Title _____ Date 19/09/2022

Page _____

EXPERIMENT - 1

AIM : Implementation of Adder Circuit in Logisim.

REQUIREMENTS :

THEORY :

Adder : In electronics an adder is digital circuit that performs addition of numbers. In Modern computers adder resides in the Arithmetic Logic Unit (ALU).

- Adders are important not only in the computer but also in many types of digital systems in which the numeric data are processed.

Types of Adder

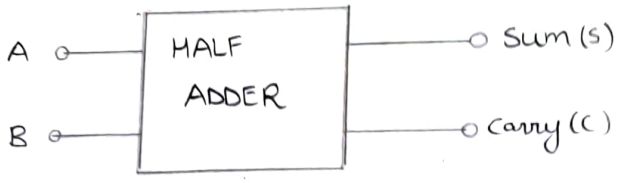
1) **Half Adder :** The half adder accepts two binary digits on it's inputs and produce two binary digits outputs, a sum bit and a carry bit.

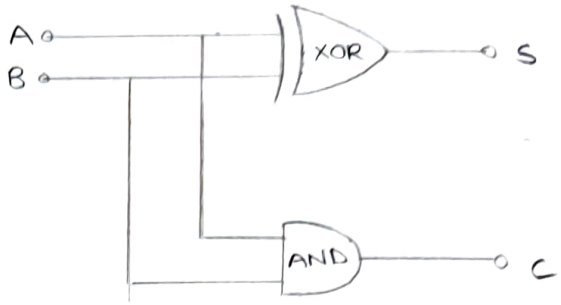
- The half adder is an example of a simple, functional digital circuit built from two logic gates. The half adder adds to one-bit binary numbers (AB). The output is the sum of the two bits sum (S) and the carry (C).
- The same two inputs are directed to two different gates. The inputs to the XOR gate are also the inputs to the AND gate. The input "wires" to the XOR gate are tied to the input wires of the AND gate; thus, when voltage is applied to the A input of the XOR gate, the A input to the AND gate recieves the same voltage.

The expressions obtained are :

CIRCUIT DIAGRAM :

1) HALF ADDER :

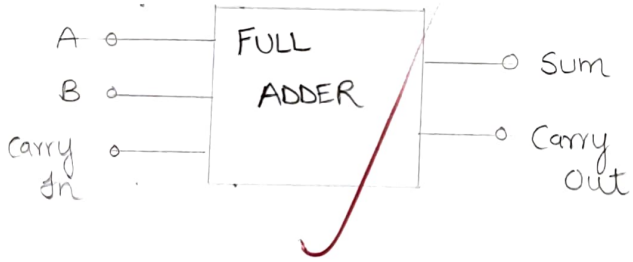


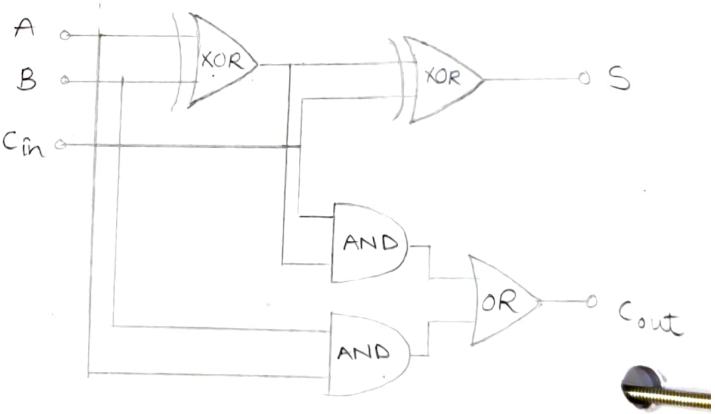


TRUTH TABLE :

A	B	Sum	Carry Out
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

2) FULL ADDER :





TRUTH TABLE :

A	B	Carry In	Sum	carry out
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Title _____ Date _____

Page _____

$S = A \oplus B$

$C = A \cdot B$

2.) **Full Adder :** The full adder accepts two inputs bits and an input carry and generates a sum output and an output carry.

- The full adder adds three one-bit binary numbers (A, B, Cin) and outputs two one-bit binary numbers, a sum (S) and a carry (Cout). The full adder is usually a component in a cascade of adders, which adds 8, 16, 32 etc binary numbers.
- The full adder is simply two half adders joined by an OR.
- We can implement a full adder circuit with help of two half adder circuits. The first half adder will be used to add A and B to produce a partial sum. The second half adder logic can be used to add Cin to the sum produced by the first half adder to get the final S output. If any of the adder logic produces a carry, there will be an output carry, Thus Cout will be an OR function of the half adder carry outputs.

The expressions obtained are :

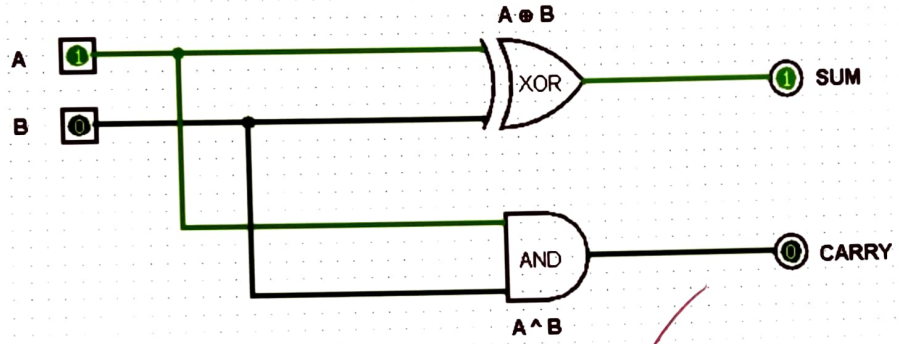
$S = A \oplus B \oplus C_{in}$

$C_{out} = AB + BC_{in} + C_{in}A$

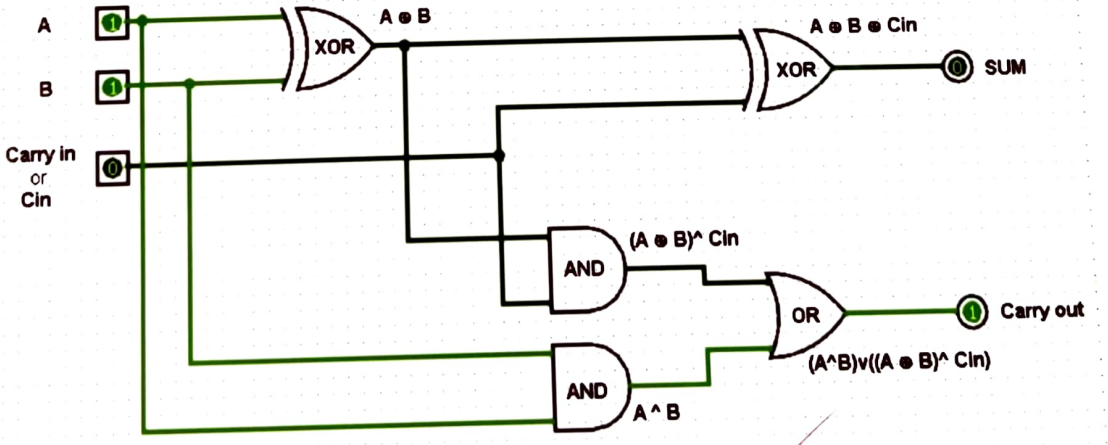
RESULT : The circuit of Half adder and Full adder have been studied and their truth tables were verified.

OUTPUT :

HALF ADDER

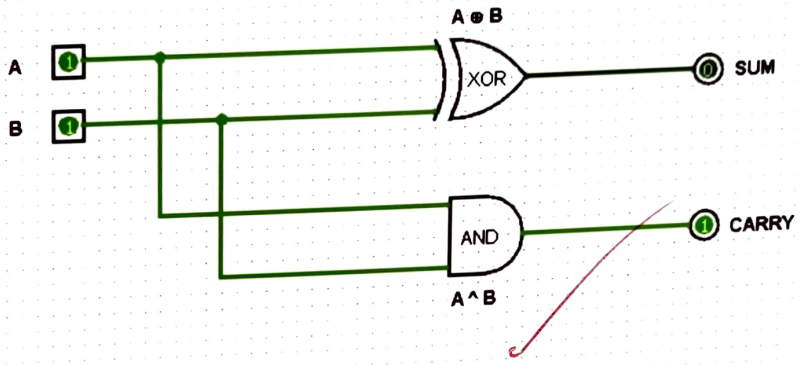


FULL ADDER

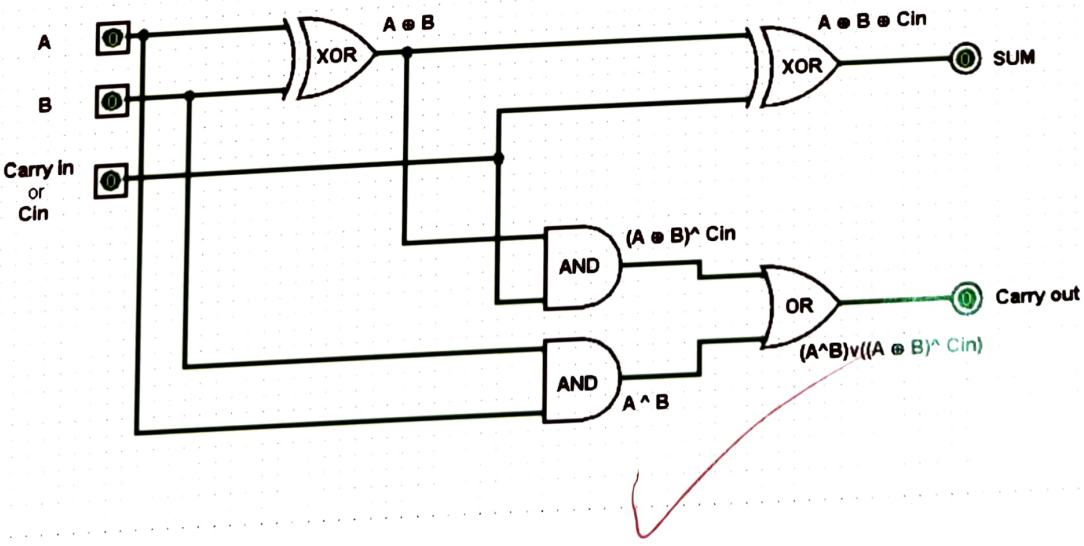


OUTPUT :

HALF ADDER



FULL ADDER



Title _____ Date 10/10/2022

Page _____

EXPERIMENT-2

AIM : To design and implement binary parallel adder circuit in Logisim.

REQUIREMENTS : Logisim software.

THEORY :

Parallel Adder :

- It is a circuit consisting of n-full adders, that will add n-bit binary numbers.
- The output consists of n sum bits and a carry bit.
- Cout of one full adder is connected to Cinput of the next full adder.

Structure of Parallel Adder :

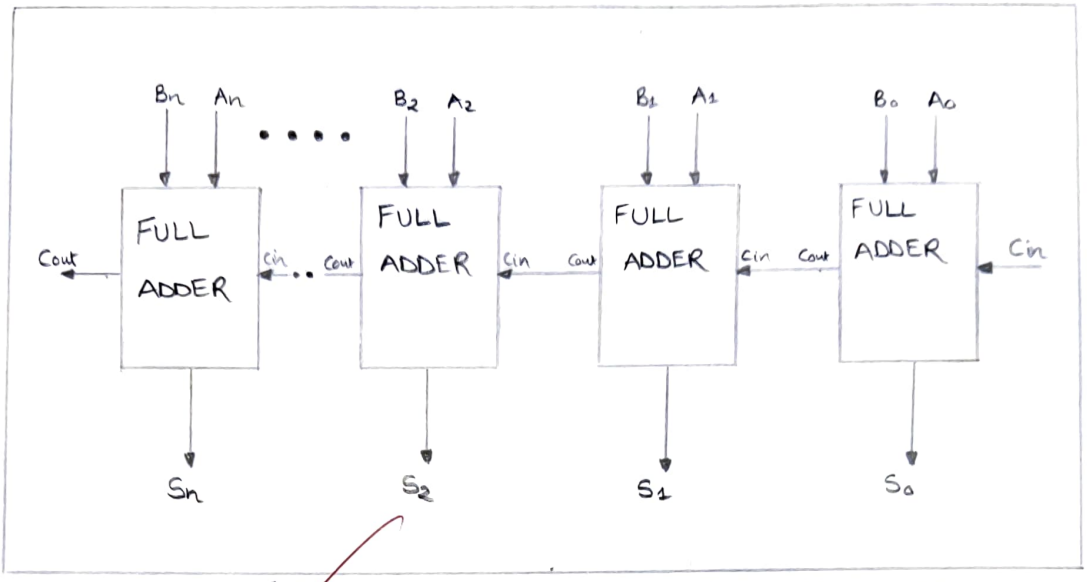
- A parallel adder is nothing but a cascade of several full adder.
- The number of full adders used will depend on the number of bits in the binary digits which require to be added.

Block diagram of n-bit binary parallel adder :

When a n-bit binary number is added to another, each column generates a sum and a 0 or 1 carry to the next higher order column.

PROCEDURE :

- The bits are added with full adder.
- Starting from the least significant position (LSB) to form the sum and carry.
- The input carry Cin in the least significant position must be zero.
- The value of C_{i+1} is a given significant portion is the output carry Cout of the full adder.
- This value is transferred into the input carry Cin of the full



BLOCK DIAGRAM OF N-bit BINARY PARALLEL ADDER

Demonstration :

For eg : A = 1010 , B = 1011

A + B = 1010
+ 1011
Sum = 10101

SUBSCRIPT	3	2	1	0	
Input carry	0	1	0	0	C_i
Augend (A)	1	0	1	0	A_i
Addend (B)	1	0	1	1	B_i
Sum	0	1	0	1	S_i
Output carry	1	0	1	0	C_{i+1}

Title _____ Date _____

_____ Page _____

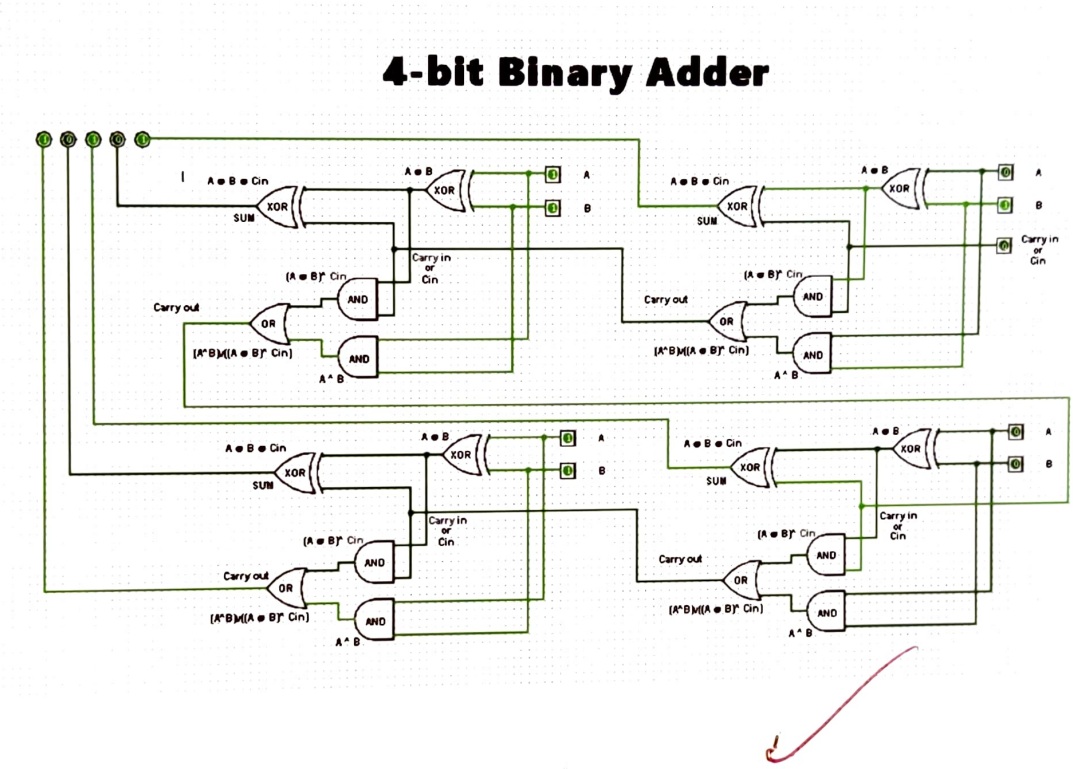
adder that adds the bits one higher significant position to the left.

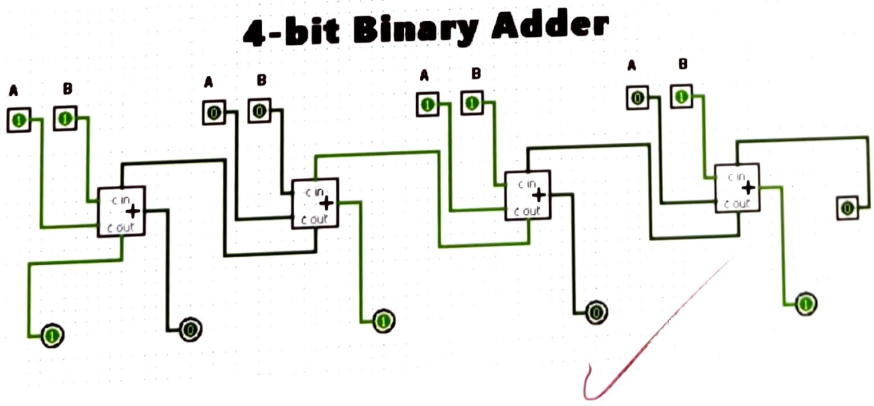
- The sum bits are thus generated starting from the right most position and are available as soon as the corresponding previous carry bit is generated.

RESULT :

The circuit of a 4-bit binary Adder has been studied and it's output has been verified.

OUTPUT :


4-bit Binary Adder
A ⊕ B ⊕ Cin
XOR
SUM
A ⊕ B
XOR
A
B
Carry in or Cin
(A ⊕ B)* Cin
AND
Carry out
OR
[A*B]v[(A ⊕ B)* Cin]
AND
A * B


4-bit Binary Adder
A
B
C in
C out

Title _____ Date 17/10/22

Page _____

EXPERIMENT – 3

AIM : To implement a subtractor circuit in Logisim.

REQUIREMENTS : Logisim software.

THEORY :

Subtractor : In electronics a subtractor is digital circuit that performs subtraction of numbers. subtractors are used in electronic calculators as well as digital devices.

Types of subtractor :

1.) **Half Subtractor :** It is used for subtracting one single bit binary number from another single bit binary number. It takes two inputs a Minuend (A) and subtrahend (B) and two outputs i.e Difference (D) and Borrow (Bout). The half subtractor is designed with the help of the following logic gates : i) 2-input AND gate ii) 2-input Exclusive-OR gate or XOR gate and iii) NOT gate. The difference bit (D) is generated with the help of the XOR gate and the Borrow bit (Bout) is generated with the help of a NOT and AND gate.

Boolean expression : from the truth table and K-map, the boolean expression can be derived as :

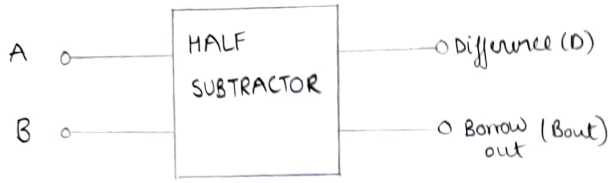
Difference (D) = $\bar{A} \cdot B + A \cdot \bar{B} = A \oplus B$

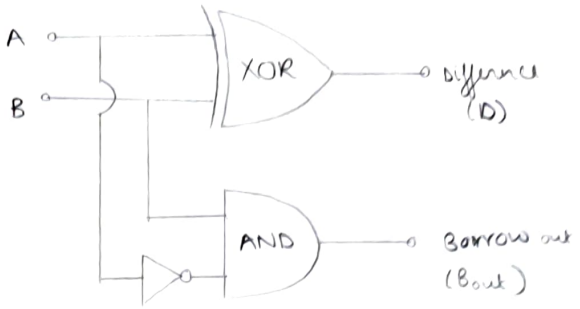
Borrow (Bout) = $\bar{A} \cdot B$

2.) **Full Subtractor :** It is a combinational circuit that performs subtraction of two single bit number. It takes three inputs a Minuend (A), subtrahend (B) and previous

CIRCUIT DIAGRAM :

1.) HALF SUBTRACTOR :

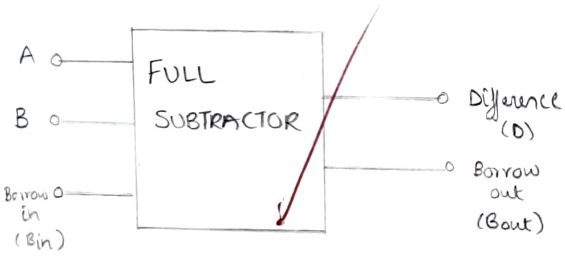


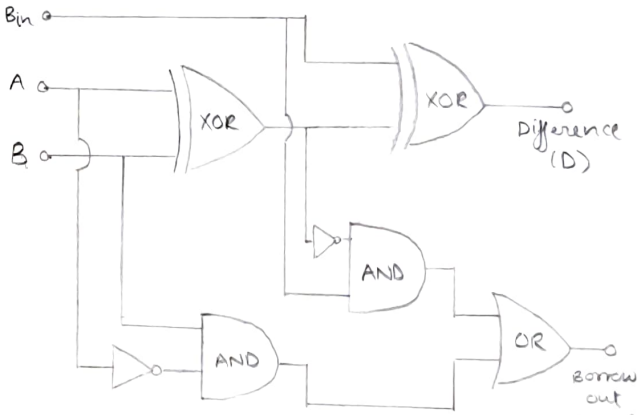


TRUTH TABLE :

A	B	Difference (D)	Borrow out (Bout)
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

2.) FULL SUBTRACTOR :





TRUTH TABLE :

A	B	Bin	Difference (D)	Borrow Out (Bout)
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Title _____ Date _____

_____ Page _____

borrow (Bin).

The full subtractor is designed with the help of the following logic gates : i) two 2-input AND gate ii) two 2-input Exclusive-OR gate or XOR gate iii) two NOT gate iv) 2-input OR gate.

Full subtractor generates two outputs a Difference bit (D) and a Borrow bit (Bout).

Boolean Expression : from the truth table and K-map, the boolean expression can be derived as :

Difference (D) $= \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$ where, $C = Bin$

$= A \oplus B \oplus C \equiv A \oplus B \oplus Bin$

Borrow (Bout) $= \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + ABC$

$= \bar{A}C(B+\bar{B}) + \bar{A}B(C+\bar{C}) + BC(A+\bar{A})$

$= \bar{A}C + \bar{A}B + BC$ where $C = Bin$

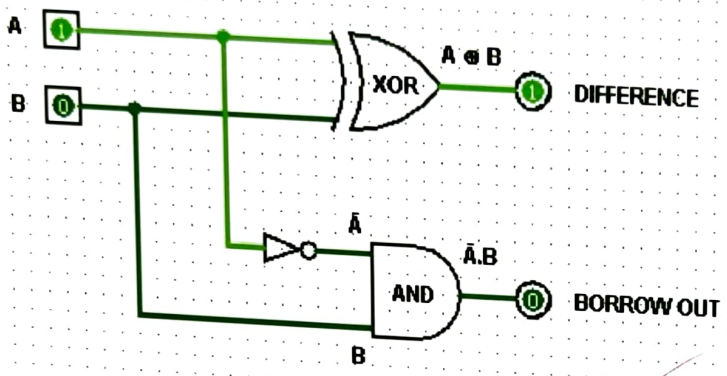
$= \bar{A}Bin + \bar{A}B + BBin$

$\equiv BC + (B \oplus C)A$

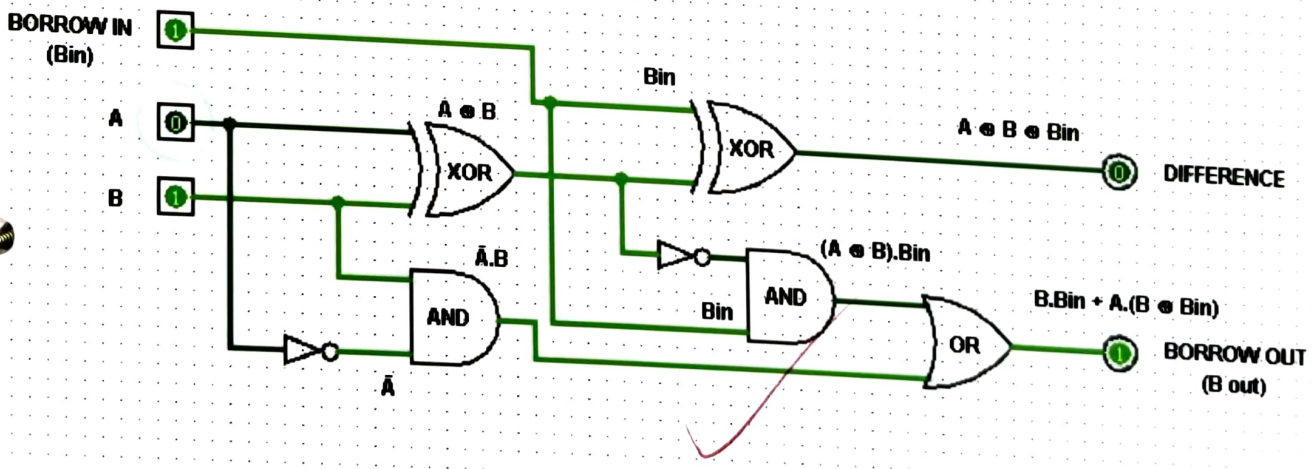
RESULT : The circuits of Half subtractor and Full subtractor have been studied and their truth tables were verified.

OUTPUT :

HALF SUBTRACTOR



FULL SUBTRACTOR



Title _____ Date 24/10/22

Page _____

EXPERIMENT - 4

AIM : Design and Implementation of Multiplexer and De-Multiplexer in Logisim.

REQUIREMENTS : Logisim Software

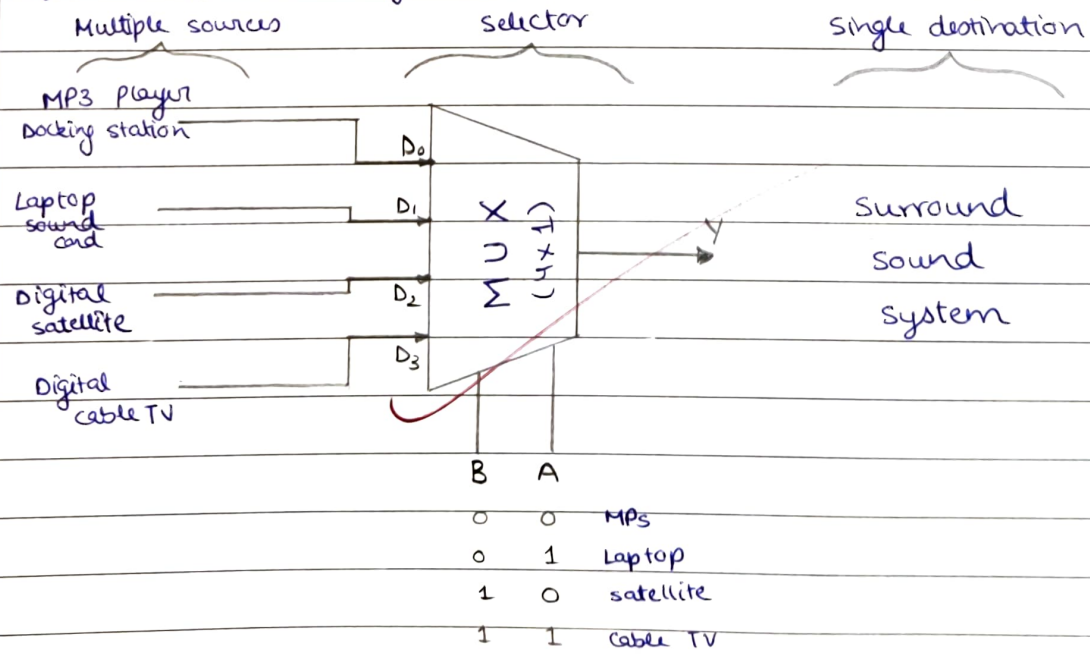
THEORY :

Multiplexer (MUX) :

- A MUX is a digital switch that has multiple inputs (sources) and a single output (destination)
- The select lines determine which input is connected to the output.
- **MUX Types :**
  - (i) 2 to 1 (1 select line)
  - (ii) 4 to 1 (2 select lines)
  - (iii) 8 to 1 (3 select lines)
  - (iv) 16 to 1 (4 select lines)

In a generalized NX1 MUX where N is 2^m has 'm' select lines.

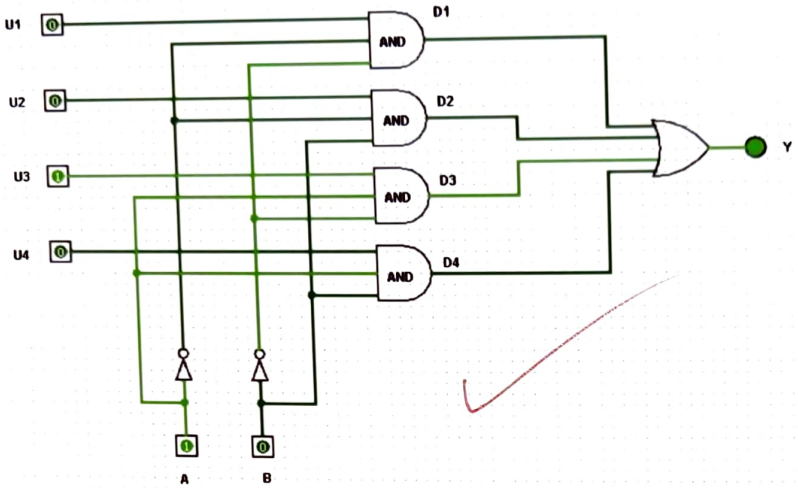
Typical Application of a MUX :



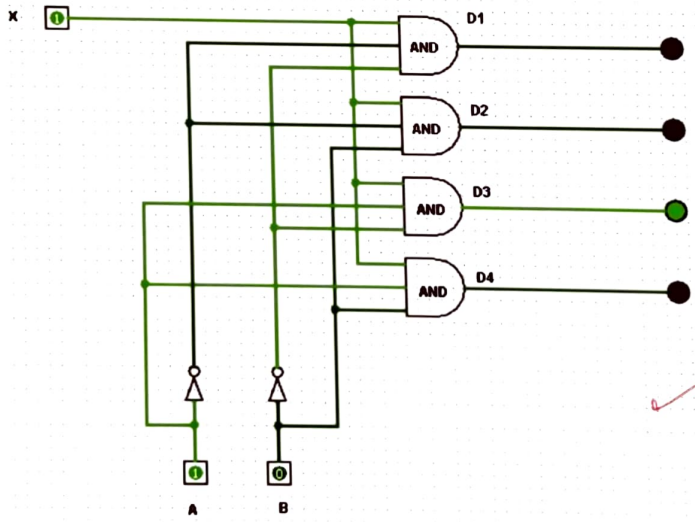
CLASSTIME

OUTPUT :

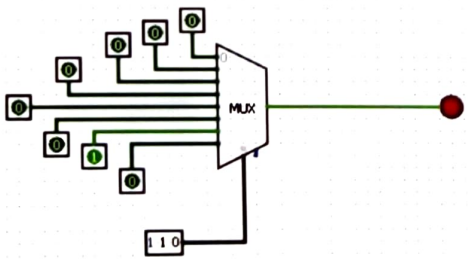
MULTIPLEXER (4 X 1)



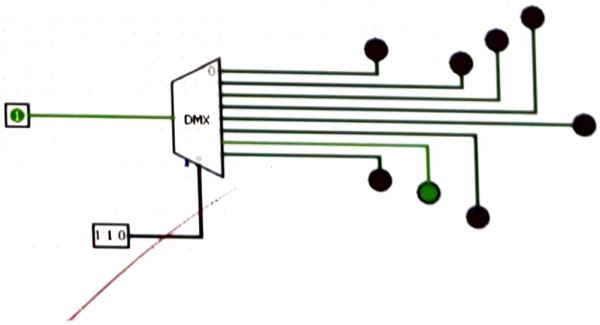
DE-MULTIPLEXER (1 X 4)



MULTIPLEXER (8 X 1)



DE-MULTIPLEXER (1 X 8)



Title _____ Date 7/11/22

Page _____

EXPERIMENT - 5

AIM : Design BCD to 7 segment LED Display Decoder using Logisim.

REQUIREMENTS : Logisim software

THEORY :

- In Binary Coded Decimal (BCD) encoding scheme each of the decimal numbers (0-9) is represented by it's equivalent binary pattern (which is generally of 4 bits).
- whereas, Seven segment display is an electronic device which consists of seven Light Emitting Diodes (LED's) arranged in a same definite pattern which is used to display numbers 0 to 9.
- BCD to seven segment decoder has four input lines (A, B, C and D) and 7 output lines (a, b, c, d, e, f and g), thus output is given to seven segment LED display which displays the decimal number depending upon inputs.
- seven segment displays are used to display the digits in calculators, clocks, various measuring instruments, digital watches and digital counters.

- From the Truth Table, the Boolean expressions of each output functions can be written as :

$a = F_1(A,B,C,D) = \Sigma m\,(0,2,3,5,7,8,9)$

$b = F_2(A,B,C,D) = \Sigma m\,(0,1,2,3,4,7,8,9)$

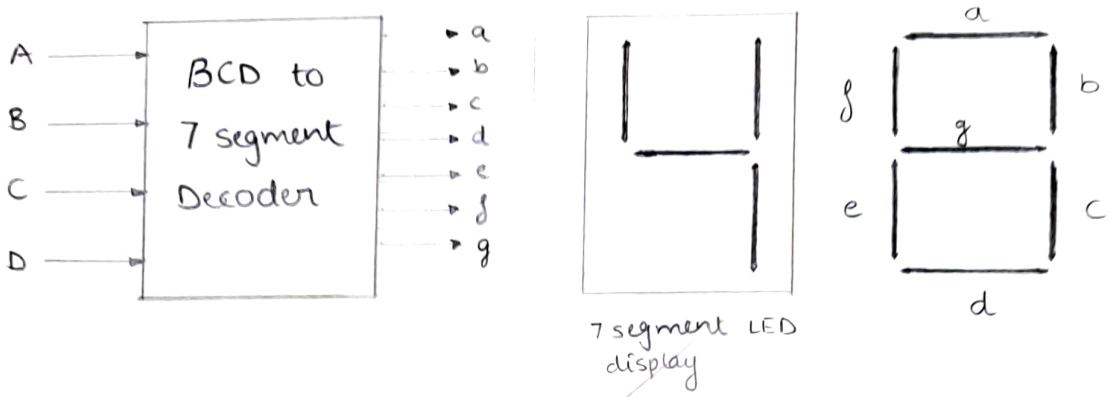
$c = F_3(A,B,C,D) = \Sigma m\,(0,1,3,4,5,6,7,8,9)$

$d = F_4(A,B,C,D) = \Sigma m\,(0,2,3,5,6,8)$

$e = F_5(A,B,C,D) = \Sigma m\,(0,2,6,8)$

$f = F_6(A,B,C,D) = \Sigma m\,(0,4,5,6,8,9)$

$g = F_7(A,B,C,D) = \Sigma m\,(2,3,4,5,6,8,9)$



TRUTH TABLE :

Decimal Digit	Input lines				Output lines							Display pattern
	A	B	C	D	a	b	c	d	e	f	g	
0	0	0	0	0	1	1	1	1	1	1	0	
1	0	0	0	1	0	1	1	0	0	0	0	
2	0	0	1	0	1	1	0	1	1	0	1	
3	0	0	1	1	1	1	1	1	0	0	1	
4	0	1	0	0	0	1	1	0	0	1	1	
5	0	1	0	1	1	0	1	1	0	1	1	
6	0	1	1	0	1	0	1	1	1	1	1	
7	0	1	1	1	1	1	1	0	0	0	0	
8	1	0	0	0	1	1	1	1	1	1	1	
9	1	0	0	1	1	1	1	1	0	1	1	

Title _____ Date _____

_____ Page _____

•) From the above simplification, we get the output value as :

$$a = A + C + BD + \bar{B}\bar{D}$$

$$b = \bar{B} + \bar{C}\bar{D} + CD$$

$$c = B + \bar{C} + D$$

$$d = \bar{B}\bar{D} + C\bar{D} + B\bar{C}D + \bar{B}C + A$$

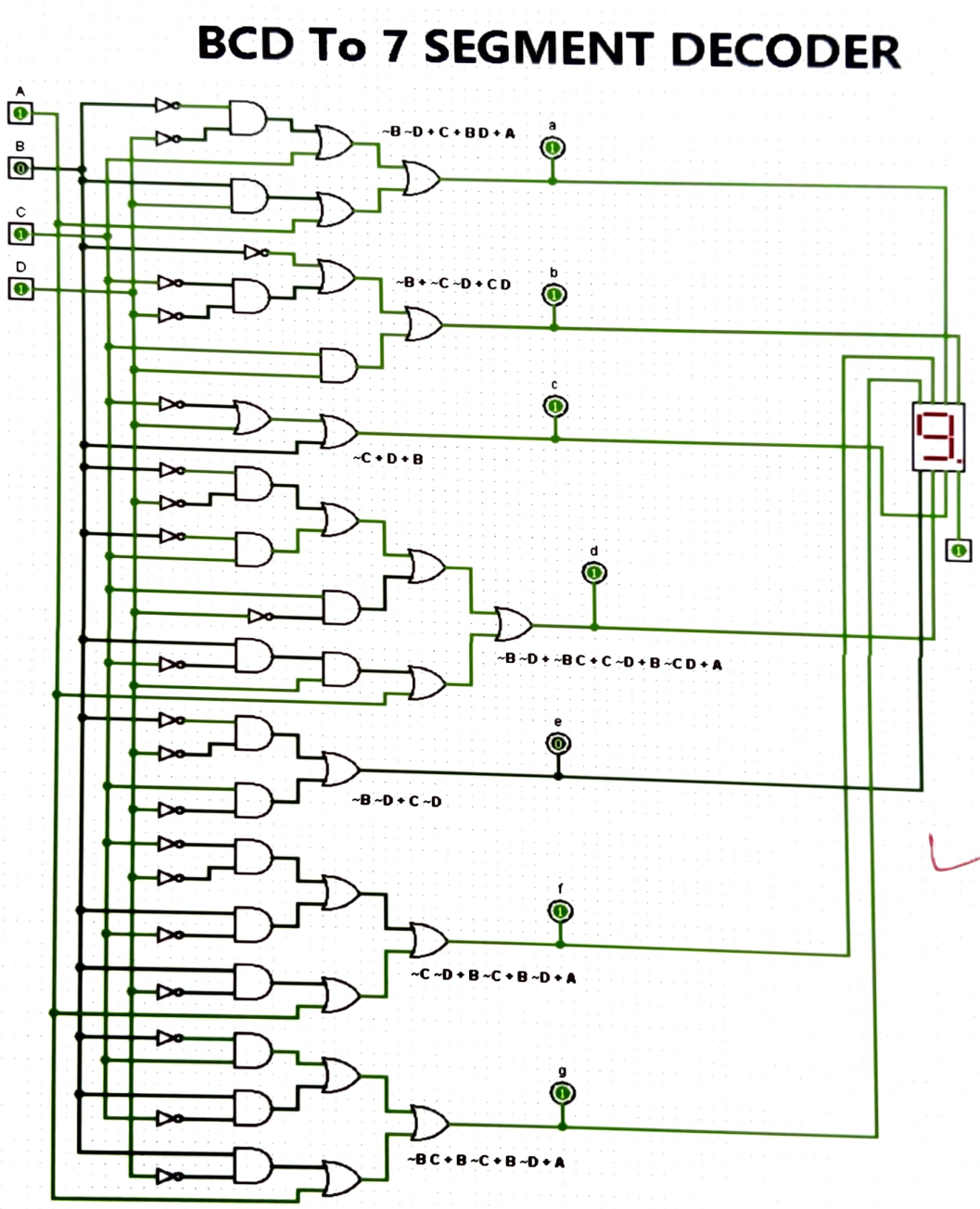
$$e = \bar{B}\bar{D} + C\bar{D}$$

$$f = A + \bar{C}\bar{D} + B\bar{C} + B\bar{D}$$

$$g = A + B\bar{C} + \bar{B}C + C\bar{D}$$

RESULT : The circuit of BCD to 7 segment decoder has been studied and it's output has been verified.

OUTPUT :


BCD To 7 SEGMENT DECODER
A
B
C
D
~B~D+C+BD+A
a
~B+~C~D+CD
b
~C+D+B
c
~B~D+~BC+C~D+B~CD+A
d
~B~D+C~D
e
~C~D+B~C+B~D+A
f
~BC+B~C+B~D+A
g
9

Topic _____ Date 14/11/2022 Page No. _____

EXPERIMENT - 6

AIM : Design 4-bit Ripple Counter in Logisim :

- Up Counter.
- Down Counter
- MOD - 12 Counter

REQUIREMENTS : Logisim Software

THEORY :

i) **Ripple Counter :**

Ripple Counter is a special type of asynchronous counter in which the clock pulse ripples through the circuit. In a ripple counter, the flip-flop output transition serves as a source of triggering other flip-flops.

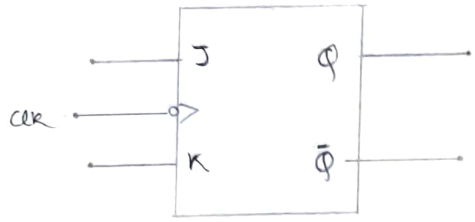
ii) **Asynchronous Counter :**

Asynchronous counters are also called Ripple counters because of the way the clock pulse ripples its way through the flip-flops. Counters are of two types depending upon the clock pulse applied. These counters are Asynchronous and Synchronous counters.

iii) **Synchronous Counter :**

In Synchronous counter, all flip-flops are triggered with same clock simultaneously. In asynchronous counter, different flip-flops are triggered with different clocks, not simultaneously.

• Negative edge Triggered Flip Flop :



TRUTH TABLE

CLK	J	K	Q_{n+1}
↓	0	0	Q_n
↓	0	1	0
↓	1	0	1
↓	1	1	Q_n

4- BIT RIPPLE UP - COUNTER

TRUTH TABLE :

CLK	Q_D	Q_C	Q_B	Q_A	Decimal
Initial	0	0	0	0	0
1st (↓)	0	0	0	1	1
2nd (↓)	0	0	1	0	2
3rd (↓)	0	0	1	1	3
4th (↓)	0	1	0	0	4
5th (↓)	0	1	0	1	5
6th (↓)	0	1	1	0	6
7th (↓)	0	1	1	1	7
8th (↓)	1	0	0	0	8
9th (↓)	1	0	0	1	9
10th (↓)	1	0	1	0	10
11th (↓)	1	0	1	1	11
12th (↓)	1	1	0	0	12
13th (↓)	1	1	0	1	13
14th (↓)	1	1	1	0	14
15th (↓)	1	1	1	1	15

4-BIT RIPPLE DOWN - COUNTER

TRUTH TABLE :

CLK	$\bar{Q}_D$	$\bar{Q}_C$	$\bar{Q}_B$	$\bar{Q}_A$	Decimal
Initial	1	1	1	1	15
1st (↓)	1	1	1	0	14
2nd (↓)	1	1	0	1	13
3rd (↓)	1	1	0	0	12
4th (↓)	1	0	1	1	11
5th (↓)	1	0	1	0	10
6th (↓)	1	0	0	1	9
7th (↓)	1	0	0	0	8
8th (↓)	0	1	1	1	7
9th (↓)	0	1	1	0	6
10th (↓)	0	1	0	1	5
11th (↓)	0	1	0	0	4
12th (↓)	0	0	1	1	3
13th (↓)	0	0	1	0	2
14th (↓)	0	0	0	1	1
15th (↓)	0	0	0	0	0

Topic _____ Date 14/11/2022 Page No. _____

iv) Up Counter :

This counter counts from zero to the maximum number value.

v) Down Counter :

This itype of counter counts from the maximum value to zero value.

vi) MOD Counter :

MOD counters are dfined based on the number of states that the counter will sequence through before returning back to its original value.

A Modulus - 12 counter will count from 0 (0000) to 11 (1011) and would require four flip-flops.

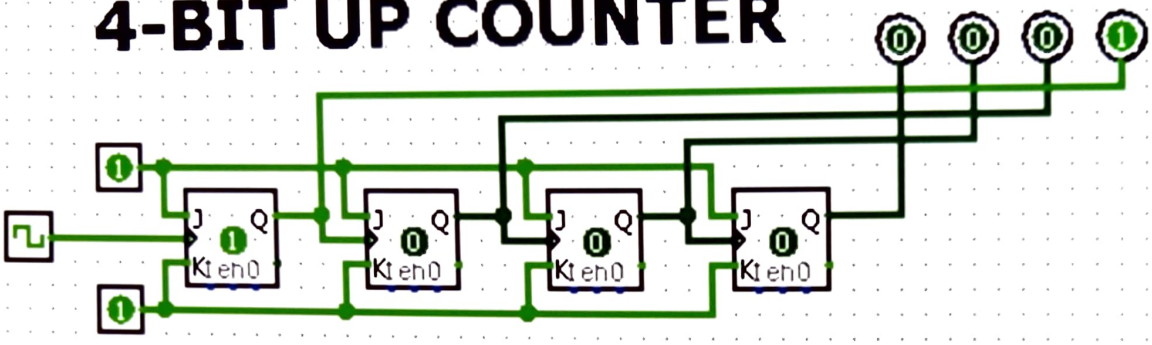
MOD - 12 UP COUNTER

TRUTH TABLE :

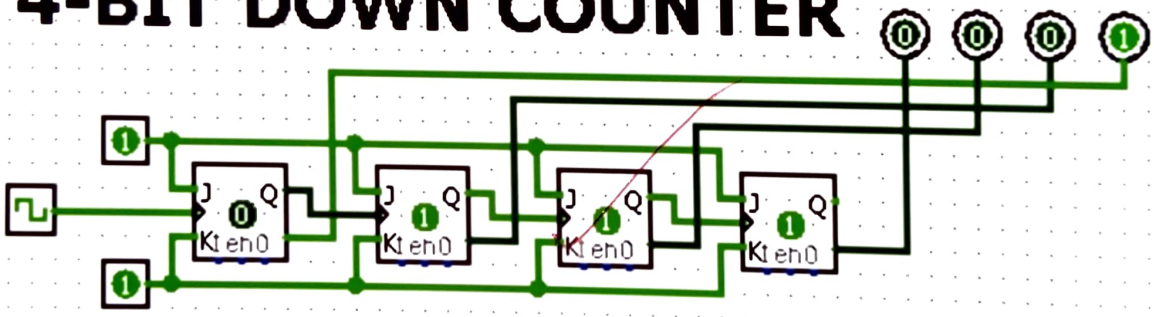
CLK	Q_D	Q_C	Q_B	Q_A	Decimal
Initial	0	0	0	0	0
1st (↓)	0	0	0	1	1
2nd (↓)	0	0	1	0	2
3rd (↓)	0	0	1	1	3
4th (↓)	0	1	0	0	4
5th (↓)	0	1	0	1	5
6th (↓)	0	1	1	0	6
7th (↓)	0	1	1	1	7
8th (↓)	1	0	0	0	8
9th (↓)	1	0	0	1	9
10th (↓)	1	0	1	0	10
11th (↓)	1	0	1	1	11
12th (↓)	0	0	0	0	0

OUTPUT :

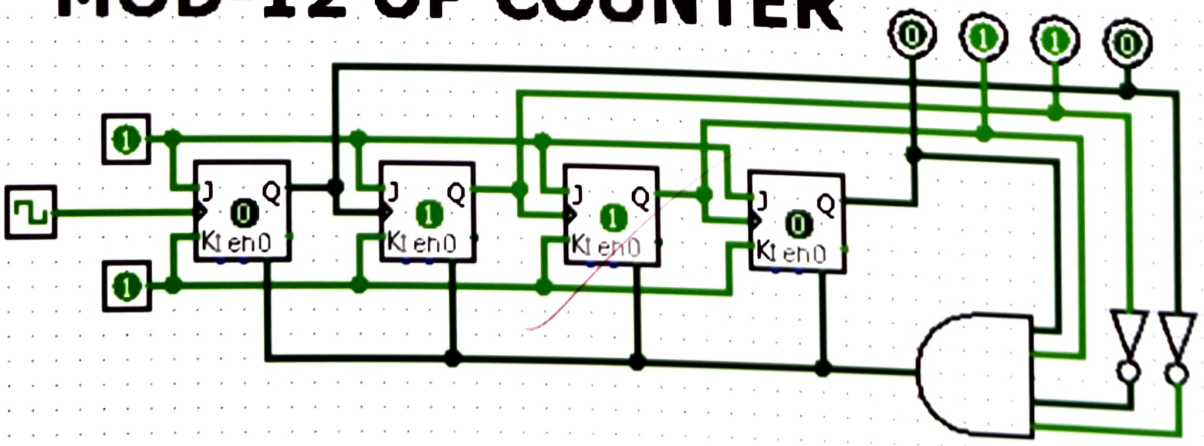
4-BIT UP COUNTER



4-BIT DOWN COUNTER



MOD-12 UP COUNTER



Topic _____ Date 12/12/22 Page No. _____

EXPERIMENT - 7

AIM : Implementation of 3-bit ripple counter (synchronous) on logisim.

SOFTWARE USED : Logisim

THEORY : Synchronous generally is type of counter in which the clock signal is simultaneously provided to each flip flop present in the counter circuit. More specifically, we can say that each flip flop is triggered in synchronous with the clock input.

→ Steps to design Synchronous Counter

1. Decide the number of flip flops
2. Excitation table of flip flops
3. State diagram and circuit excitation table
4. Obtain simplified eq" using K' map
5. Draw the logic Diagram.

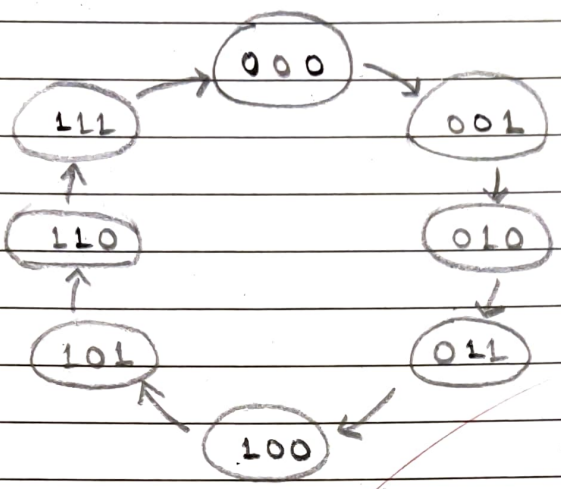
Design 3-bit Synchronous up Counter

1) Excitation table of JK flip flop.

Qn	Qn+1	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Topic _____ Date _____ Page No. _____

2) State diagram and Circuit Excitation table is

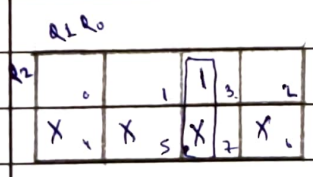


Circuit Excitation table :-

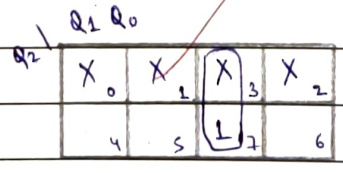
Present State			Next State								
Q_2	Q_1	Q_0	Q_2^*	Q_1^*	Q_0^*	J_2	K_2	J_1	K_1	J_0	K_0
0	0	0	0	0	1	0	X	0	X	1	X
0	0	1	0	1	0	0	X	1	X	X	1
0	1	0	0	1	1	0	X	X	0	1	X
0	1	1	1	0	0	1	X	X	1	X	1
1	0	0	1	0	1	X	0	0	X	1	X
1	0	1	1	1	0	X	0	1	X	X	1
1	1	0	1	1	1	X	0	X	0	1	X
1	1	1	0	0	0	X	1	X	1	X	1

K-map for Excitation Table

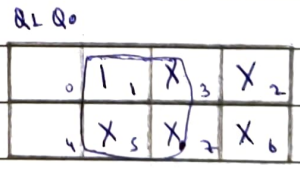
For J_2



For K_2



For J_1



$J_2 = Q_1 Q_0$ $K_2 = Q_1 Q_0$ $J_1 = Q_0$

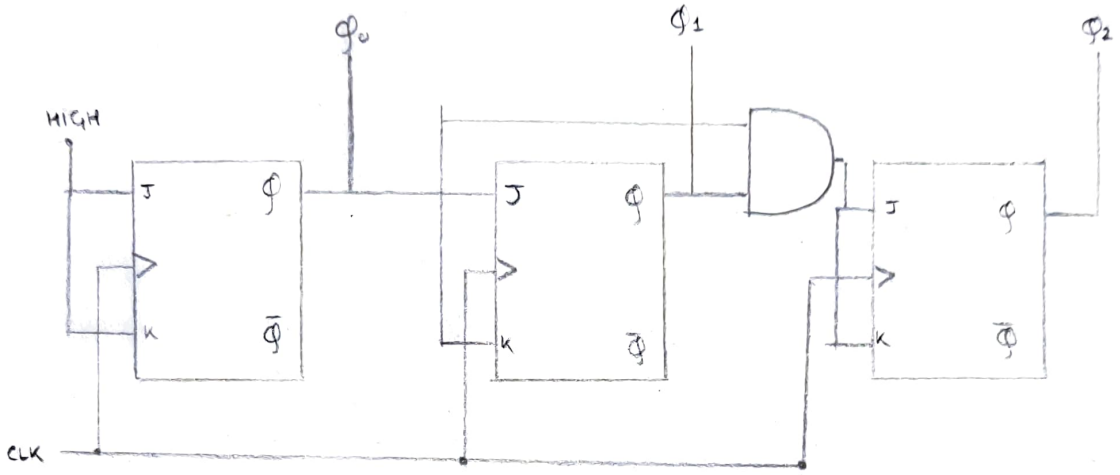


Fig. 3 BIT SYNCHRONOUS UP COUNTER

For K_1

Q_2 \ Q_1Q_0	00	01	11	10
0	X 0	X 1	1 3	2
1	X 4	X 5	1 7	6

$K_1 = Q_0$

For J_0

Q_2 \ Q_1Q_0	00	01	11	10
0	1 0	X 1	X 3	1 2
1	1 4	X 5	X 7	1 6

$J_0 = 1$

For K_0

Q_2 \ Q_1Q_0	00	01	11	10
0	X 0	1 1	1 3	X 2
1	X 4	1 5	1 7	X 6

$K_0 = 1$

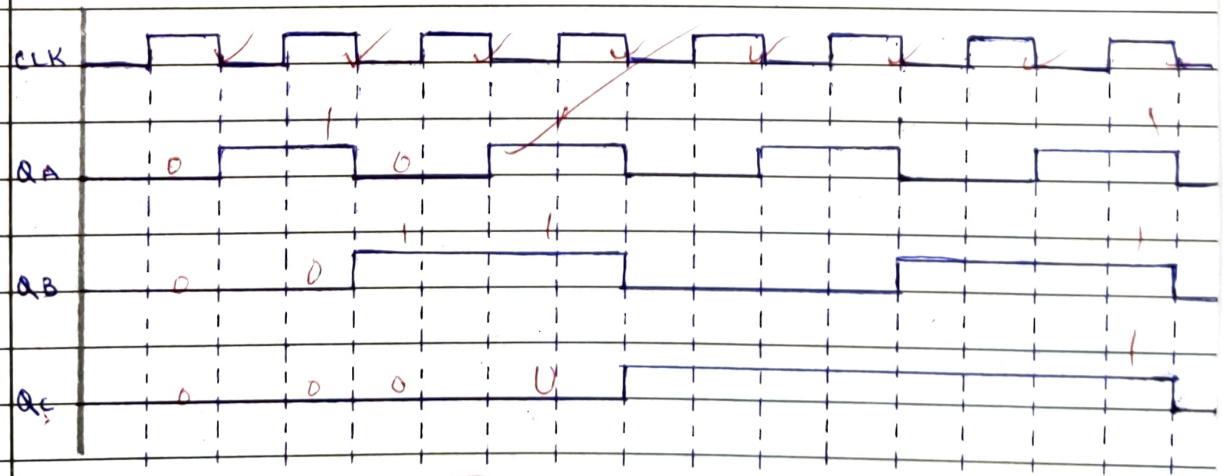
From the previous simplification we get the output as follows,

$J_0 = 1$ $\quad J_2 = Q_1 Q_0$

$K_0 = 1$ $\quad K_2 = Q_1 Q_0$

$J_1 = Q_0$

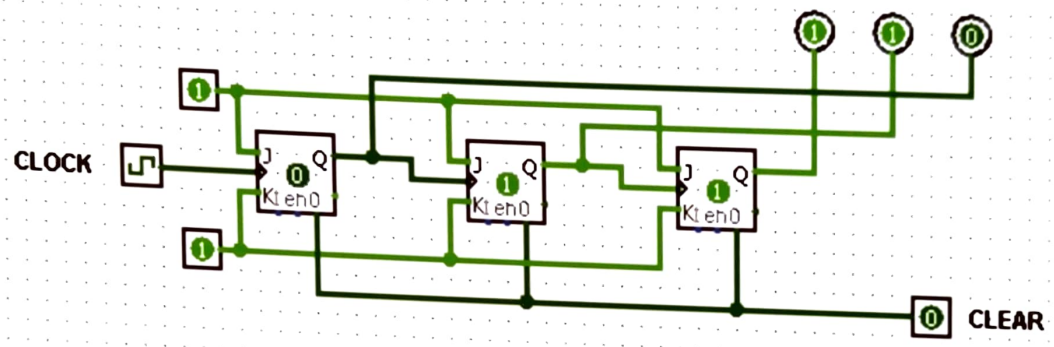
$K_1 = Q_0$



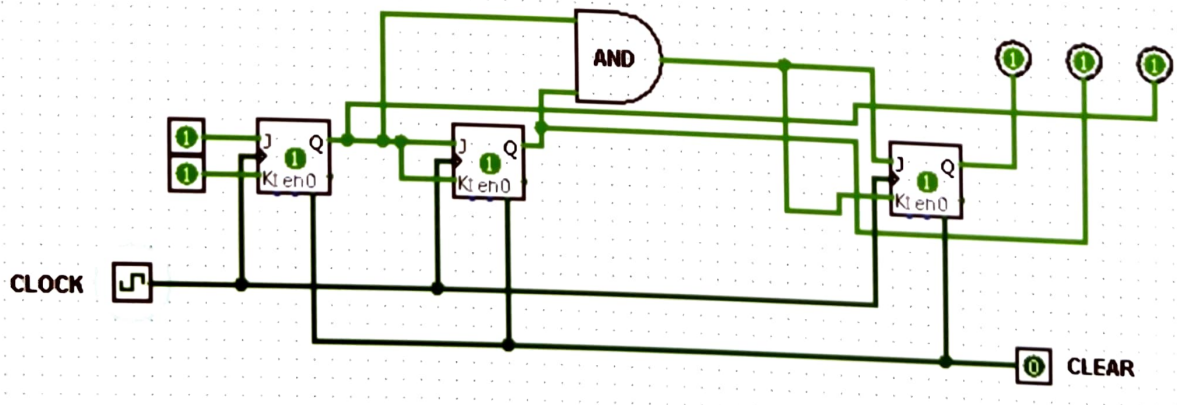
Ripple Counter

OUTPUT :

3 - BIT ASYNCHRONOUS COUNTER



3 - BIT SYNCHRONOUS COUNTER



Topic _____ Date _____ Page No. _____

EXPERIMENT - 8

AIM : Writing and executing programs in gnuSim 8085.

SOFTWARE USED : GNU Sim 8085

THEORY :

- GNUsim 8085 is a software that simulates the 8085 microprocessor. It can assemble, debug and execute 8085 assembly code.
- There are 6 general purpose registers to store data, namely B, C, D, E, H, L and an Accumulator to store the result.
- We can convert Decimal to Hex and vice versa.
- We can also manually update port values and memory.
- Some common commands

MVI => Move Immediately
MOV => Store data at a particular address
LDA => Load Accumulator
ADD => Add to the result in Accumulator
STA => Copy Accumulator contents at address
HLT => Halt.

PROGRAMS :

→ To Add two values

```
1.) MVI B, 05H
2.) MVI C, 03H
3.) ADD B
4.) ADD C
5.) HLT
```


Teacher's Sign. _____

Topic _____ Date _____ Page No. _____

→ To subtract two values

```
MVI A, 05H
MVI B, 03H
SUB B
HLT
```

→ Load values and store Addition to another address

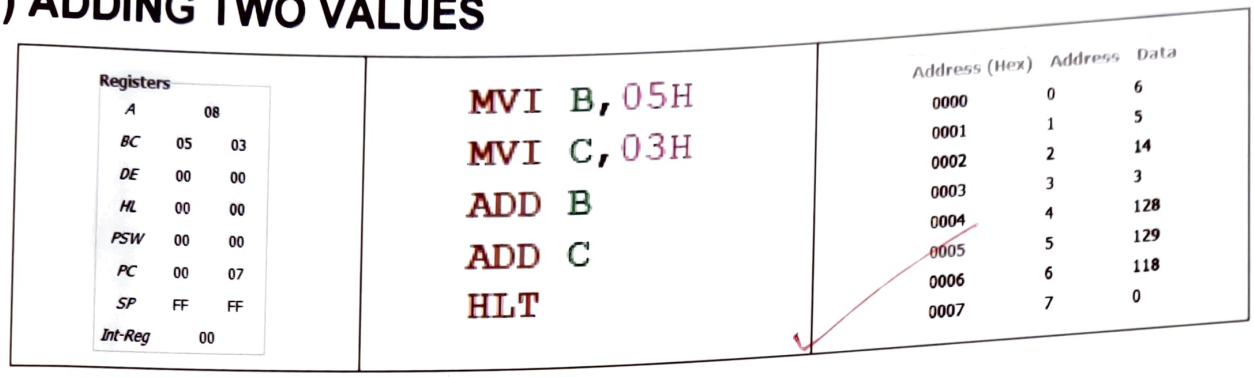
```
LDA 0000
MOV B, A
LDA 0001
ADD B
STA 0002
HLT
```

→ Load values and store subtraction to another address

```
LDA 0000
MOV B, A
LDA 0001
SUB B
STA 0002
HLT
```

OUTPUT

1) ADDING TWO VALUES



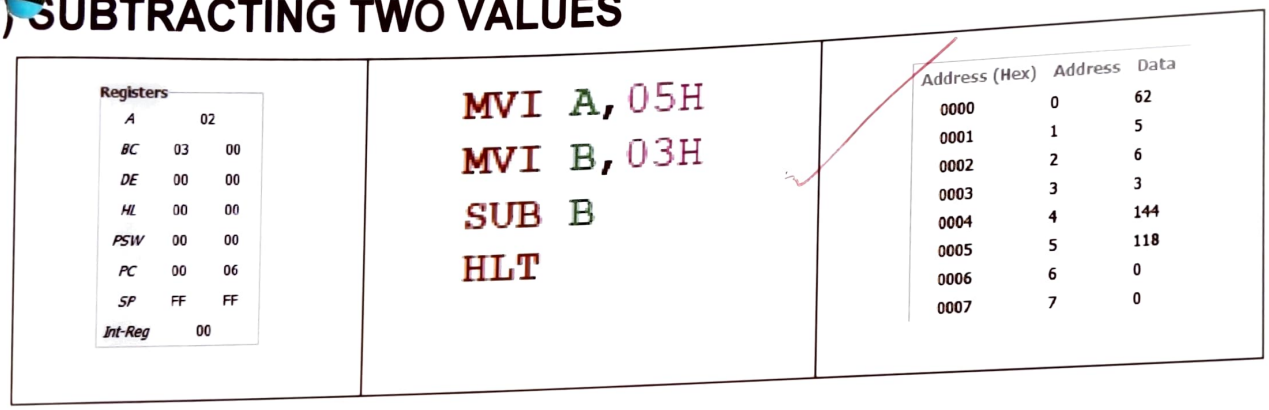
Registers

A	08	
BC	05	03
DE	00	00
HL	00	00
PSW	00	00
PC	00	07
SP	FF	FF
Int-Reg	00	

```
MVI B,05H
MVI C,03H
ADD B
ADD C
HLT
```

Address (Hex)	Address	Data
0000	0	6
0001	1	5
0002	2	14
0003	3	3
0004	4	128
0005	5	129
0006	6	118
0007	7	0

2) SUBTRACTING TWO VALUES



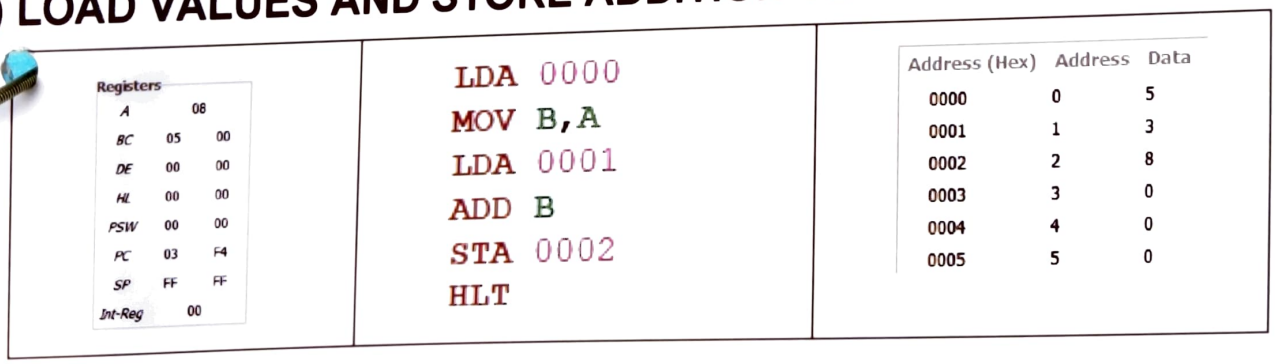
Registers

A	02	
BC	03	00
DE	00	00
HL	00	00
PSW	00	00
PC	00	06
SP	FF	FF
Int-Reg	00	

```
MVI A,05H
MVI B,03H
SUB B
HLT
```

Address (Hex)	Address	Data
0000	0	62
0001	1	5
0002	2	6
0003	3	3
0004	4	144
0005	5	118
0006	6	0
0007	7	0

3) LOAD VALUES AND STORE ADDITION TO ANOTHER ADDRESS



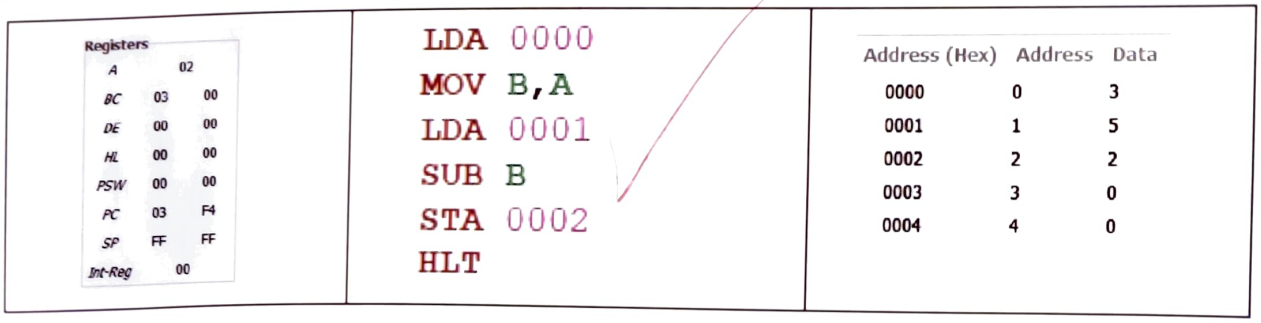
Registers

A	08	
BC	05	00
DE	00	00
HL	00	00
PSW	00	00
PC	03	F4
SP	FF	FF
Int-Reg	00	

```
LDA 0000
MOV B,A
LDA 0001
ADD B
STA 0002
HLT
```

Address (Hex)	Address	Data
0000	0	5
0001	1	3
0002	2	8
0003	3	0
0004	4	0
0005	5	0

4) LOAD VALUES & STORE SUBTRACTION TO ANOTHER ADDRES



Registers

A	02	
BC	03	00
DE	00	00
HL	00	00
PSW	00	00
PC	03	F4
SP	FF	FF
Int-Reg	00	

```
LDA 0000
MOV B,A
LDA 0001
SUB B
STA 0002
HLT
```

Address (Hex)	Address	Data
0000	0	3
0001	1	5
0002	2	2
0003	3	0
0004	4	0