cable sideways and remove it. The SATA cables can be removed easily by pressing the tab and pulling the connector straight back.

Now remove the screws and slide the drive out the back of the bay.

- **Remove the memory module:**

Memory modules are mounted on the motherboard as the chips that can be damaged by manual force if applied improperly. Be careful and handle the chip only by the edges.

- **Remove the power supply:**

The power supply is attached into tower cabinet at the top back end of the tower. Make sure the power connector is detached from the switchboard. Start removing the power connector connected to motherboard including CPU fan power connector, cabinet fan, the front panel of cabinet power buttons and all the remaining drives if not detached yet.

Now remove the screws of SMPS from the back of the cabinet and the SMPS can be detached from the tower cabinet.

- **Remove the motherboard:**

Before removing all the connectors from the motherboard, make sure u memorize the connectors for assembling the computer if required, as that may require connecting the connectors at its place. Remove the screws from the back of the motherboard and you will be able to detach it from the cabinet. Now remove the CPU fan from the motherboard. The heat sink will be visible now which can be removed by the pulling the tab upward. Finally, the processor is visible now, which can be removed by the plastic tab which can be pulled back one stretching it side way.

RESULT : The computer system was successfully disassembled.

EXPERIMENT NO. 4a

DATE:

ADDITION OF TWO 8-BIT NUMBERS

AIM : Adding two 8 Bit Numbers.

SOFTWARE USED : emu 8086

CODE :

```
data segment
```

```
a db 15h
```

```
b db 12h
```

```
data ends
```

```
code segment
```

```
assume cs: code, ds: data
```

```
start:
```

```
    mov ax, data
```

```
    mov ds, ax
```

```
    mov al, a
```

```
    mov bl, b
```

```
    add al, bl
```

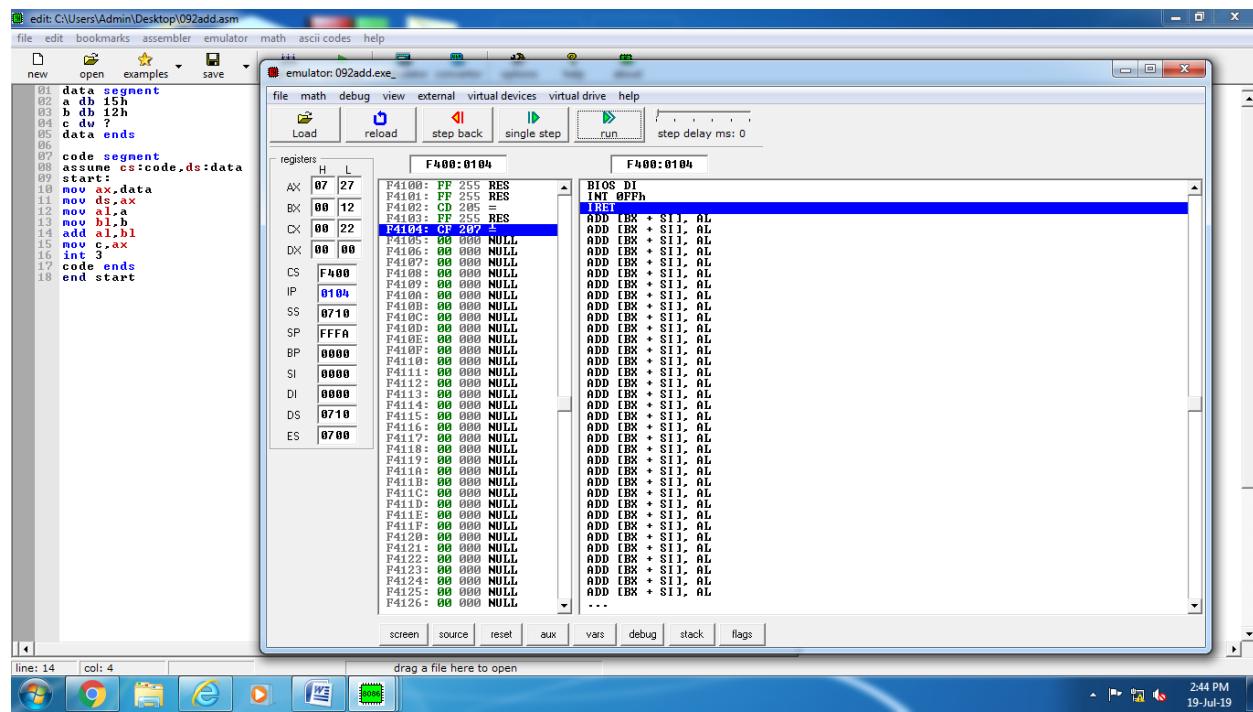
```
    mov c, ax
```

```
    int 3
```

```
code ends
```

```
end start
```

OUTPUT :



RESULT : Addition of two 8 bit numbers was successfully done.

EXPERIMENT NO. 4b

DATE:

SUBTRACTION OF TWO 8-BIT NUMBERS

AIM : Subtracting two 8 Bit Numbers.

SOFTWARE USED : emu 8086

CODE :

```
data segment
```

```
a db 15h
```

```
b db 12h
```

```
data ends
```

```
code segment
```

```
assume cs: code, ds: data
```

```
start:
```

```
    mov ax, data
```

```
    mov ds, ax
```

```
    mov al, a
```

```
    mov bl, b
```

```
    sub al, bl
```

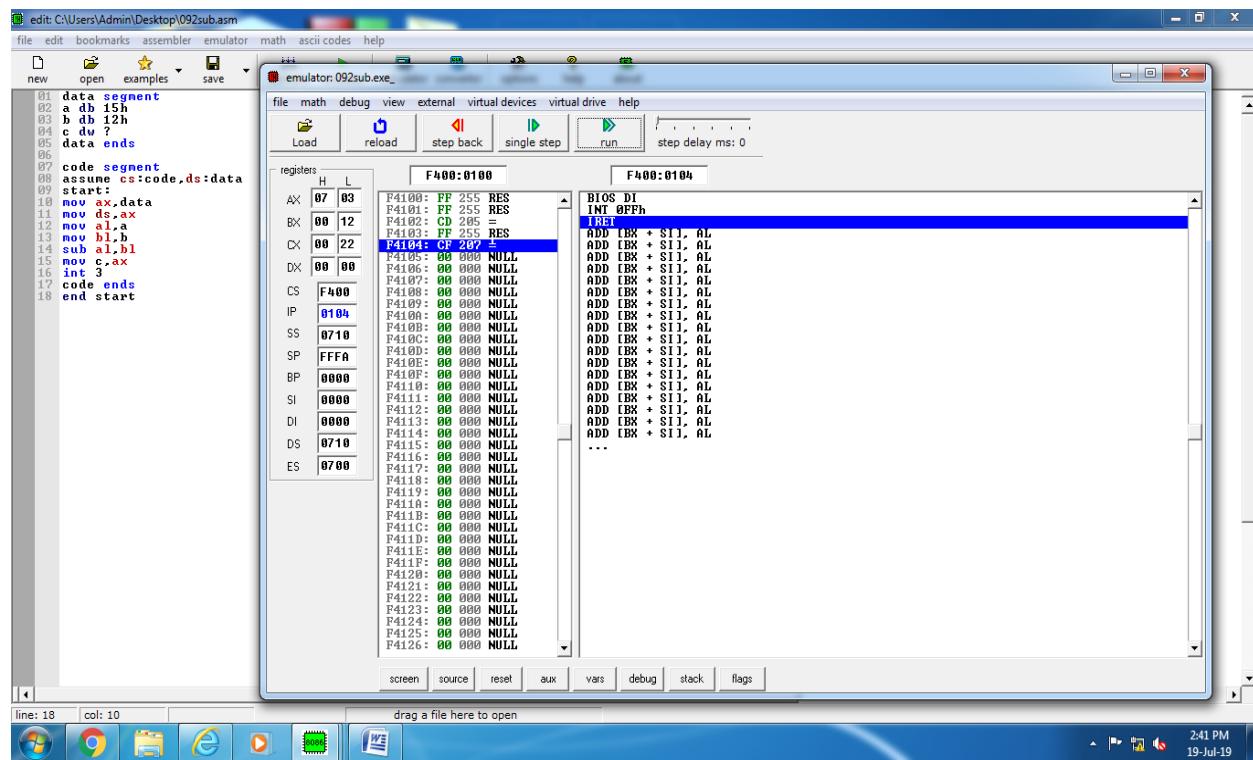
```
    mov c, ax
```

```
    int 3
```

```
code ends
```

```
end start
```

OUTPUT :



RESULT : Subtraction of two 8 bit numbers was successfully done.

EXPERIMENT NO. 5a

DATE:

ADDITION OF TWO 16-BIT NUMBERS

AIM : Adding two 16 Bit Numbers.

SOFTWARE USED : emu 8086

CODE :

```
data segment
```

```
N1 dw 4004h
```

```
N2 dw 1001h
```

```
data ends
```

```
code segment
```

```
assume cs: code, ds: data
```

```
start:
```

```
    mov ax, data
```

```
    mov ds, ax
```

```
    mov ax, N1
```

```
    mov bx, N2
```

```
    add ax, bx
```

```
    mov Res, ax
```

```
    int 3
```

```
code ends
```

```
end start
```

OUTPUT :

The screenshot shows the assembly code and the state of the CPU registers during the execution of the program.

Assembly Code:

```
01 DATA SEGMENT
02     M1 DW 40004h
03     M2 DW 10001h
04     RES DW ?
05 DATA ENDS
06 CODE SEGMENT
07     ASSUME CS:CODE, DS:DATA
08     START: MOU AX, DATA
09     MOU DS, AX
10     MOU AX, M1
11     MOU BX, M2
12     ADD AX, BX
13     MOU RES, AX
14     INT 3
15     CODE ENDS
16 END START
```

Registers:

	H	L
AX	50	05
BX	10	01
CX	00	22
DX	00	00
CS	F400	
IP	0104	
SS	0710	
SP	FFF8	
BP	0000	
SI	0000	
DI	0000	
DS	0710	
ES	0700	

Memory Dump:

	F400:0104	F400:0104
P4100:	FF 255 RES	BIOS DI
P4101:	FF 255 RES	INT 0FFh
P4102:	FF 255 RES	
P4103:	FF 255 RES	
P4104:	CF 207 ±	ADD BX + SI1, AL
P4105:	00 000 NULL	ADD BX + SI1, AL
P4106:	00 000 NULL	ADD BX + SI1, AL
P4107:	00 000 NULL	ADD BX + SI1, AL
P4108:	00 000 NULL	ADD BX + SI1, AL
P4109:	00 000 NULL	ADD BX + SI1, AL
P410A:	00 000 NULL	ADD BX + SI1, AL
P410B:	00 000 NULL	ADD BX + SI1, AL
P410C:	00 000 NULL	ADD BX + SI1, AL
P410D:	00 000 NULL	ADD BX + SI1, AL
P410E:	00 000 NULL	ADD BX + SI1, AL
P410F:	00 000 NULL	ADD BX + SI1, AL
P4110:	00 000 NULL	ADD BX + SI1, AL
P4111:	00 000 NULL	ADD BX + SI1, AL
P4112:	00 000 NULL	ADD BX + SI1, AL
P4113:	00 000 NULL	ADD BX + SI1, AL
P4114:	00 000 NULL	ADD BX + SI1, AL
P4115:	00 000 NULL	ADD BX + SI1, AL
P4116:	00 000 NULL	ADD BX + SI1, AL
P4117:	00 000 NULL	ADD BX + SI1, AL
P4118:	00 000 NULL	ADD BX + SI1, AL
P4119:	00 000 NULL	ADD BX + SI1, AL
P411A:	00 000 NULL	ADD BX + SI1, AL
P411B:	00 000 NULL	ADD BX + SI1, AL
P411C:	00 000 NULL	ADD BX + SI1, AL
P411D:	00 000 NULL	ADD BX + SI1, AL
P411E:	00 000 NULL	ADD BX + SI1, AL
P411F:	00 000 NULL	ADD BX + SI1, AL
P4120:	00 000 NULL	ADD BX + SI1, AL
P4121:	00 000 NULL	ADD BX + SI1, AL
P4122:	00 000 NULL	ADD BX + SI1, AL
P4123:	00 000 NULL	ADD BX + SI1, AL
P4124:	00 000 NULL	ADD BX + SI1, AL
P4125:	00 000 NULL	ADD BX + SI1, AL
P4126:	00 000 NULL	...

RESULT : Addition of two 16 bit numbers was successfully done.

EXPERIMENT NO. 5b

DATE:

SUBTRACTION OF TWO 16-BIT NUMBERS

AIM : Subtracting two 16 Bit Numbers.

SOFTWARE USED : emu 8086

CODE :

```
data segment
```

```
N1 dw 4004h
```

```
N2 dw 1001h
```

```
data ends
```

```
code segment
```

```
assume cs: code, ds: data
```

```
start:
```

```
    mov ax, data
```

```
    mov ds, ax
```

```
    mov ax, N1
```

```
    mov bx, N2
```

```
    sub ax, bx
```

```
    mov Res, ax
```

```
    int 3
```

```
code ends
```

```
end start
```

OUTPUT :

The screenshot shows a dual-pane assembly editor interface. The left pane displays assembly code:

```
01 DINT SEGMENT
02     N1 DW 4004h
03     N2 DW 1001h
04     RES DW ?
05     DATA ENDS
06
07 CODE SEGMENT
08     ASSUME CS:CODE, DS:DATA
09     START: MOU AX, DATA
10     MOU DS, AX
11     MOU BX, N2
12     MOU BP, AX
13     SUB AX, BX
14     MOU RES, AX
15     INT 3
16 END START
17 CODE ENDS
```

The right pane shows the CPU registers and memory dump. The registers pane lists:

	H	L
A ^X	30	03
B ^X	10	01
C ^X	00	22
D ^X	00	00
C ^S	F40B	
I ^P	0104	
S ^S	0710	
S ^F	FFFFA	
B ^F	0000	
S ^I	0000	
D ^I	0000	
D ^S	0710	
E ^S	0700	

The memory dump pane shows memory starting at address F400h:

Address	Value	Label
F400h	FF 25 55	RES
F401h	FF 25 55	RES
F402h	CD 31	INT 3
F403h	FF 25 55	RES
F404h	CF 20 ?	MOU AX, [BX]
F405h	00 00 00	NULL
F406h	00 00 00	NULL
F407h	00 00 00	NULL
F408h	00 00 00	NULL
F409h	00 00 00	NULL
F40Ah	00 00 00	NULL
F40Bh	00 00 00	NULL
F40Ch	00 00 00	NULL
F40Dh	00 00 00	NULL
F40Eh	00 00 00	NULL
F40Fh	00 00 00	NULL
F410h	00 00 00	NULL
F411h	00 00 00	NULL
F412h	00 00 00	NULL
F413h	00 00 00	NULL
F414h	00 00 00	NULL
F415h	00 00 00	NULL
F416h	00 00 00	NULL
F417h	00 00 00	NULL
F418h	00 00 00	NULL
F419h	00 00 00	NULL
F41Ah	00 00 00	NULL
F41Bh	00 00 00	NULL
F41Ch	00 00 00	NULL
F41Dh	00 00 00	NULL
F41Eh	00 00 00	NULL
F41Fh	00 00 00	NULL
F420h	00 00 00	NULL
F421h	00 00 00	NULL
F422h	00 00 00	NULL
F423h	00 00 00	NULL
F424h	00 00 00	NULL
F425h	00 00 00	NULL
F426h	00 00 00	NULL

At the bottom, there are buttons for screen, source, reset, aux, vars, debug, stack, and flags.

RESULT : Subtraction of two 16 bit numbers was successfully done.

EXPERIMENT NO. 6

DATE:

FACTORIAL OF A 8-BIT NUMBER

AIM : To find factorial of a 8 bit number.

SOFTWARE USED : emu 8086

CODE :

data segment

A db 5

data ends

code segment

assume cs: code, ds: data

start:

mov ax, data

mov ds, ax

mov ah, 00

mov al, a

L1 : Dec a

Mul A

mov cl, a

cmp cl, 01

Jnz L1

Mov ah, 4ch

int 21H

code ends

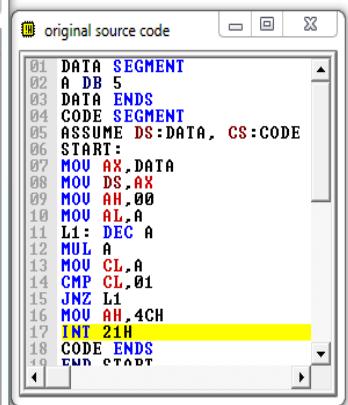
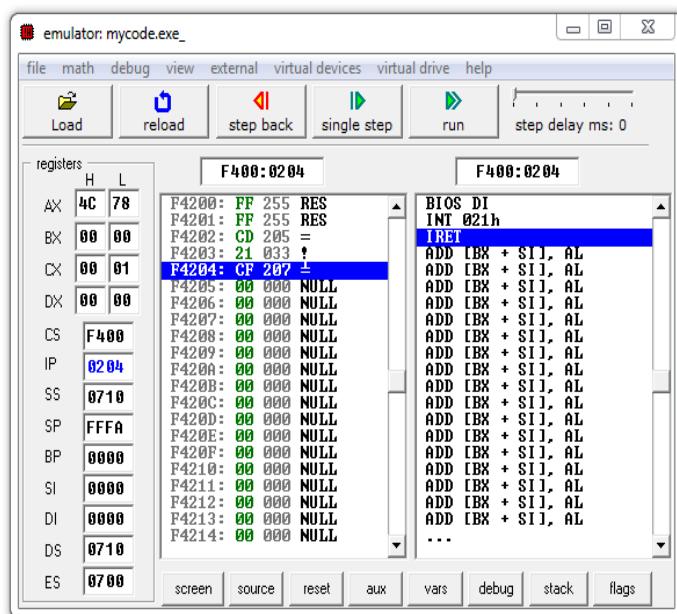
end start

OUTPUT :

```

01 DATA SEGMENT
02 A DB 5
03 DATA ENDS
04 CODE SEGMENT
05 ASSUME DS:DATA, CS:CODE
06 START:
07 MOU AX,DATA
08 MOU DS,AX
09 MOU AH,00
10 MOU AL,A
11 L1: DEC A
12 MUL A
13 MOU CL,A
14 CMP CL,01
15 JNZ L1
16 MOU AH,4CH
17 INT 21H
18 CODE ENDS
19 END START

```



RESULT : Factorial of a 8 bit number has been found.

EXPERIMENT NO. 7

DATE:

MULTIPLICATION OF TWO 8-BIT NUMBERS

AIM : Multiplying two 8 bit numbers.

SOFTWARE USED : emu 8086

CODE :

```
data segment
```

```
A db 09h
```

```
B db 02h
```

```
Res1 dw ?
```

```
data ends
```

```
code segment
```

```
assume cs: code, ds: data
```

```
start:
```

```
    mov ax, data
```

```
    mov ds, ax
```

```
    mov ax, 0000h
```

```
    mov bx, 0000h
```

```
    Mul b
```

```
    mov Res1, ax
```

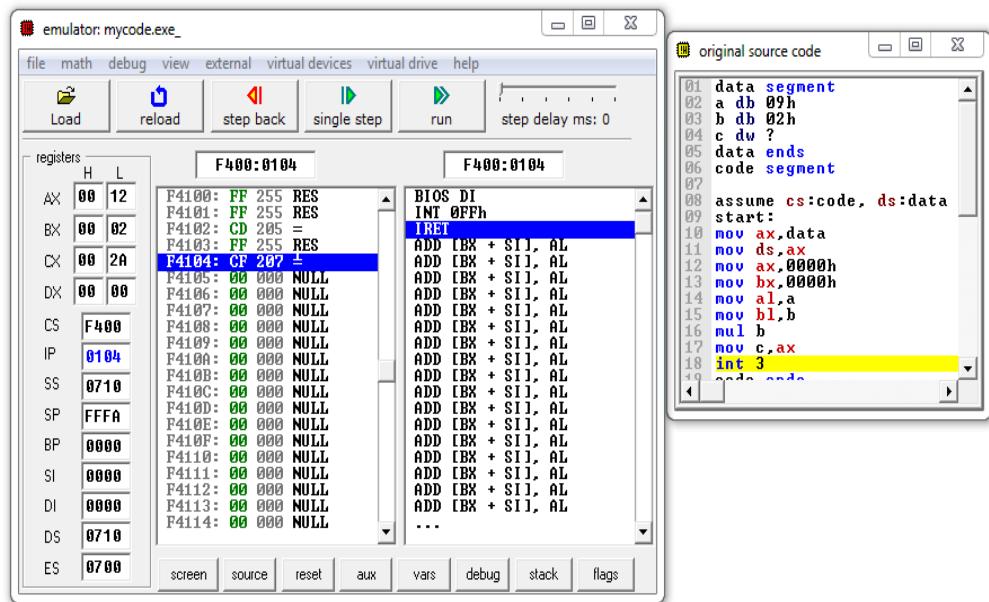
```
    int 3
```

```
code ends
```

```
end start
```

OUTPUT :

```
01 data segment
02     a db 09h
03     b db 02h
04     c dw ?
05 data ends
06 code segment
07
08 assume cs:code, ds:data
09 start:
10    mov ax,data
11    mov ds,ax
12    mov ax,0000h
13    mov bx,0000h
14    mov al,a
15    mov bl,b
16    mul b
17    mov c,ax
18    int 3
19    code ends
20 end start
```



RESULT : Multiplication of two 8 bit numbers has been done.

EXPERIMENT NO. 8

DATE:

FULL ADDER CIRCUIT

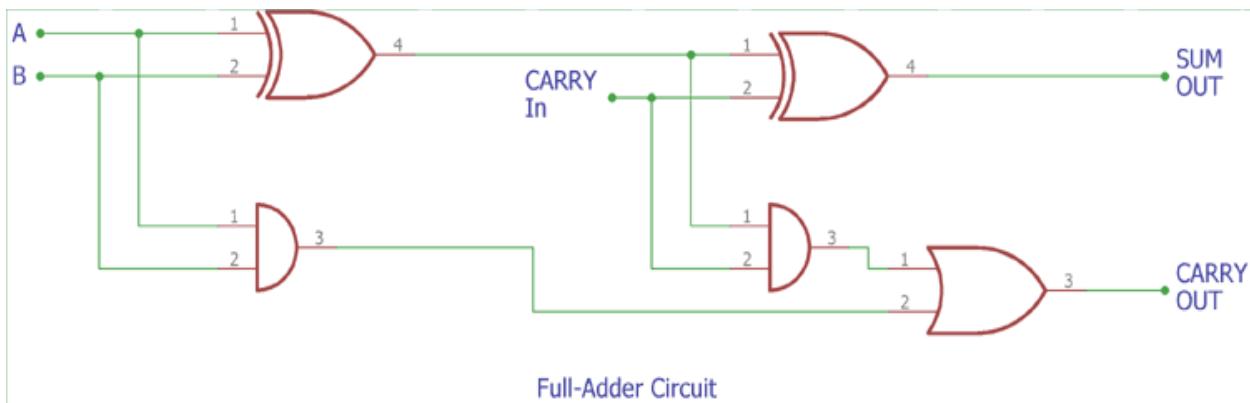
AIM : To design and implement a full adder circuit.

SOFTWARE USED : Logic gate simulator

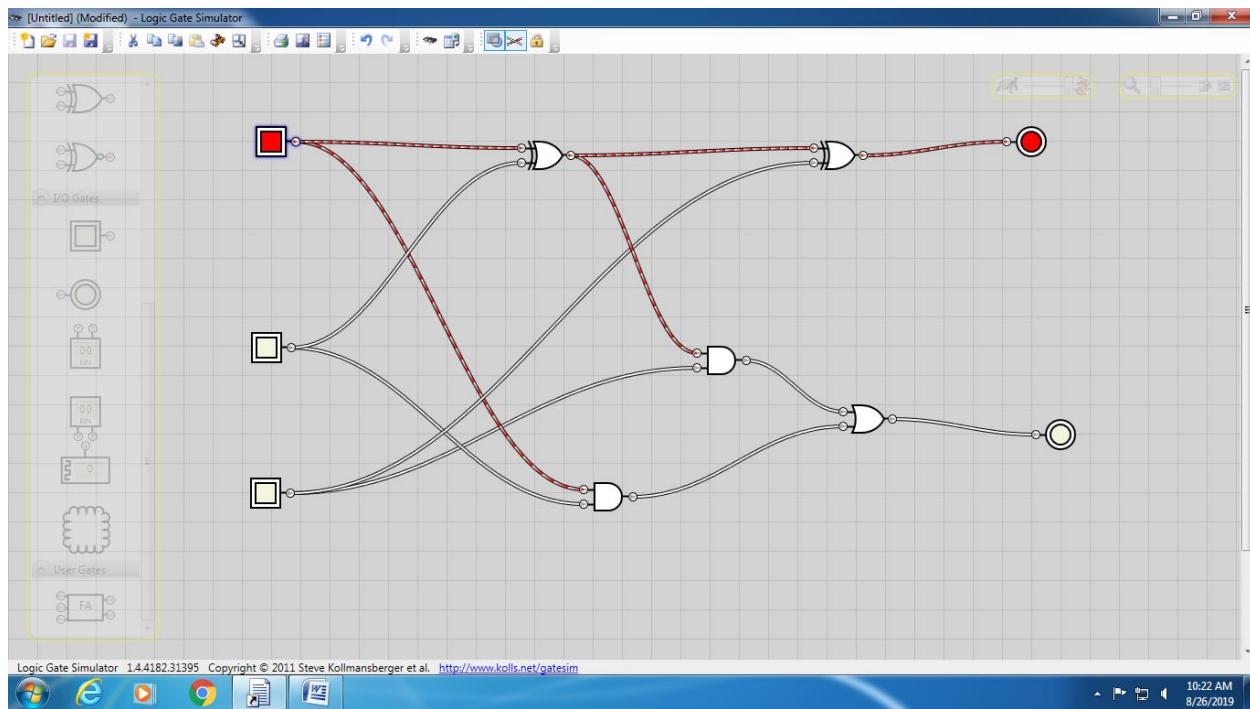
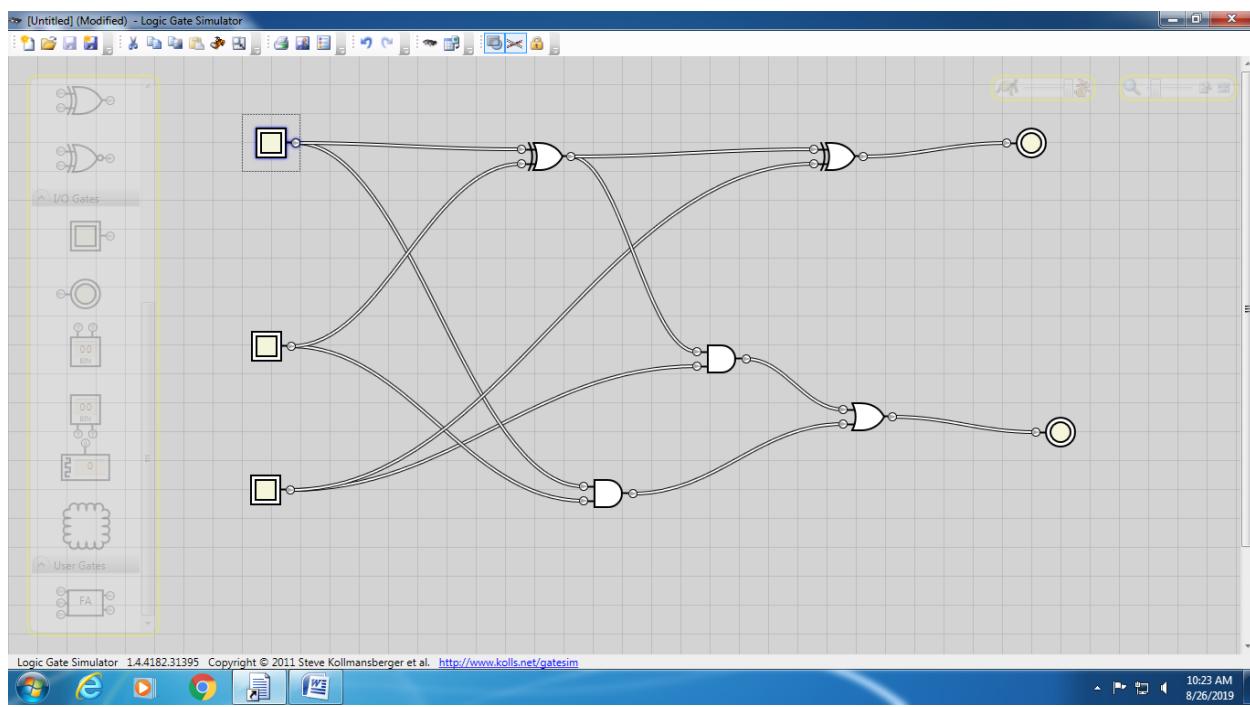
TRUTH TABLE :

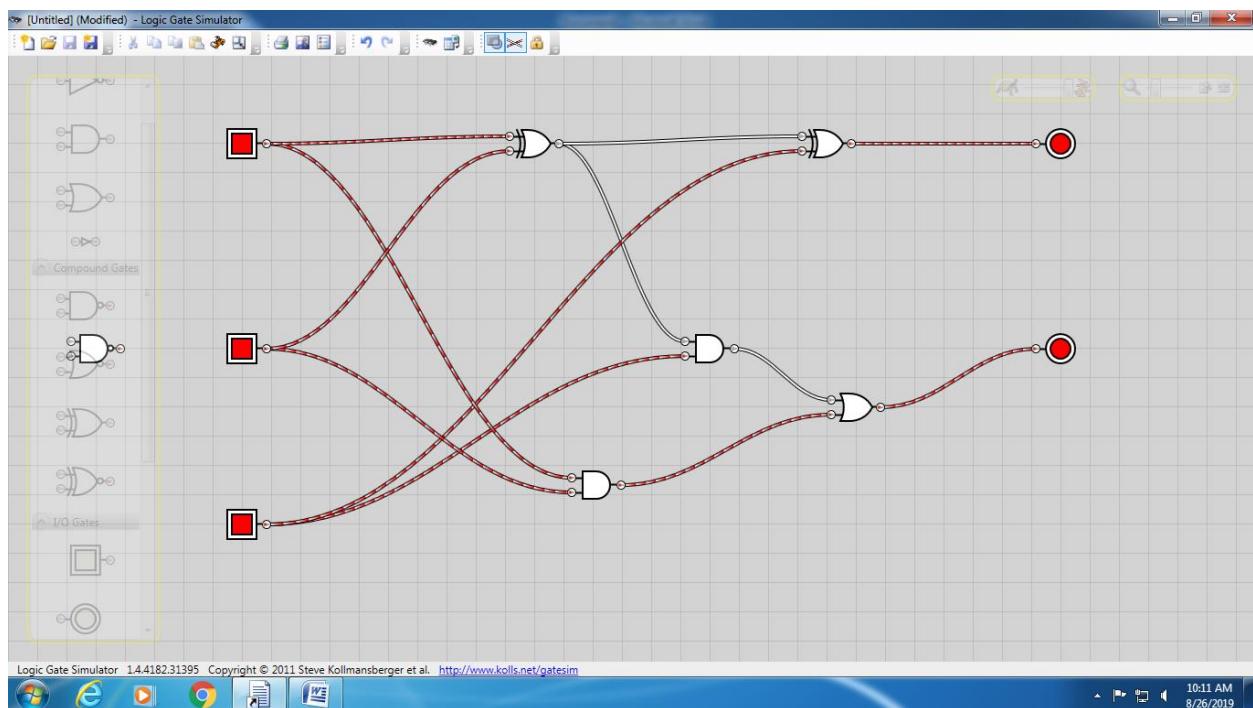
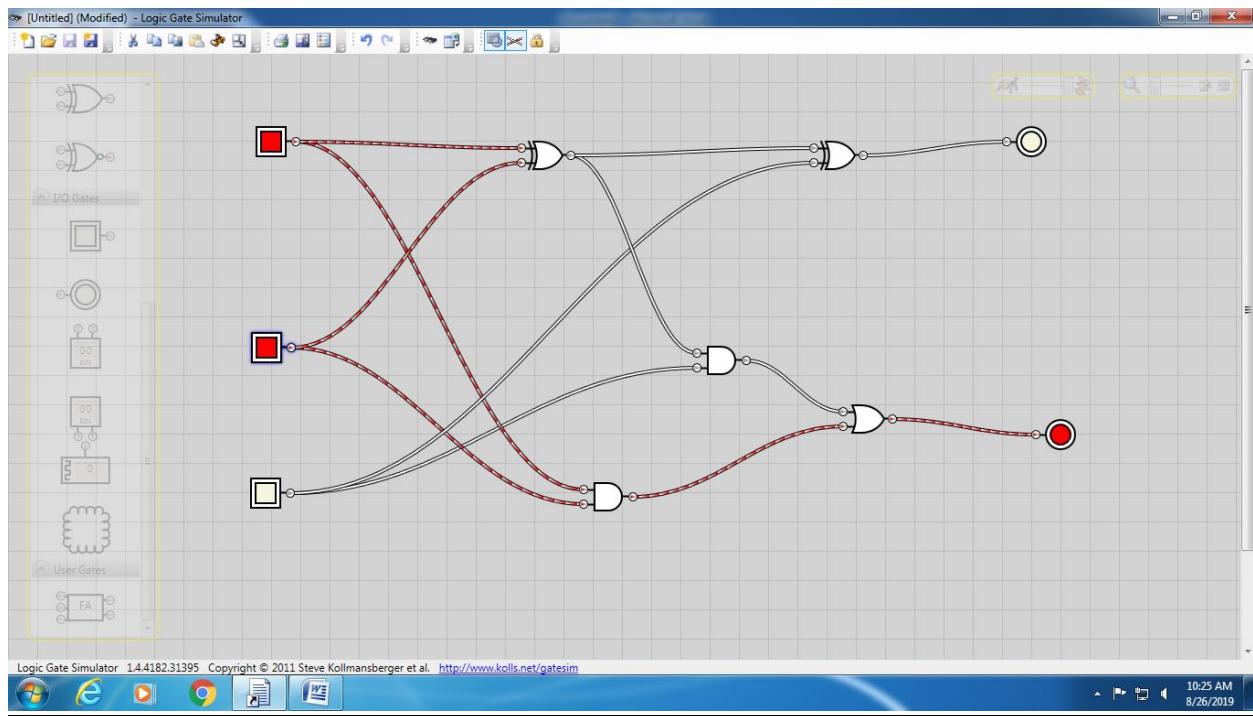
INPUTS			OUTPUT	
A	B	C-IN	C-OUT	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

CIRCUIT DIAGRAM :



OUTPUT :





RESULT : Here, we add three one bit binary numbers, two operands & a carry bit.

EXPERIMENT NO. 9

DATE:

HALF ADDER CIRCUIT

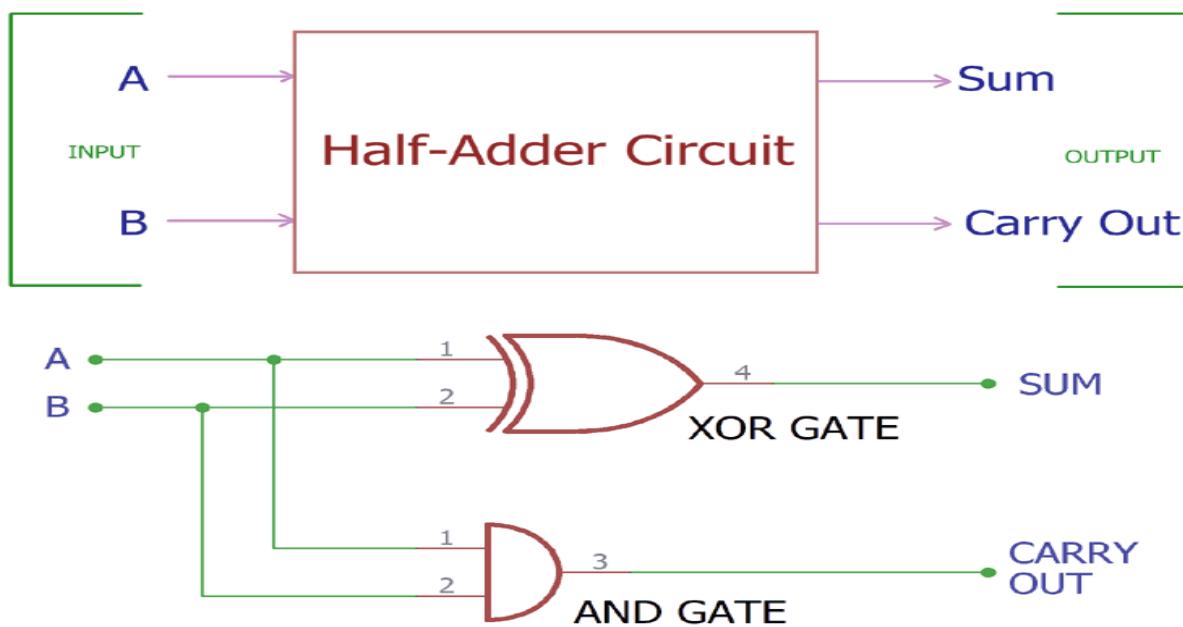
AIM : To design and implement a half adder circuit.

SOFTWARE USED : Logic gate simulator

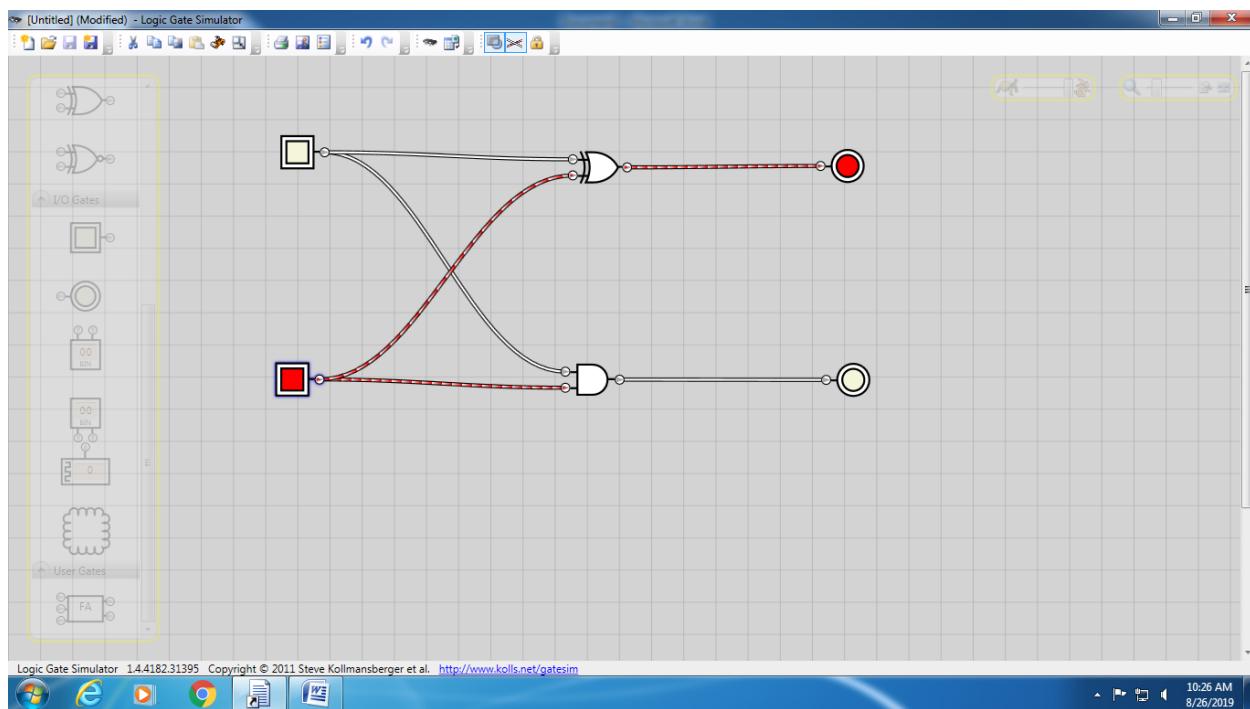
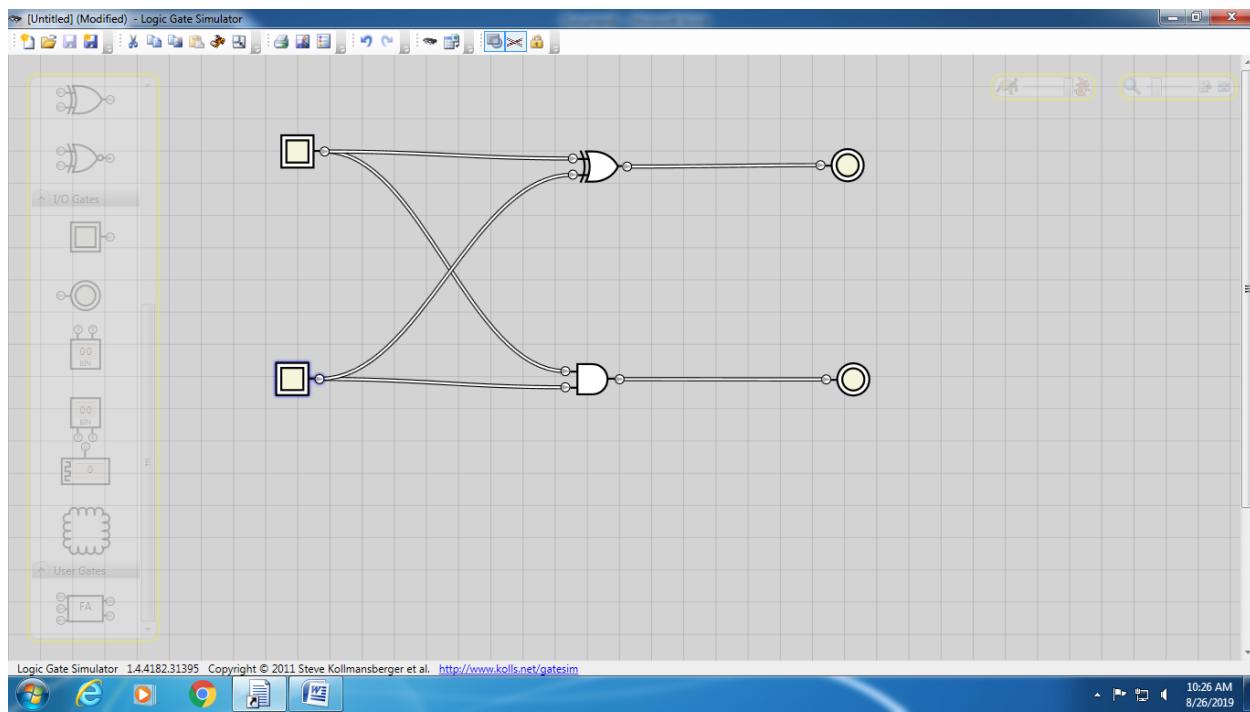
TRUTH TABLE :

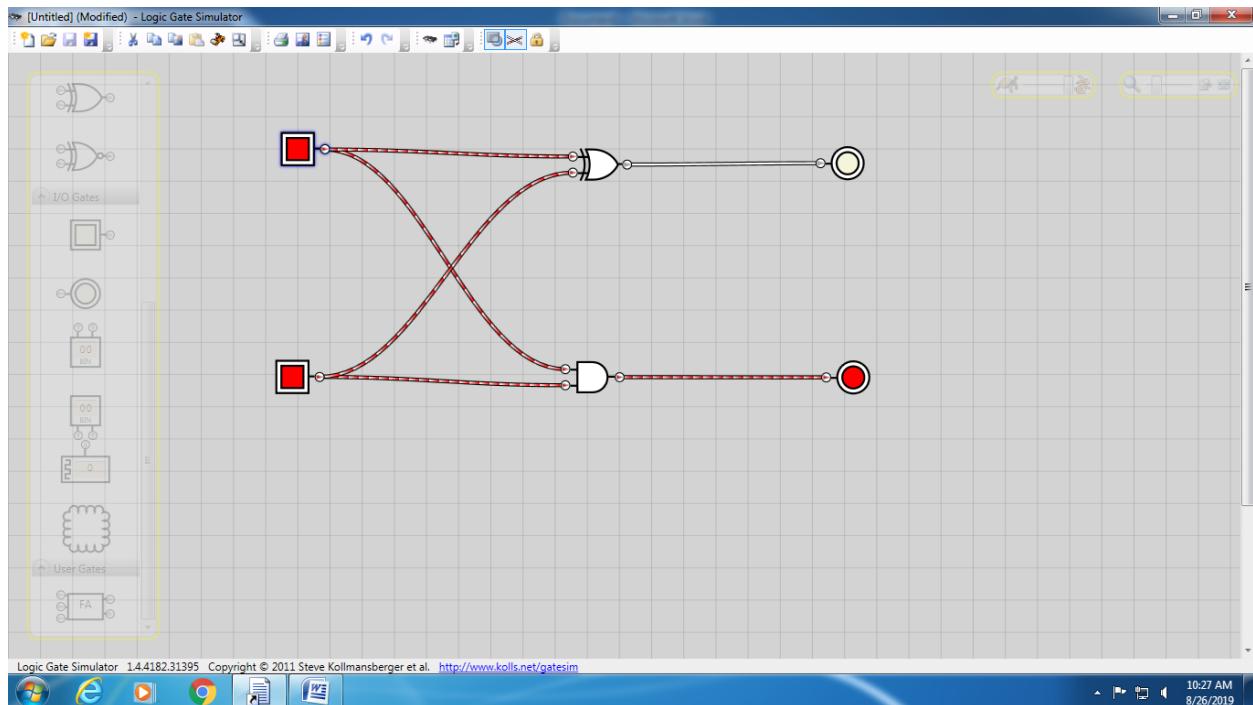
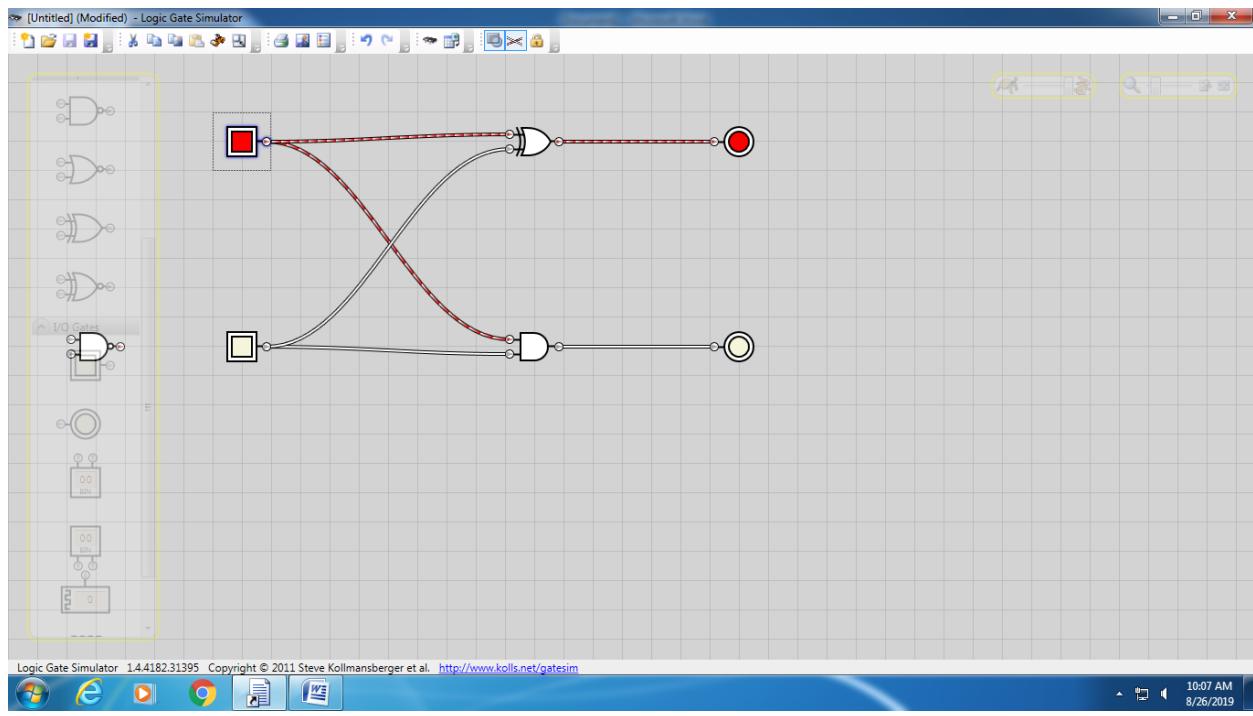
INPUTS		OUTPUTS	
A	B	SUM	CARRY
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

CIRCUIT DIAGRAM :



OUTPUT :





RESULT : Here, we add two single digit binary numbers & results in two digit out.

EXPERIMENT NO. 10

DATE:

RIPPLE CARRY ADDER

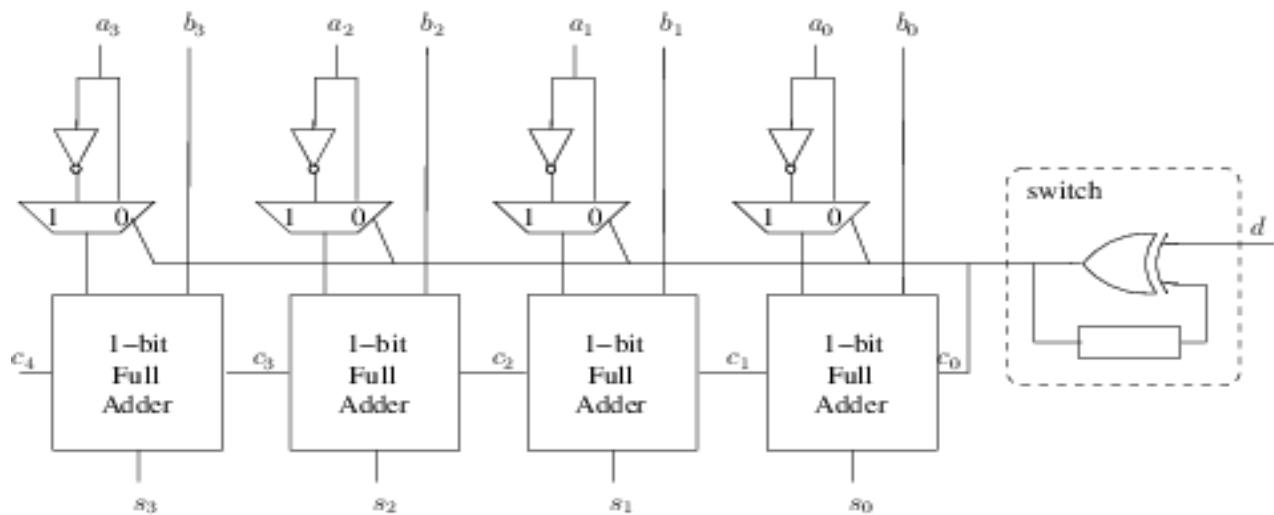
AIM : To design and implement a ripple carry adder.

SOFTWARE USED : Logic gate simulator

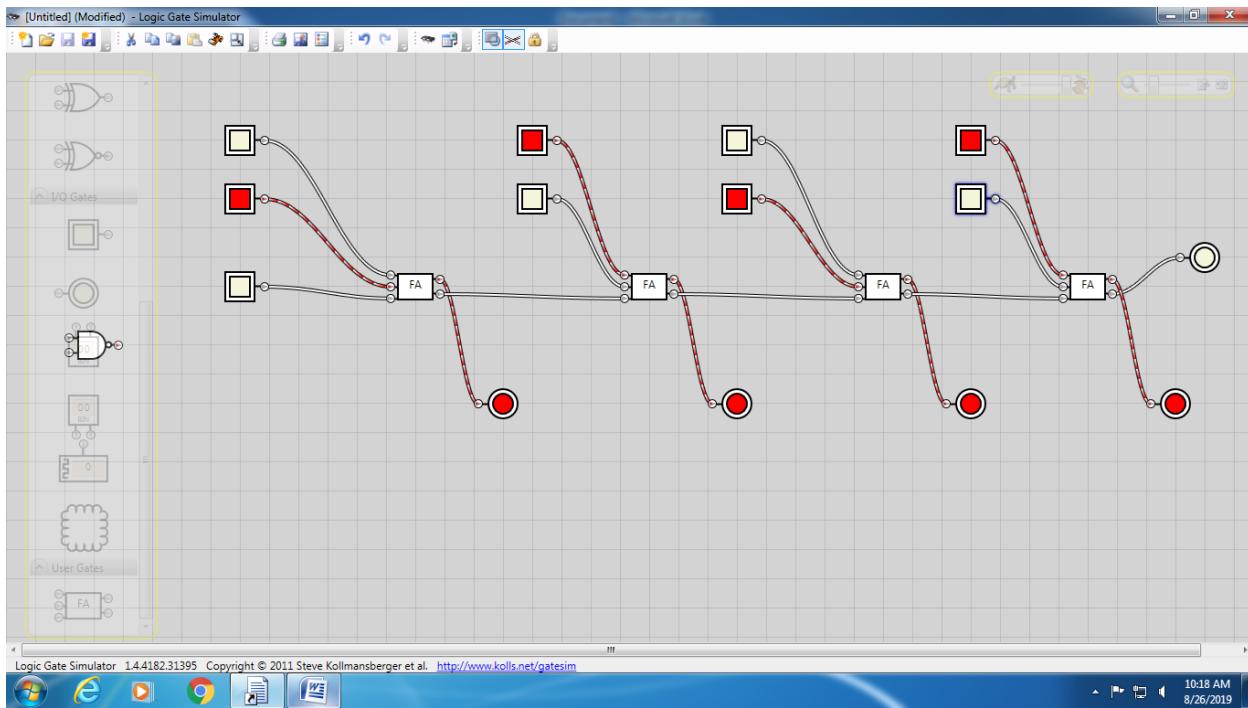
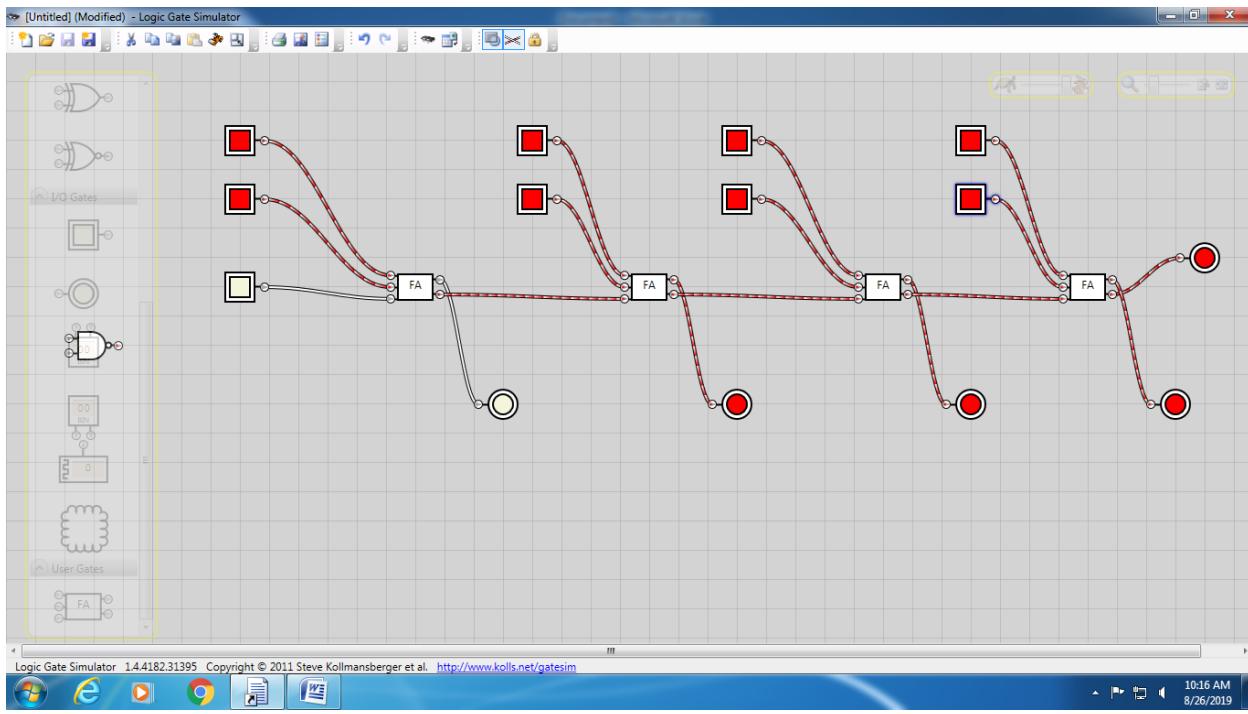
TRUTH TABLE :

A ₁	A ₂	A ₃	A ₄	B ₄	B ₃	B ₂	B ₁	S ₄	S ₃	S ₂	S ₁	carry	
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	1	0	0	1	0	0	0	0	0
1	0	0	0	1	0	0	0	0	0	0	0	0	1
1	0	1	0	1	0	1	0	0	1	0	0	0	1
1	1	0	0	1	1	0	0	1	0	0	0	0	1
1	1	1	0	1	1	1	0	1	1	0	0	0	1
1	1	1	1	1	1	1	1	1	1	1	0	0	1

CIRCUIT DIAGRAM :



OUTPUT :



RESULT : Hence, ripple carry adder was constructed in the simulator & it's characteristics were studied.

EXPERIMENT NO. 11

DATE:

CARRY LOOK AHEAD ADDER

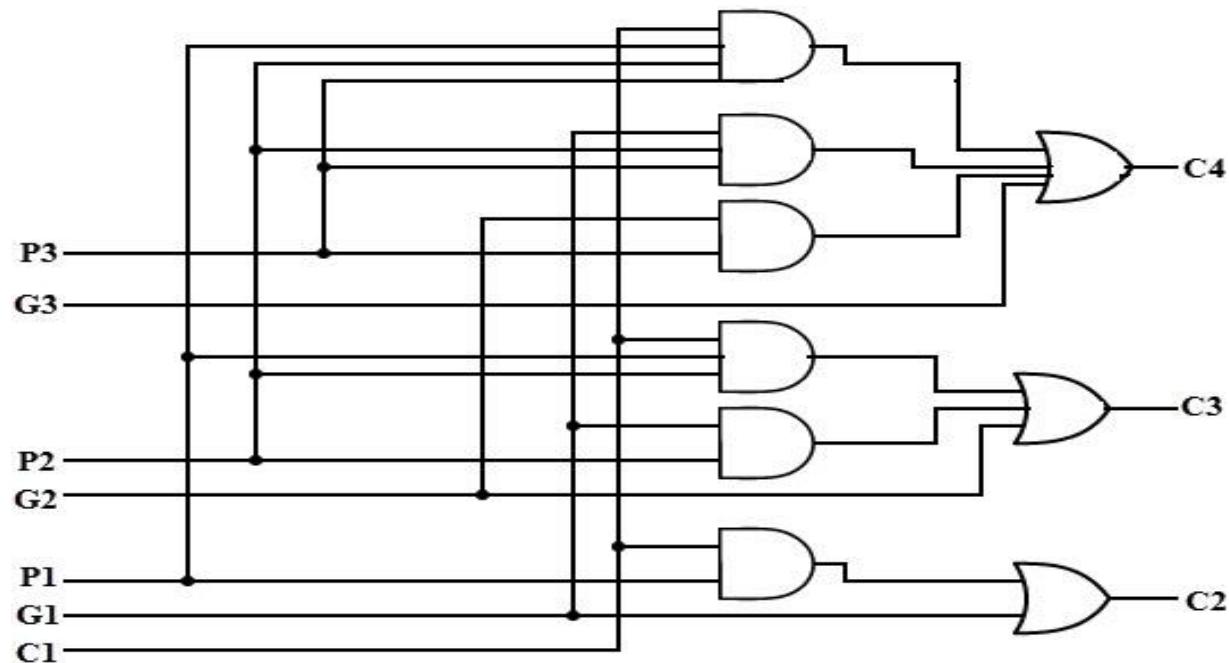
AIM : To design and implement a carry look ahead adder.

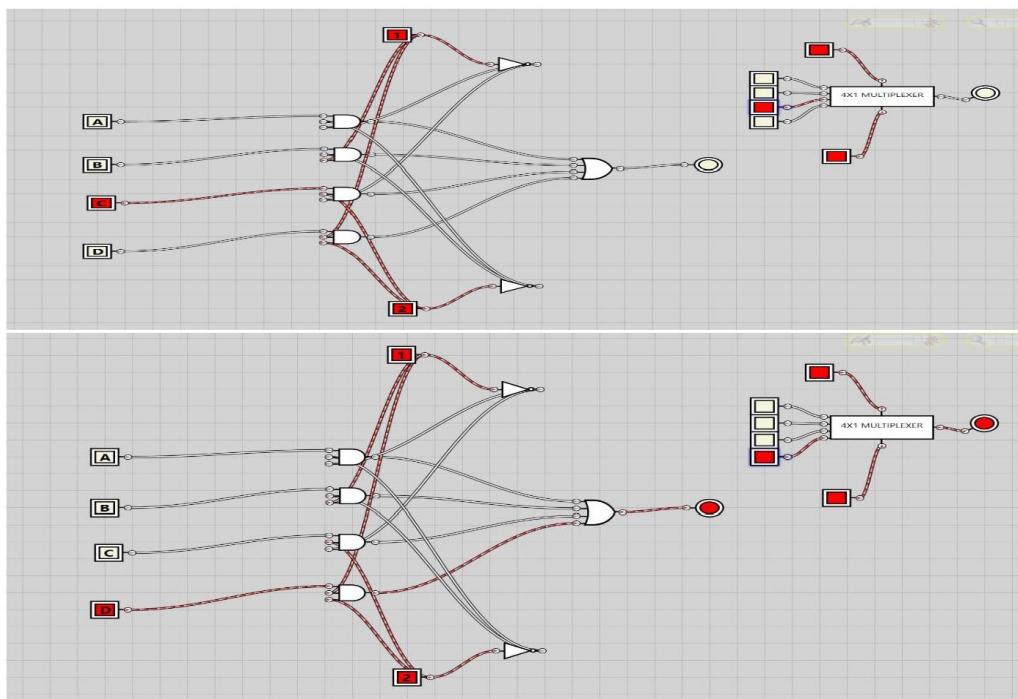
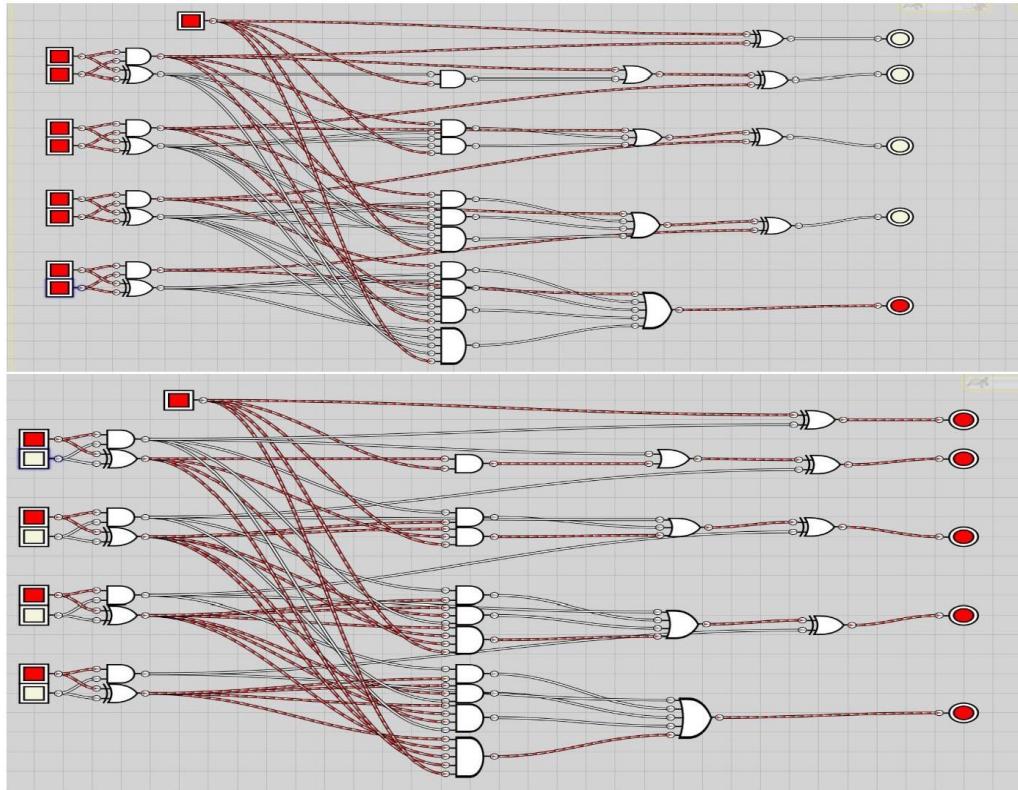
SOFTWARE USED : Logic gate simulator

TRUTH TABLE :

A	B	C _i	C _{i+1}	Condition
0	0	0	0	No carry generate
0	0	1	0	
0	1	0	0	
0	1	1	1	No carry propagate
1	0	0	0	
1	0	1	1	
1	1	0	1	Carry generate
1	1	1	1	

CIRCUIT DIAGRAM :





RESULT : Hence, carry look ahead adder was constructed & char. were studied.

EXPERIMENT NO. 12

DATE:

2-BIT MULTIPLIER

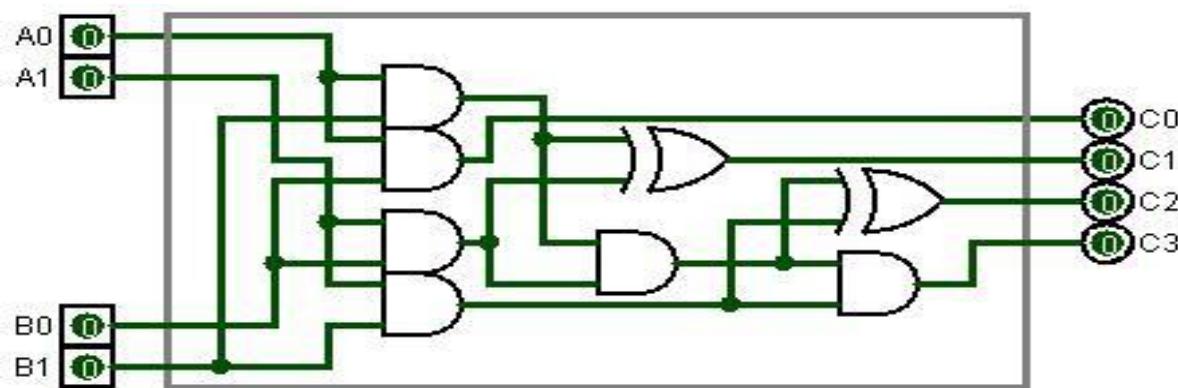
AIM : To design and implement a carry look ahead adder.

SOFTWARE USED : Logic gate simulator

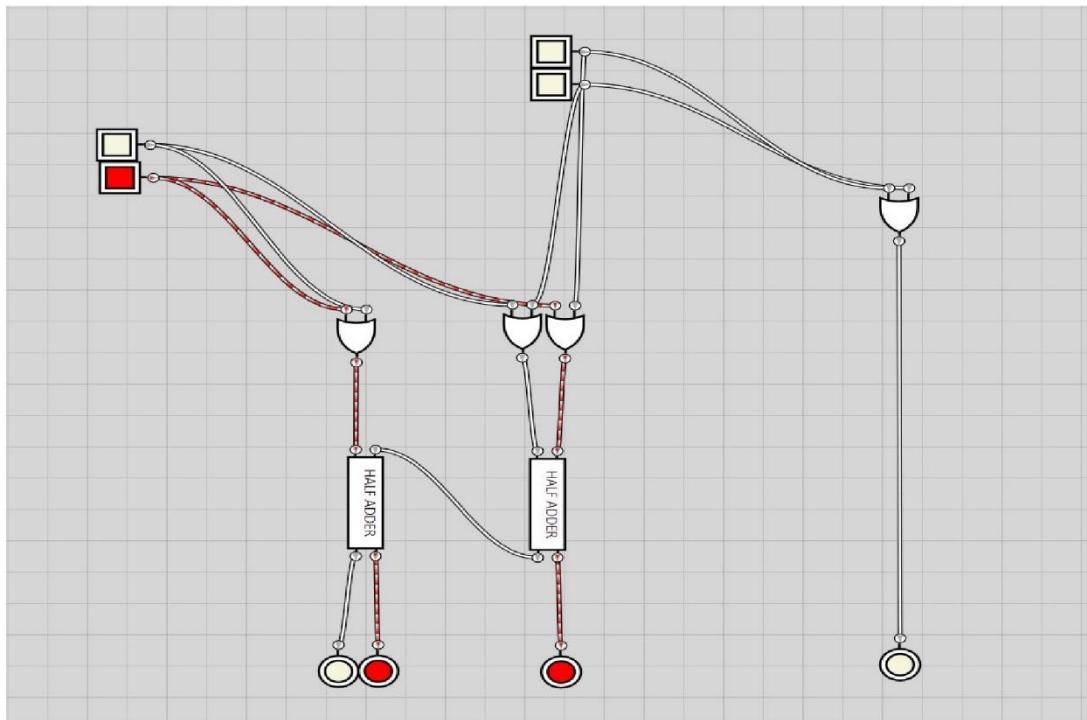
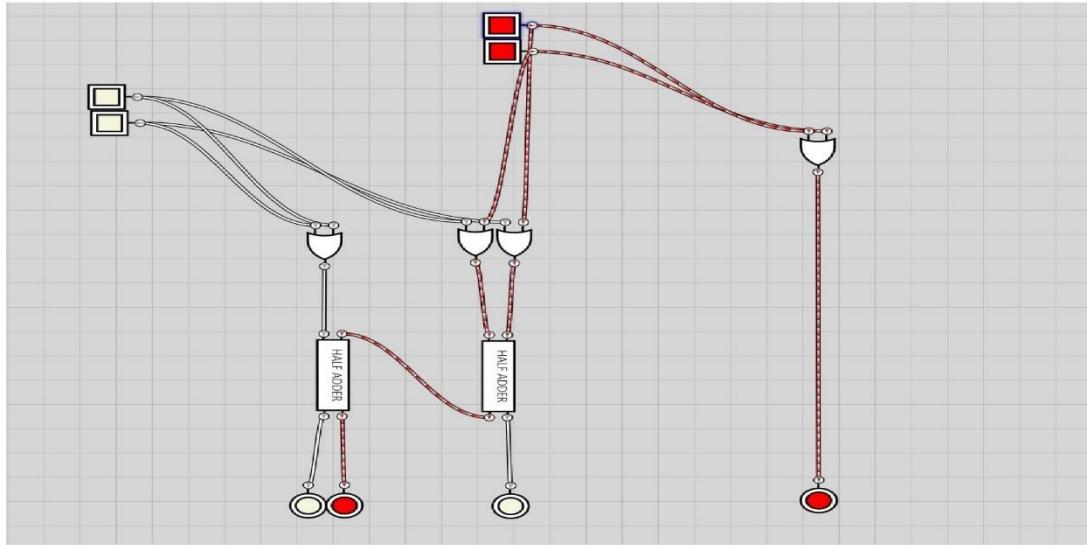
TRUTH TABLE :

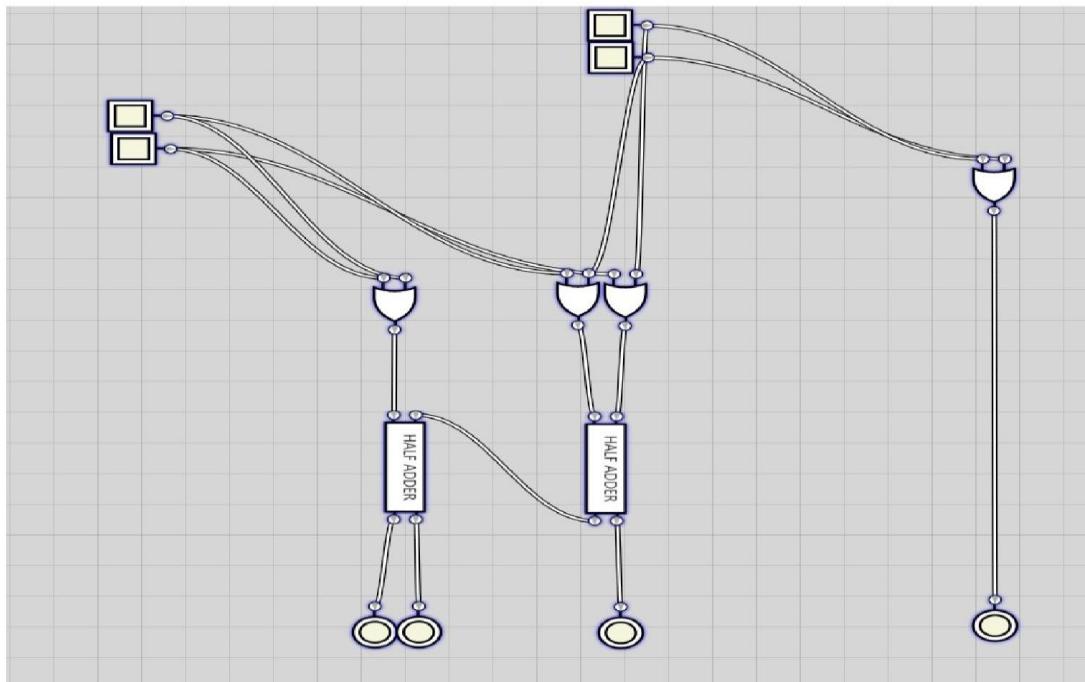
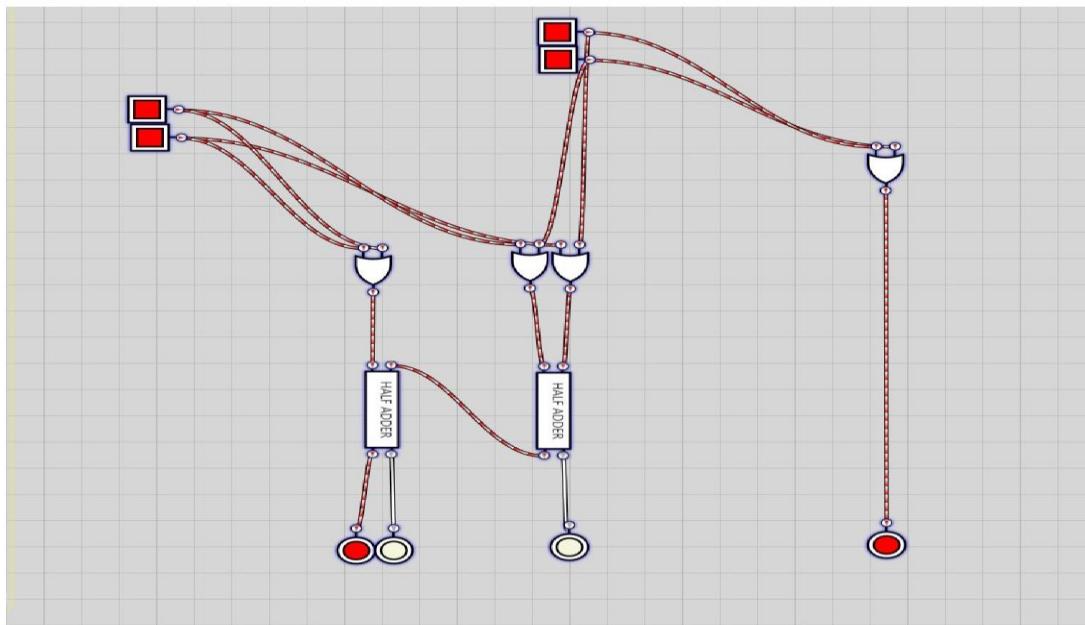
	Inputs				Outputs				
	A	B	C	D	W	X	Y	Z	
0x0	0	0	0	0	0	0	0	0	0
0x1	0	0	0	1	0	0	0	0	0
0x2	0	0	1	0	0	0	0	0	0
0x3	0	0	1	1	0	0	0	0	0
1x0	0	1	0	0	0	0	0	0	0
1x1	0	1	0	1	0	0	0	1	1
1x2	0	1	1	0	0	0	1	0	2
1x3	0	1	1	1	0	0	1	1	3
2x0	1	0	0	0	0	0	0	0	0
2x1	1	0	0	1	0	0	1	0	2
2x2	1	0	1	0	0	1	0	0	4
2x3	1	0	1	1	0	1	1	0	6
3x0	1	1	0	0	0	0	0	0	0
3x1	1	1	0	1	0	0	1	1	3
3x2	1	1	1	0	0	1	1	0	6
3x3	1	1	1	1	1	0	0	1	9

CIRCUIT DIAGRAM :



OUTPUT :





RESULT : 2- bit binary multiplier was constructed & studied.

EXPERIMENT NO. 13

DATE:

BINARY PARALLEL ADDER & SUBTRACTOR

AIM : To design and implement a binary parallel adder & subtractor.

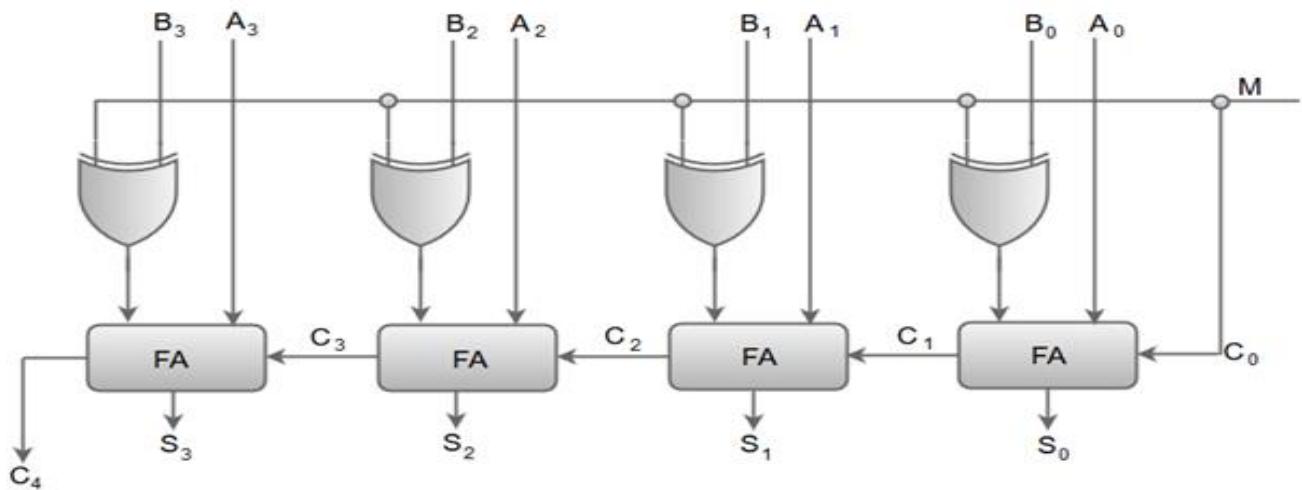
SOFTWARE USED : Logic gate simulator

TRUTH TABLE :

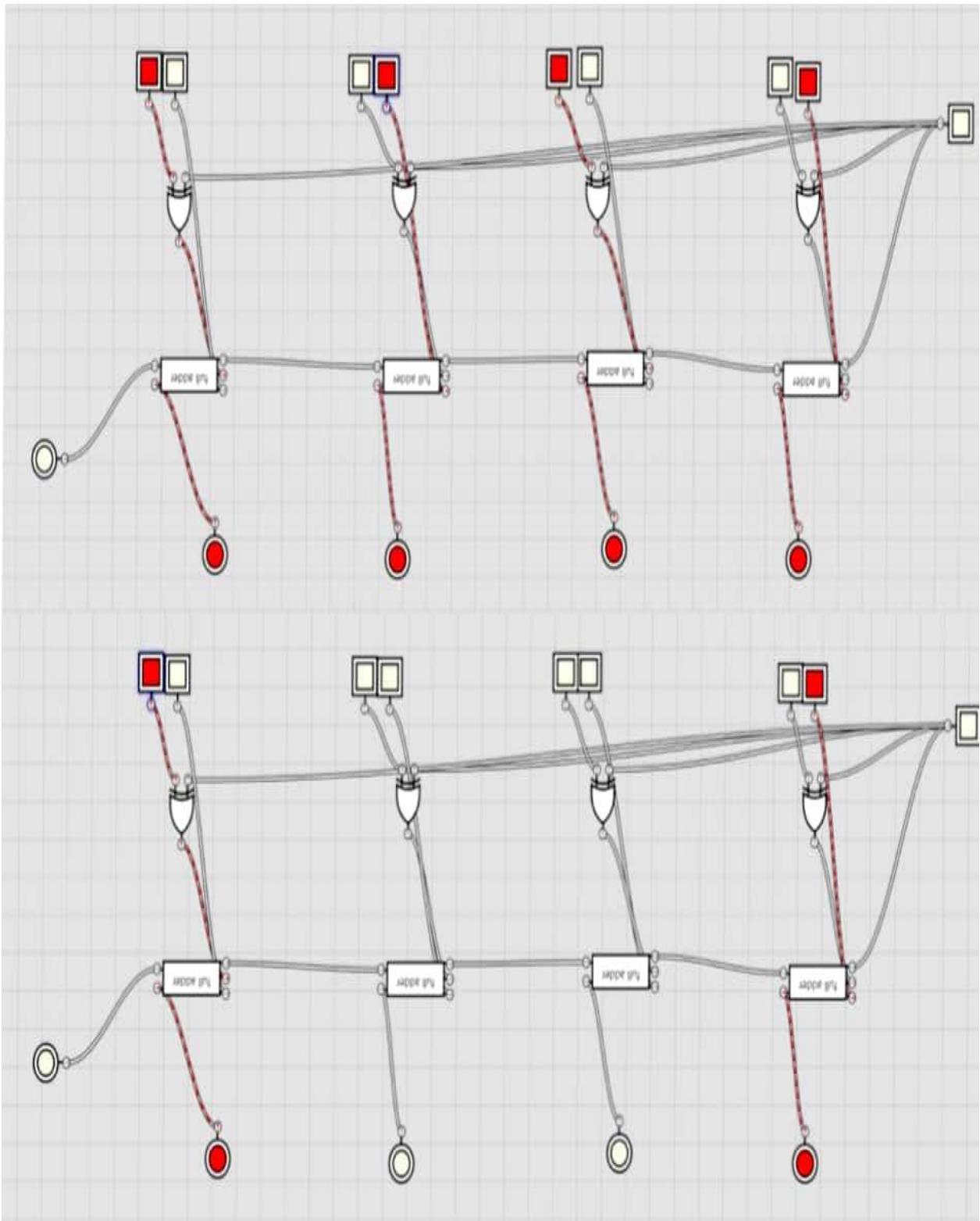
	B3	B2	B1	B0	M	X3	X2	X1	X0
Addition	0	0	0	0	0	0	0	0	0
	0	0	0	1	0	0	0	0	1
	0	0	1	0	0	0	0	1	0
	0	0	1	1	0	0	0	1	1
	0	1	0	0	0	0	1	0	0
	0	1	0	1	0	0	1	0	1
	0	1	1	0	0	0	1	1	0
	0	1	1	1	0	0	1	1	1
	1	0	0	0	0	1	0	0	0
	1	0	0	1	0	1	0	0	1
Subs traction	0	0	0	0	1	1	0	0	1
	0	0	0	1	1	1	0	0	0
	0	0	1	0	1	0	1	1	1
	0	0	1	1	1	0	1	1	0
	0	1	0	0	1	0	1	0	1
	0	1	0	1	1	0	1	0	0
	0	1	1	0	1	0	0	1	1
	0	1	1	1	1	0	0	1	0
	1	0	0	0	1	0	0	0	1
	1	0	0	1	1	0	0	0	0

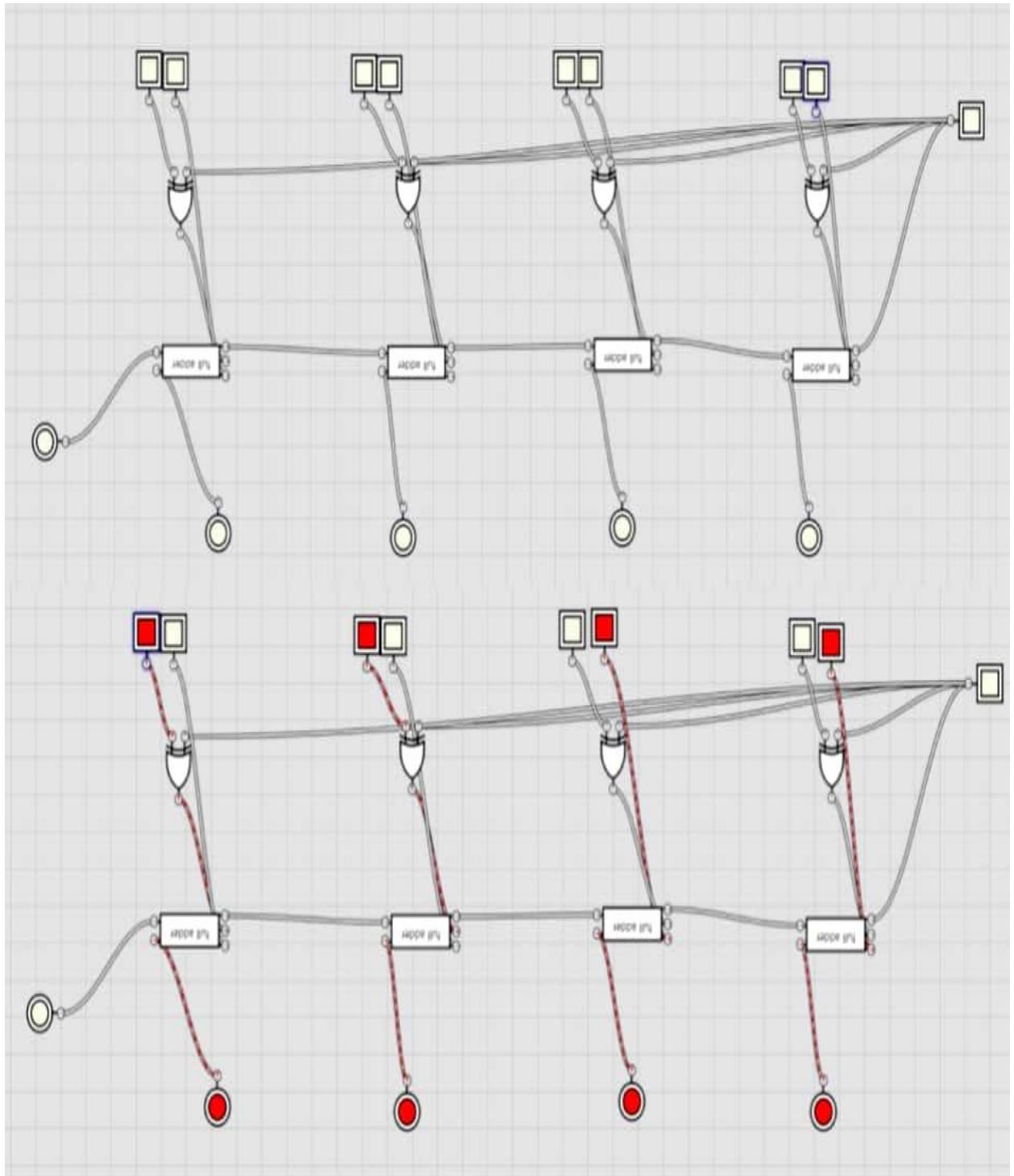
CIRCUIT DIAGRAM :

4 bit adder-subtractor:



OUTPUT :





RESULT : Binary parallel adder & subtractor was constructed & it's char. was studied.

EXPERIMENT NO. 14

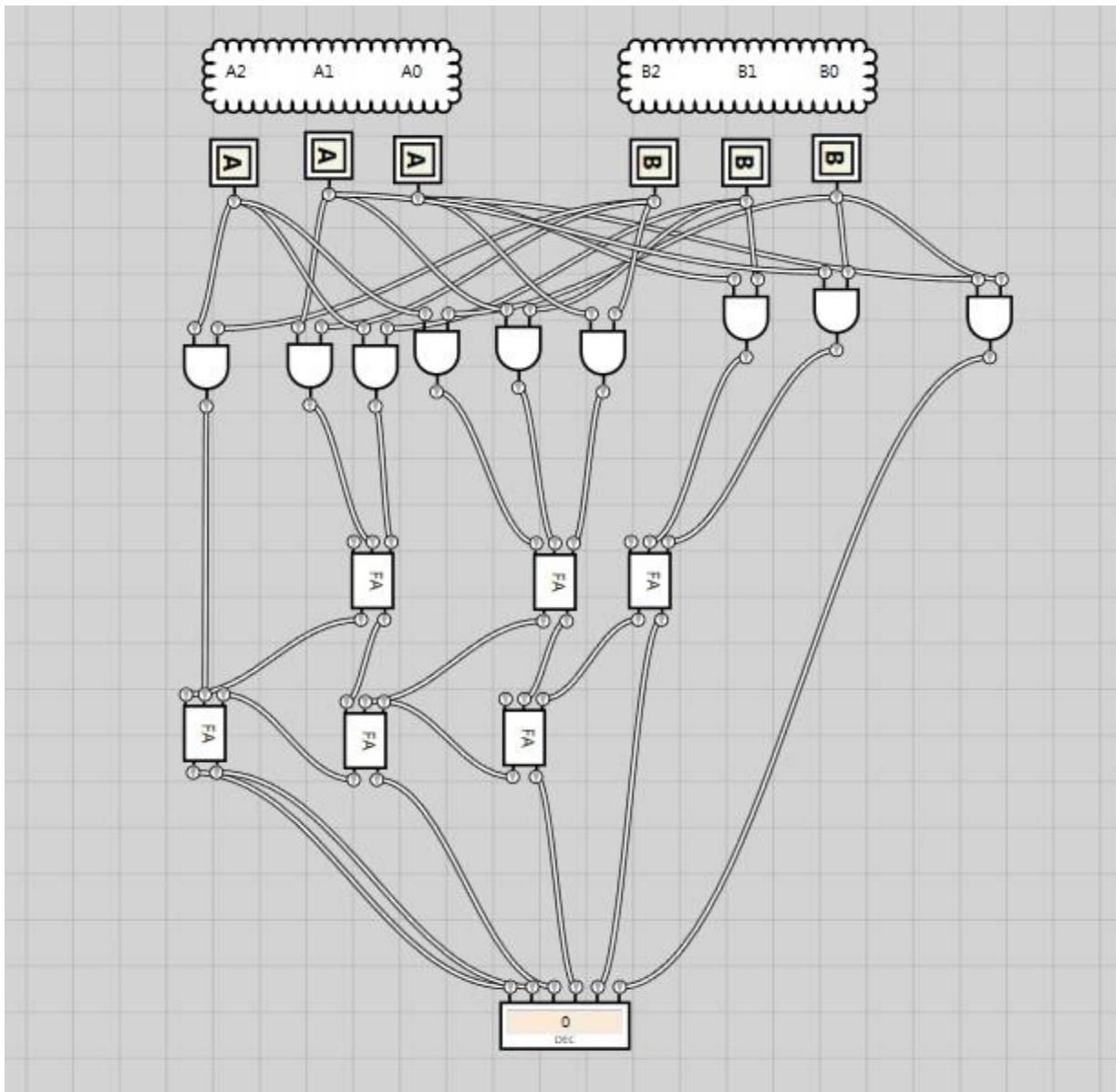
DATE:

CARRY SAVE MULTIPLIER

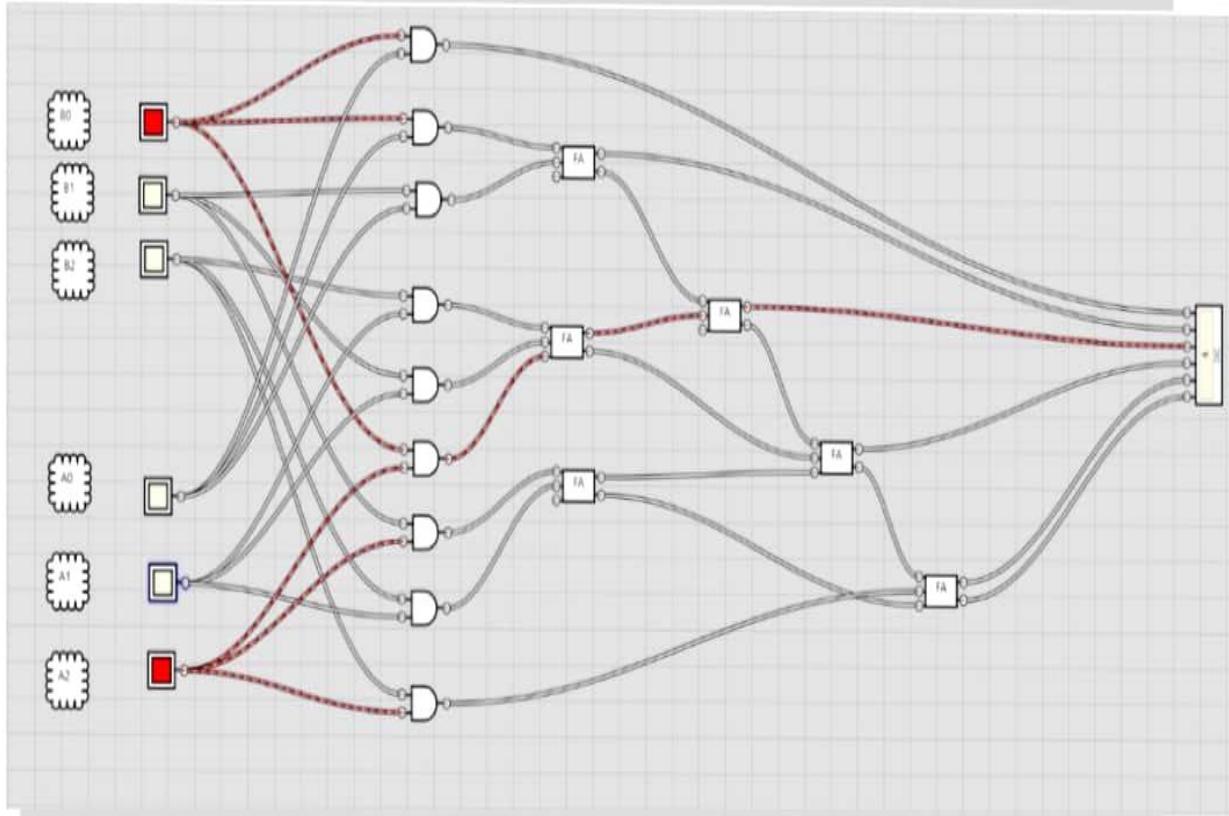
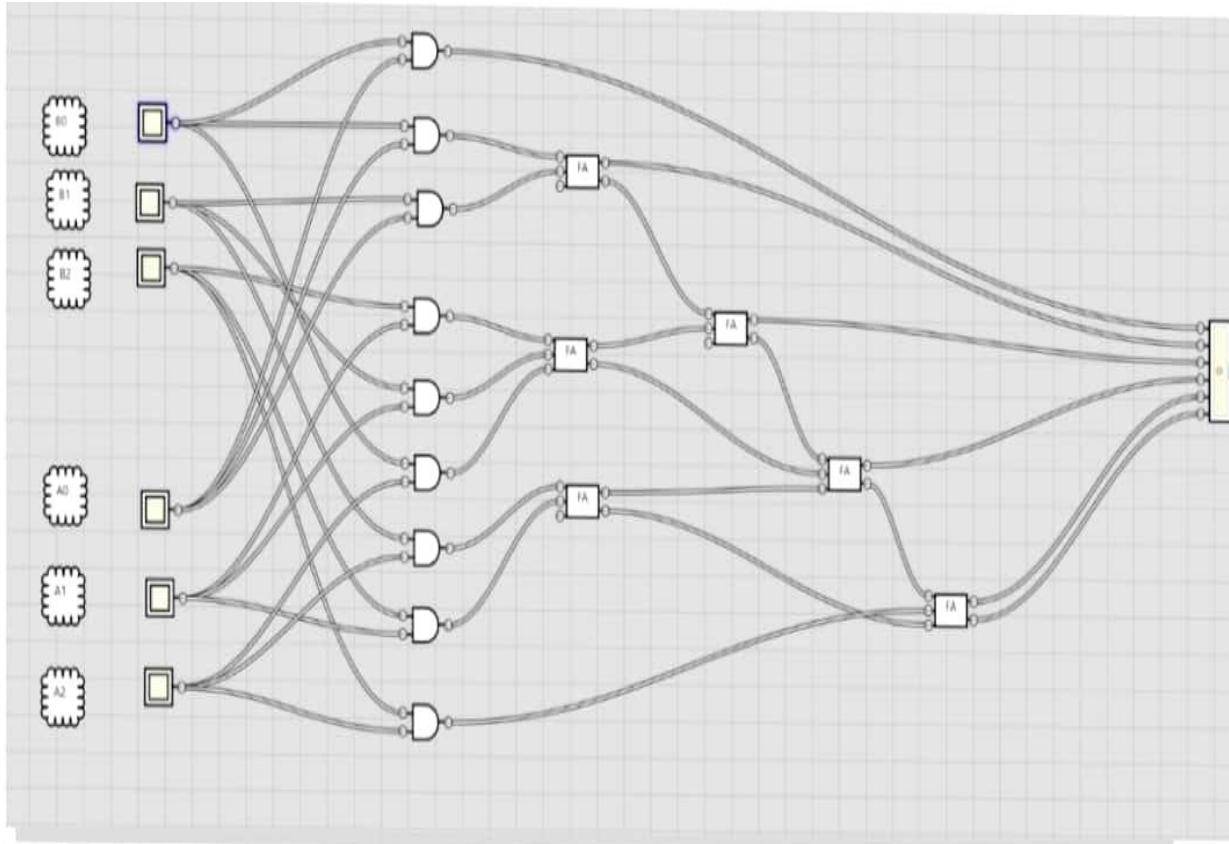
AIM : To design and implement a carry save multiplier.

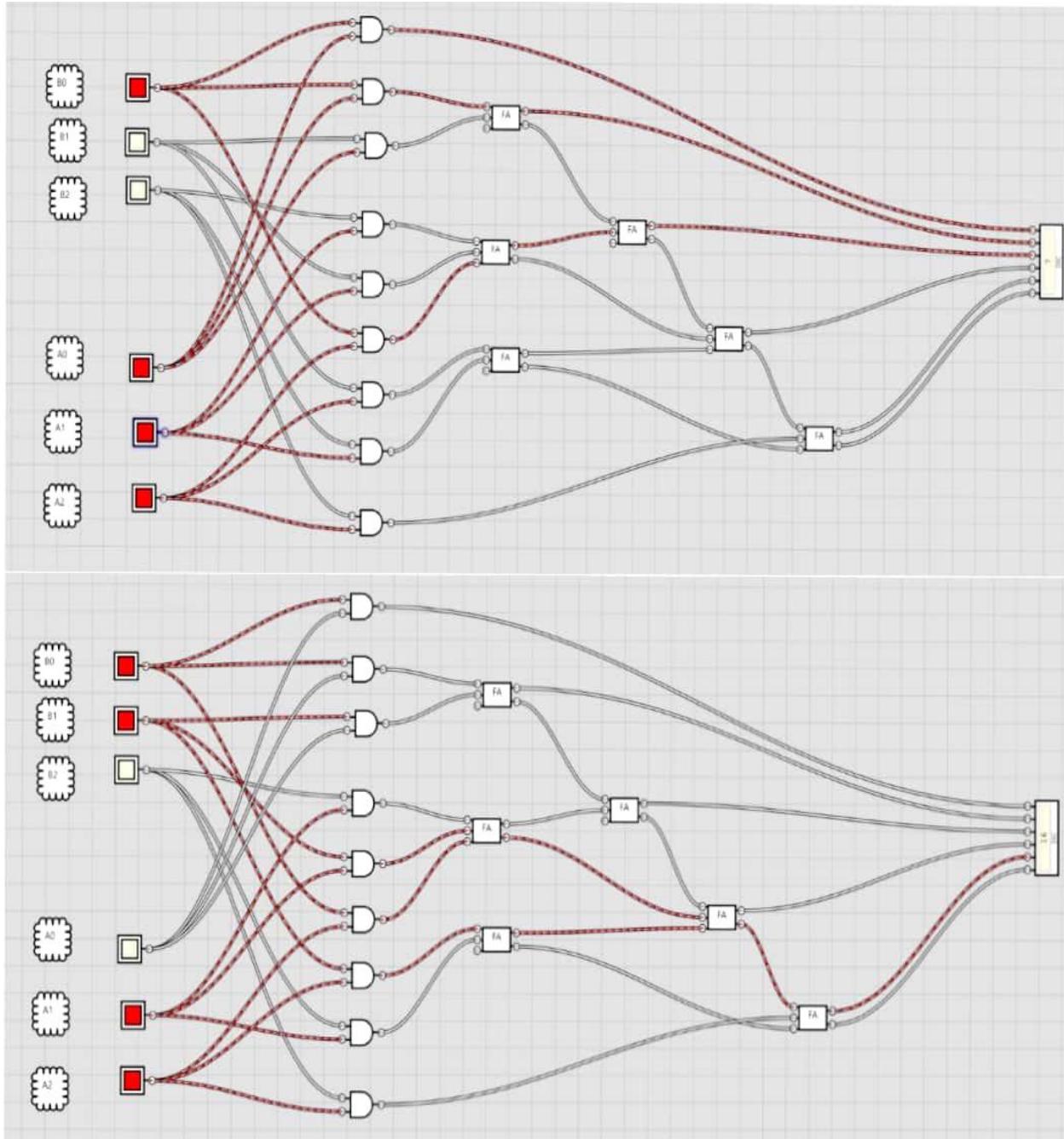
SOFTWARE USED : Logic gate simulator

CIRCUIT DIAGRAM :



OUTPUT :





RESULT : Carry save multiplier was constructed in simulator & char. were studied.

EXPERIMENT NO. 15

DATE:

MUX

AIM : To design and implement a Multiplexer.

SOFTWARE USED : Logic gate simulator

TRUTH TABLE:

(4*1)

s_1	s_0	x_3	x_2	x_1	x_0	y
0	0	x	x	x	0	0
0	0	x	x	x	1	1
0	1	x	x	0	x	0
0	1	x	x	1	x	1
1	0	x	0	x	x	0
1	0	x	1	x	x	1
1	1	0	x	x	x	0
1	1	1	x	x	x	1

(8*1)

Select Data Inputs			Output
s_2	s_1	s_0	Y
0	0	0	D_0
0	0	1	D_1
0	1	0	D_2
0	1	1	D_3
1	0	0	D_4
1	0	1	D_5
1	1	0	D_6
1	1	1	D_7

(16*1)

TRUTH TABLE

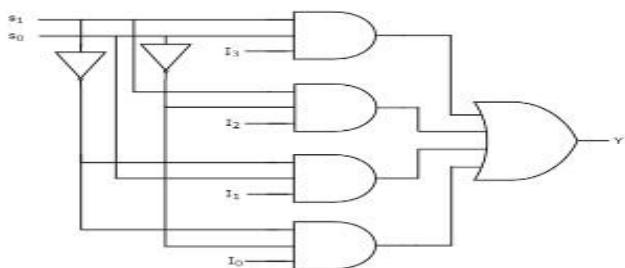
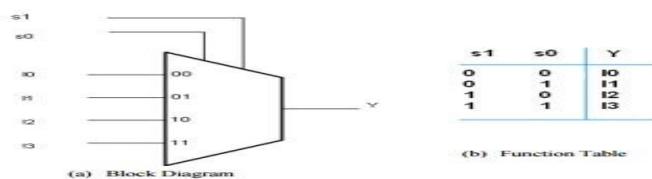
S₀	S₁	S₂	S₃	E	SELECTED CHANNEL
X	X	X	X	1	None
0	0	0	0	0	0
1	0	0	0	0	1
0	1	0	0	0	2
1	1	0	0	0	3
0	0	1	0	0	4
1	0	1	0	0	5
0	1	1	0	0	6
1	1	1	0	0	7
0	0	0	1	0	8
1	0	0	1	0	9
0	1	0	1	0	10
1	1	0	1	0	11
0	0	1	1	0	12
1	0	1	1	0	13
0	1	1	1	0	14
1	1	1	1	0	15

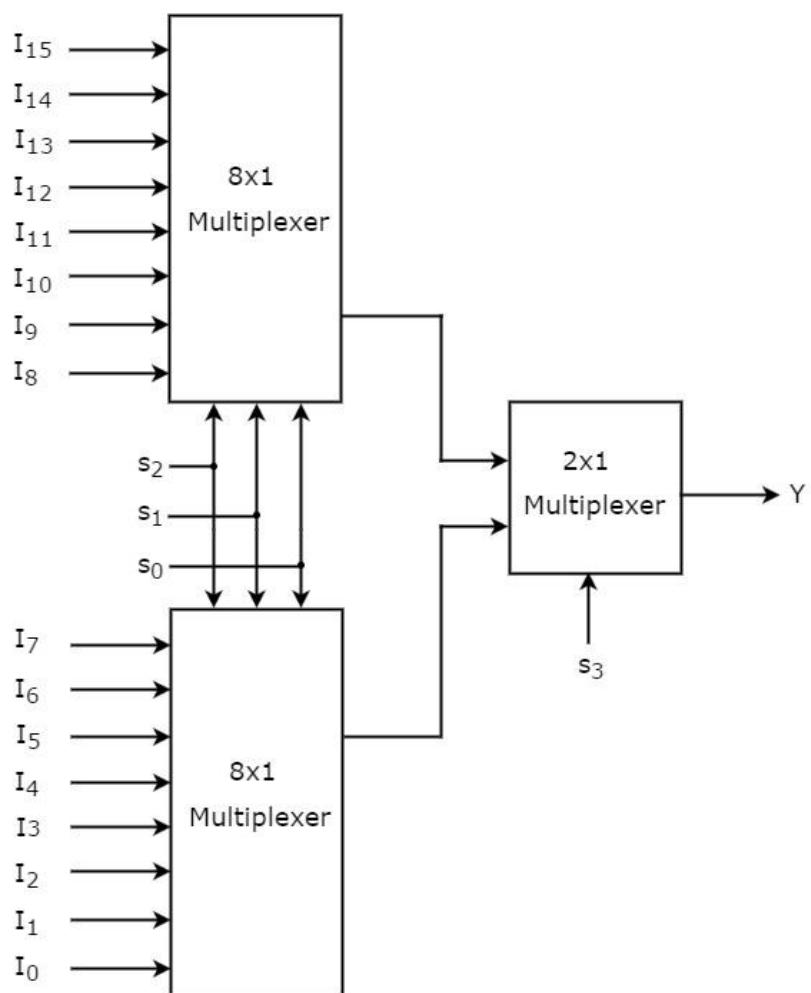
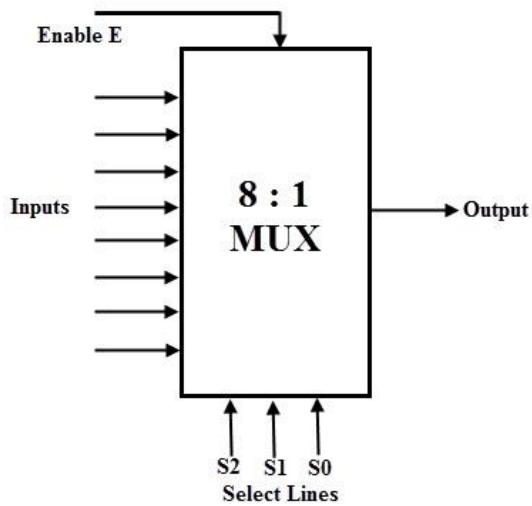
H= High Level

L= Low Level

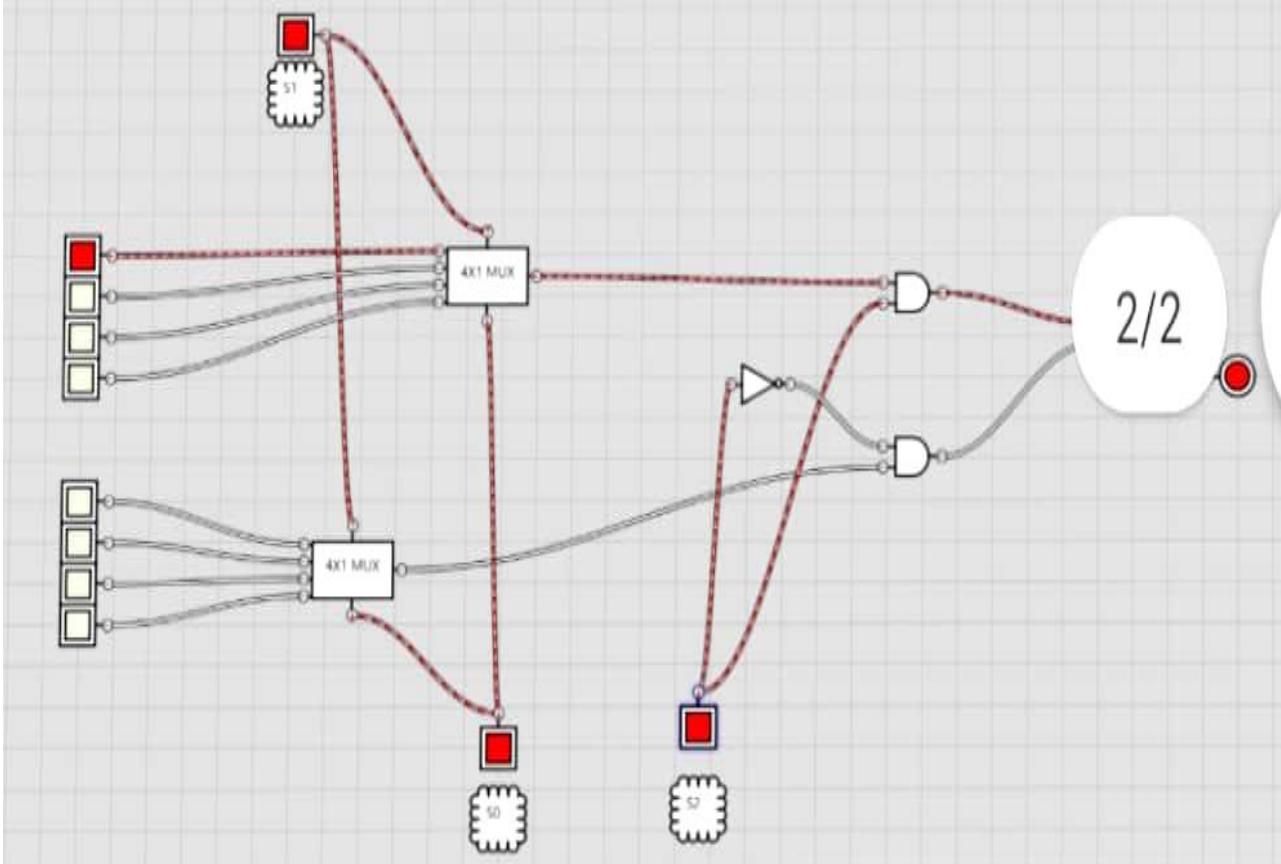
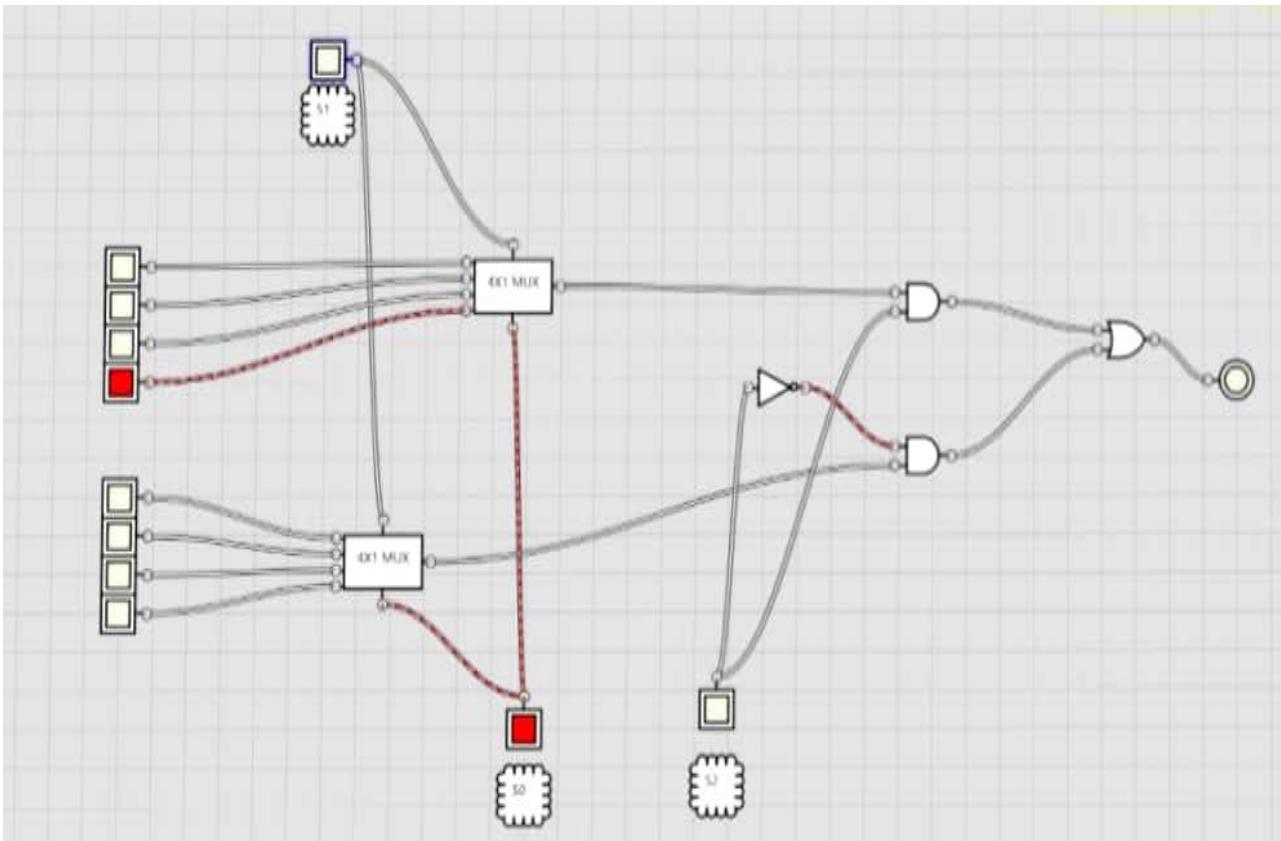
X= Don't Care

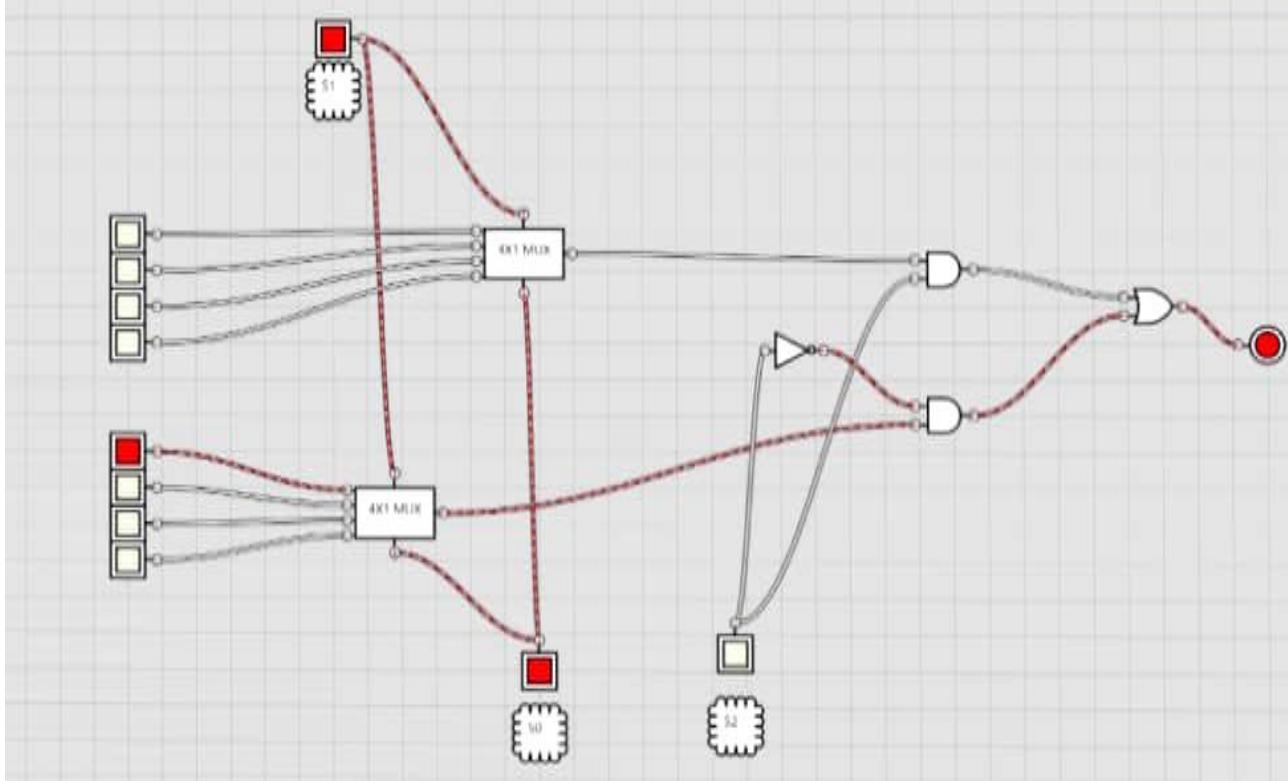
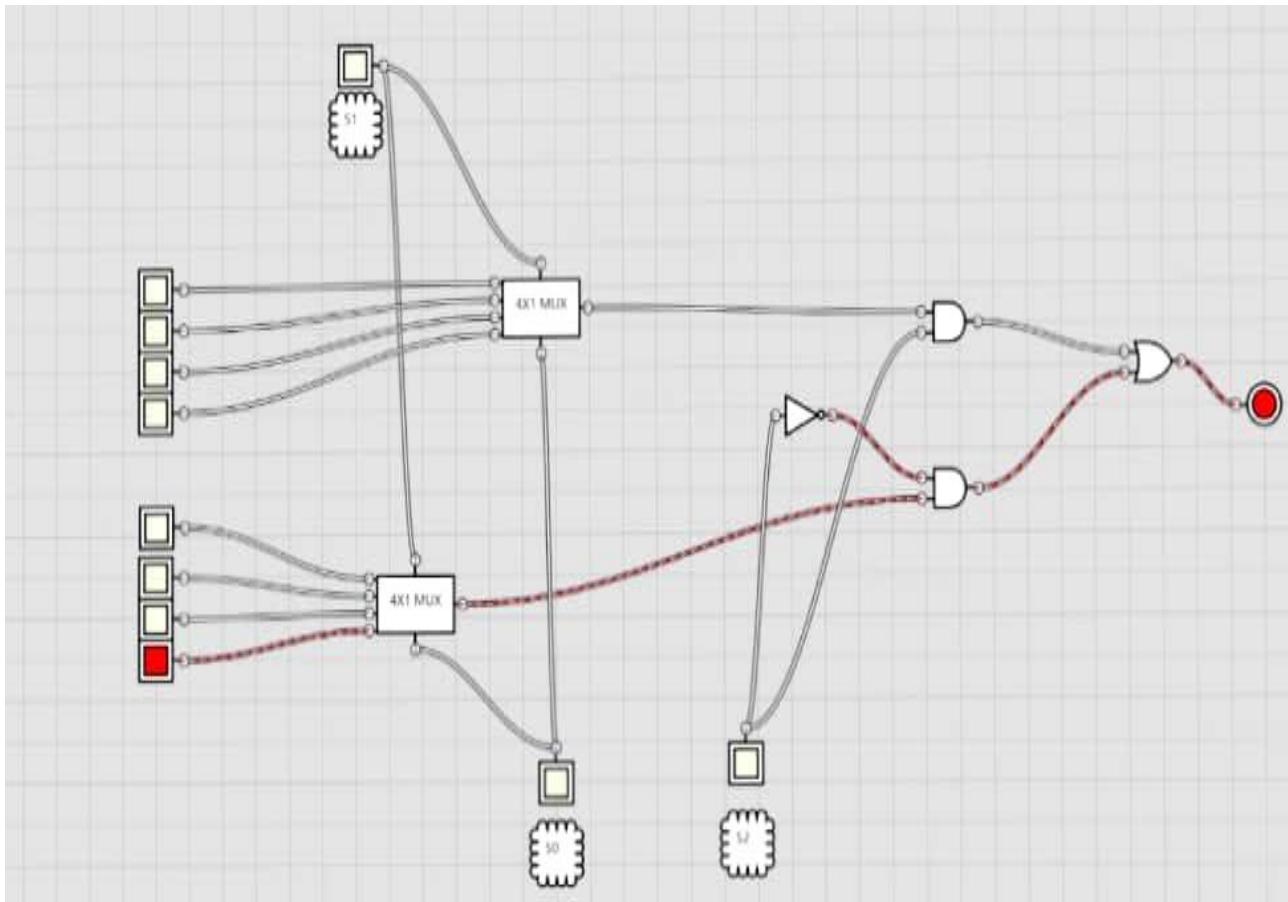
CIRCUIT DIAGRAM :

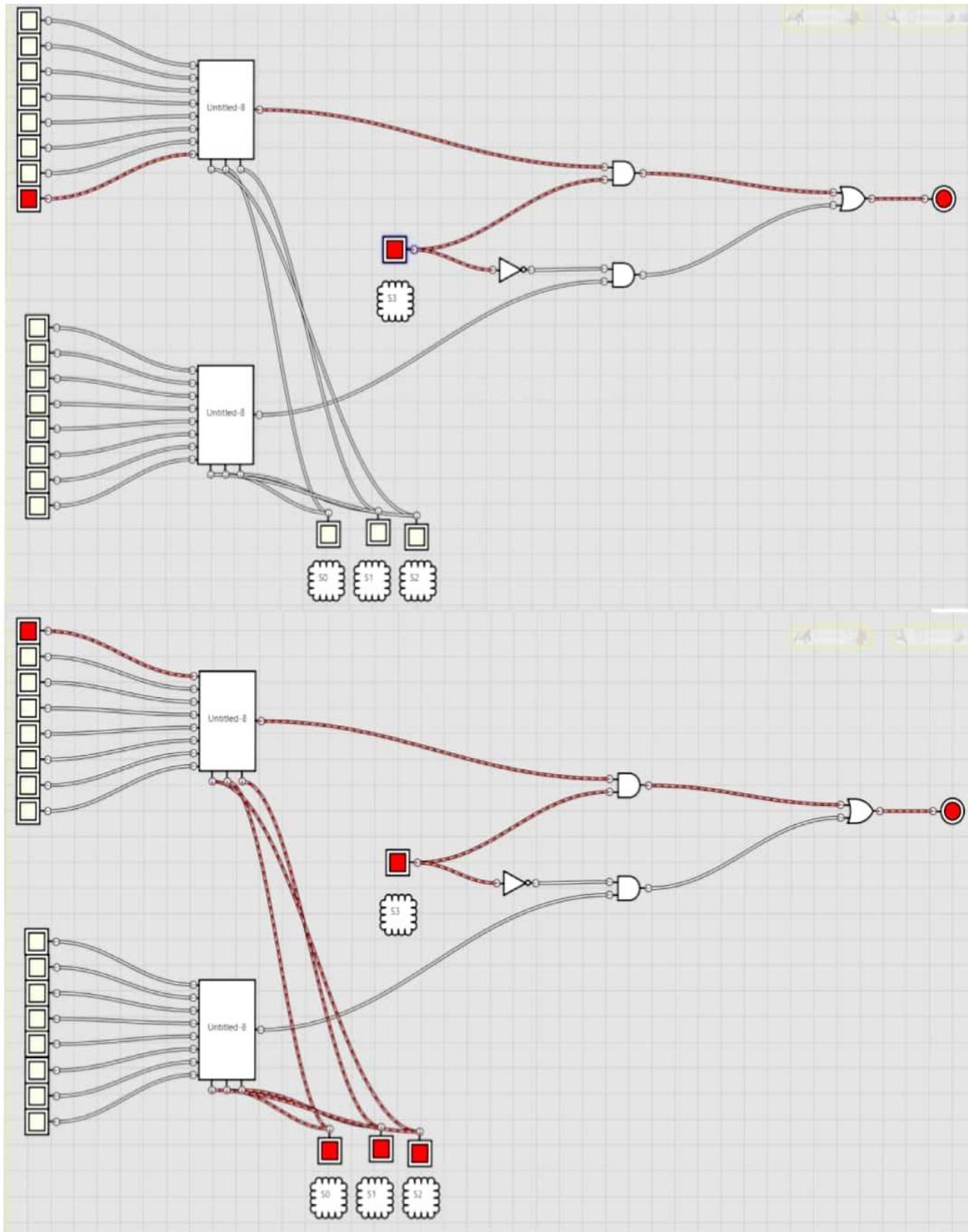




OUTPUT :







RESULT : 4:1, 8:1, 16:1 Multiplexers were constructed & studied.

EXPERIMENT NO. 16

DATE:

ARITHMETIC LOGIC UNIT (ALU)

AIM : To implement Arithmetic and Logical Unit which perform **Addition and subtraction arithmetic operations** and **AND, OR, XOR logical operations**.

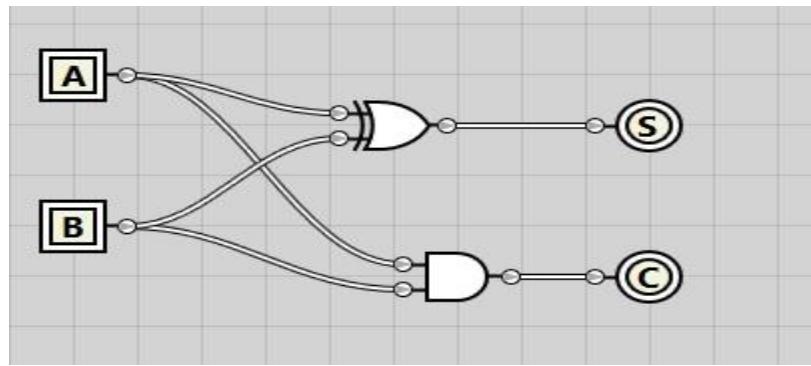
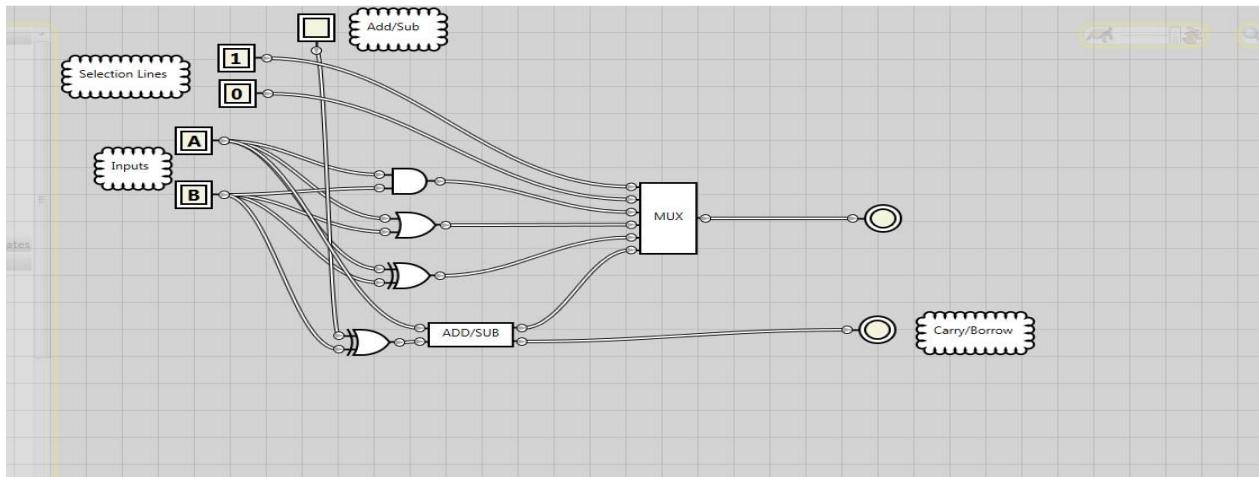
SOFTWARE USED : Logic gate simulator

TRUTH TABLE :

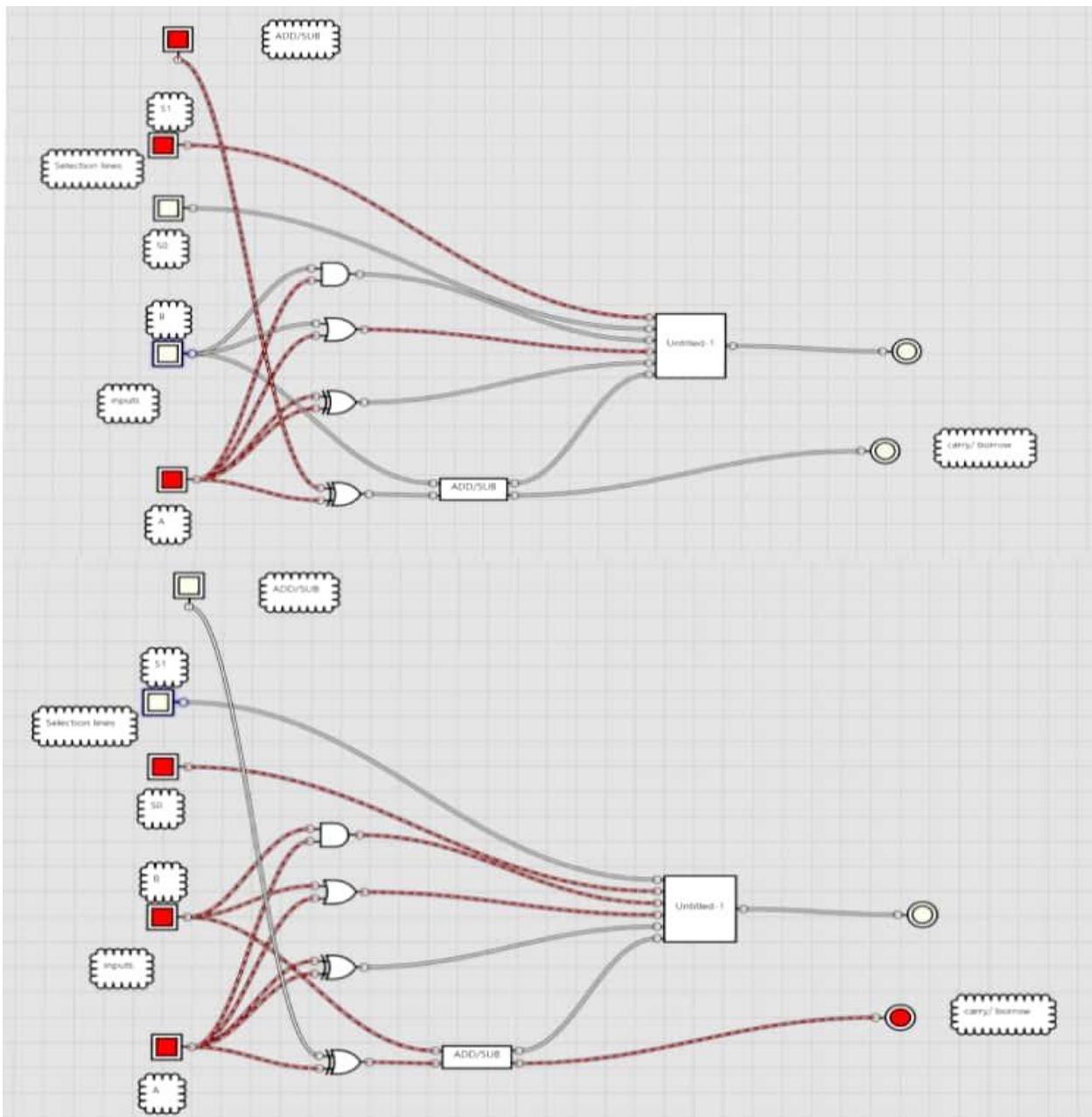
TABLE 1. TRUTH TABLE OF ALU

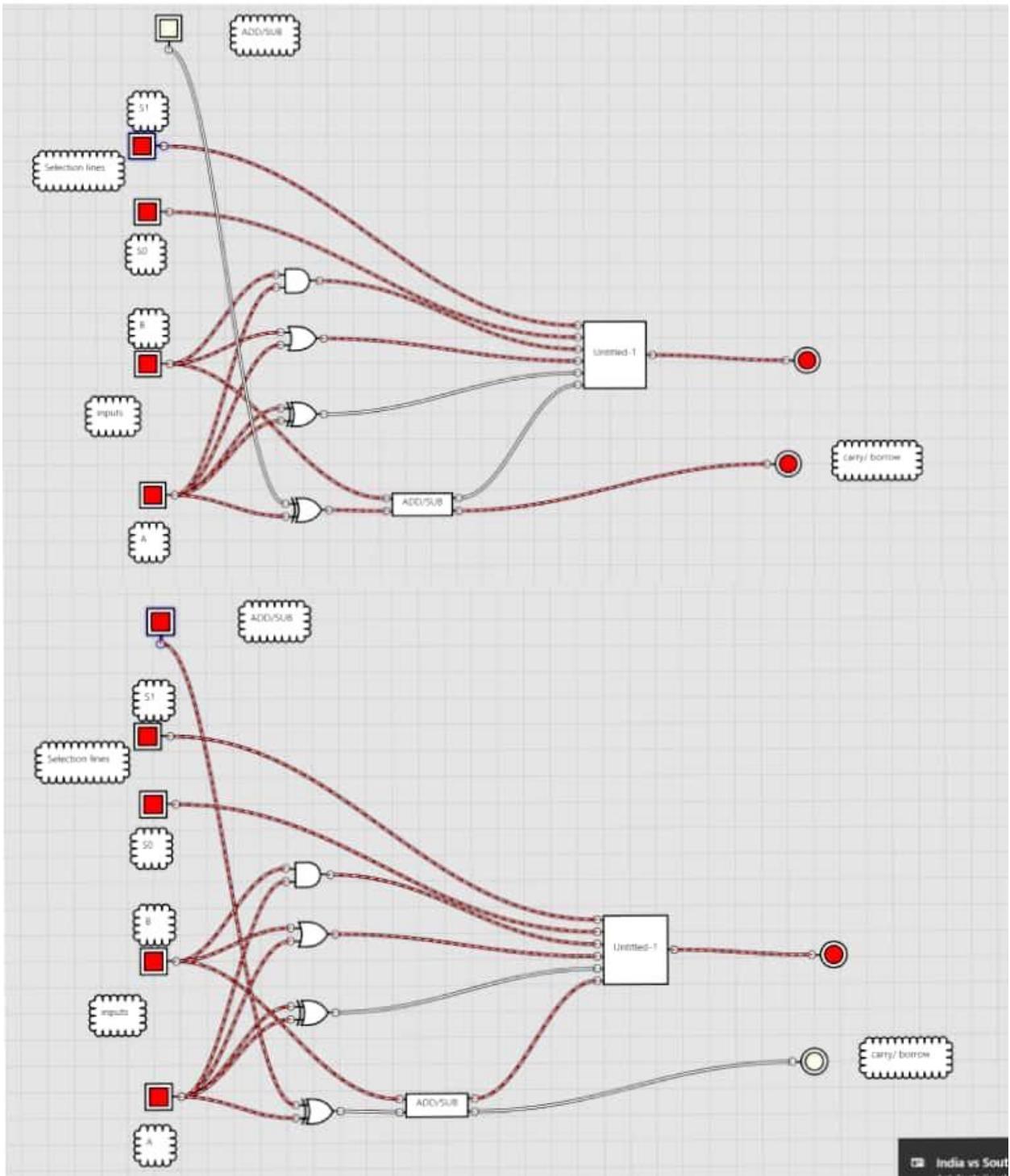
S ₂	S ₁	S ₀	OPERATION
0	0	0	OR
0	0	1	XNOR
0	1	0	XOR
0	1	1	AND
1	0	0	INCREMENT
1	0	1	ADDITION
1	1	0	DECREMENT
1	1	1	SUBTRACTION

CIRCUIT DIAGRAM :



OUTPUT :





RESULT : Arithmetic logic unit was designed & studied.