A. $(B+C) = A.B+A.C$

PROCEDURE:

1. Connections are made as per the circuit/logic diagram using logisim.
2. Apply the logic inputs to the appropriate terminals of the ICs.
3. Observe the logic output for the inputs applied.
4. Verify the observed logic output with the verification/truth table given.

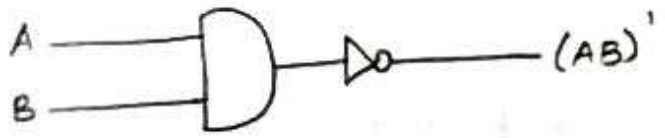
Design:

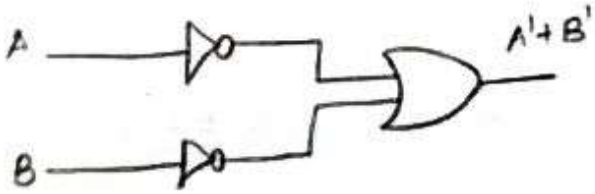
Demorgan's Theorems First Theorem: $(AB)' = A' + B'$

TRUTH TABLE:

A	B	$\bar{A}$	$\bar{B}$	AB	$\overline{AB}$	$A'+B'$
0	0	1	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	1	0	0	1	0	0

CIRCUIT DIAGRAM:





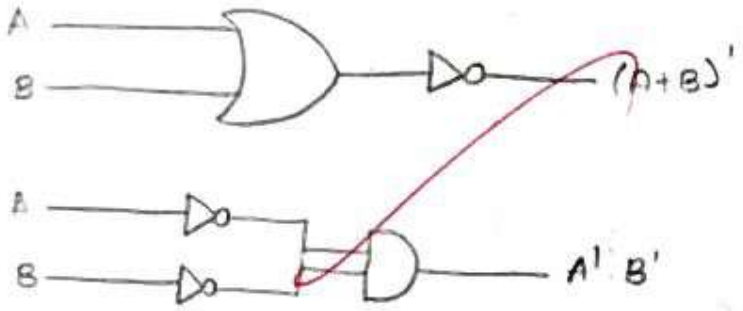
$(AB)' = A' + B'$

Demorgan's Theorems Second Theorem: $(A+B)' = A'.B'$

TRUTH TABLE:

A	B	A'	B'	A+B	(A+B)'	A' B'
0	0	1	1	0	1	1
0	1	1	0	1	0	0
1	0	0	1	1	0	0
1	1	0	0	1	0	0

CIRCUIT DIAGRAM:



$(A+B)' = A'\ B'$

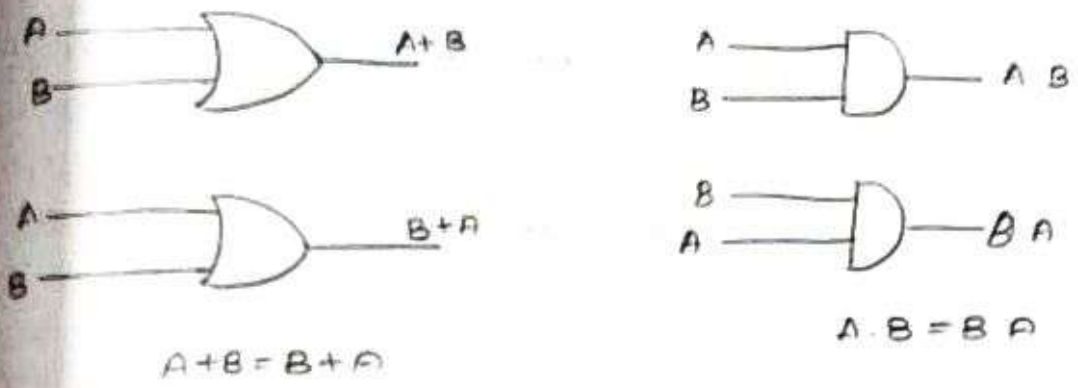
Commutative Property: $A+B = B+A$, $A \cdot B = B \cdot A$

TRUTH TABLE:

A	B	A+B	B+A
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

A	B	A.B	B.A
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

CIRCUIT DIAGRAM:

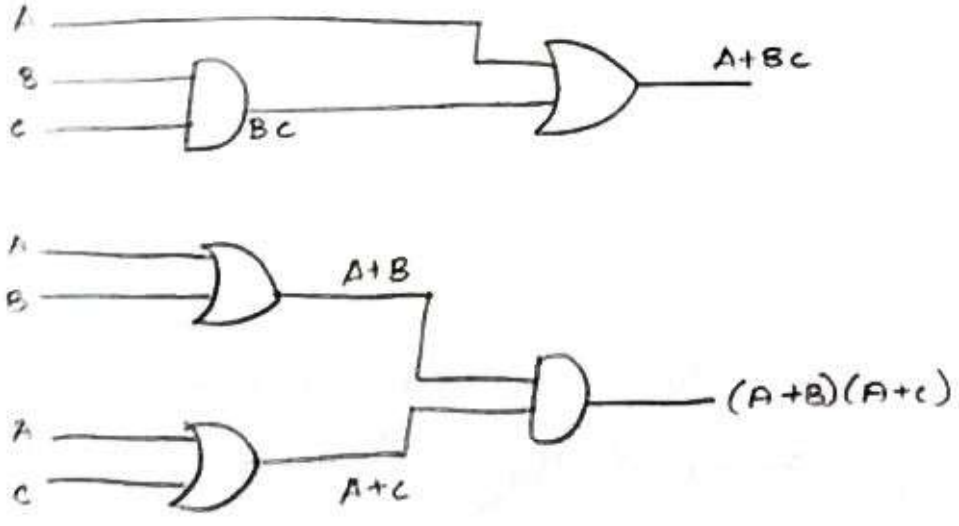


Distributive Property 1: $A + BC = (A+B)(A+C)$

TRUTH TABLE:

A	B	C	BC	A+BC	A+B	A+C	(A+B)(A+C)
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

CIRCUIT DIAGRAM:



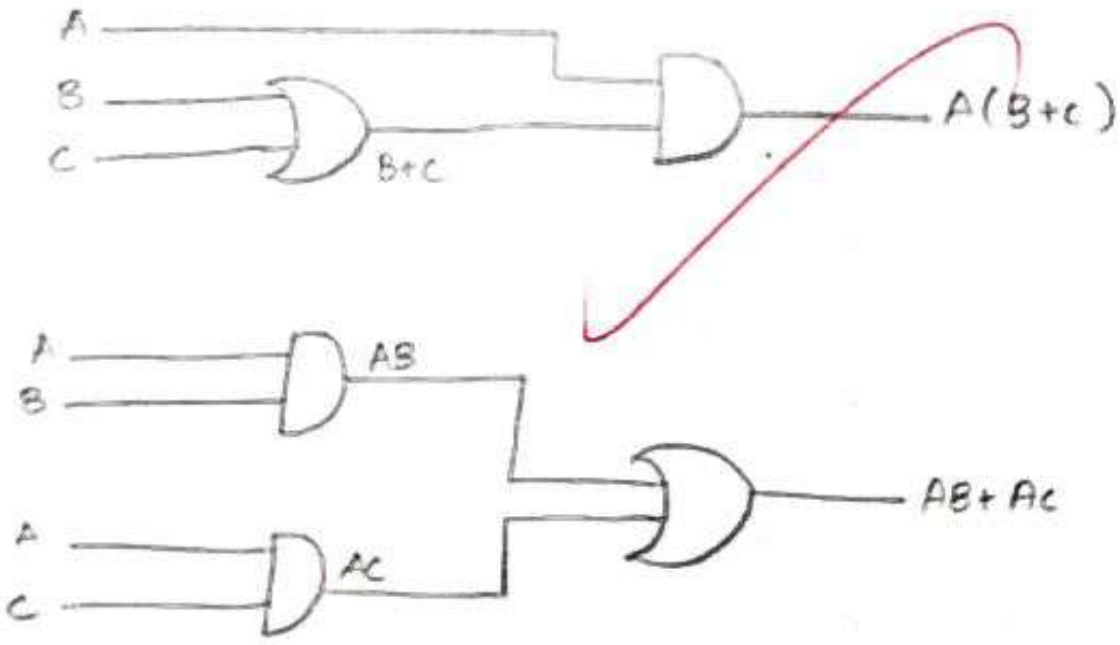
$A + BC = (A+B)(A+C)$

Distributive Property 2: $A \cdot (B + C) = A \cdot B + A \cdot C$

TRUTH TABLE:

A	B	C	B+C	A(B+C)	AB	AC	AB+AC
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

CIRCUIT DIAGRAM:



$A(B+C) = AB + AC$

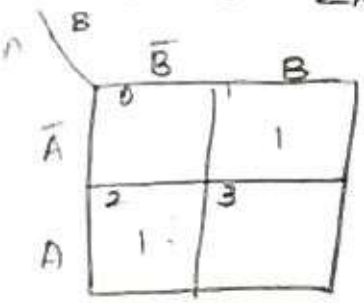
NFERENCE:

HALF ADDER:
TRUTH TABLE:

Input		Output	
A	B	Sum(s)	Carry(c)
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

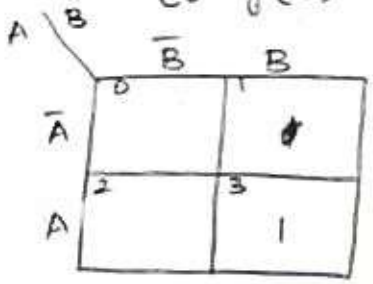
KMAP:

$Sum(S) = \Sigma m(1,2)$



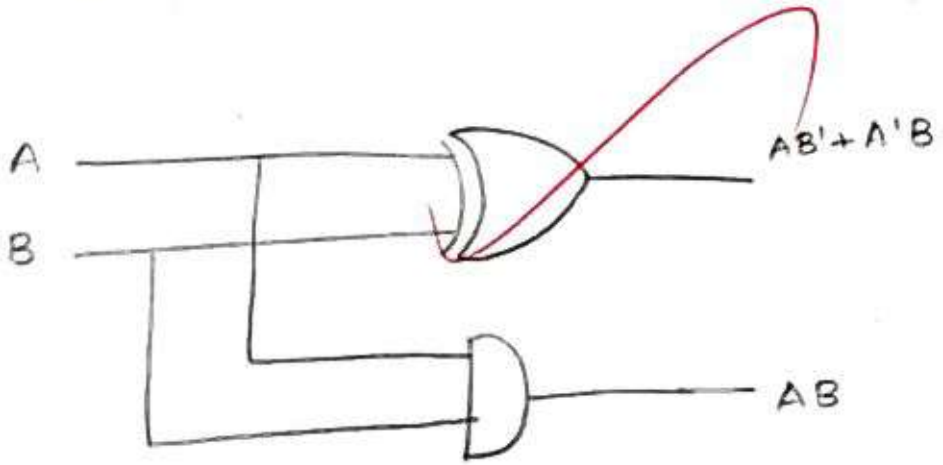
$Sum(S) = A\bar{B} + \bar{A}B = A \oplus B$

$Carry(C) = \Sigma m(3)$



$Carry(C) = AB$

CIRCUIT DIAGRAM:



FULL ADDER DESIGN PROCEDURE:

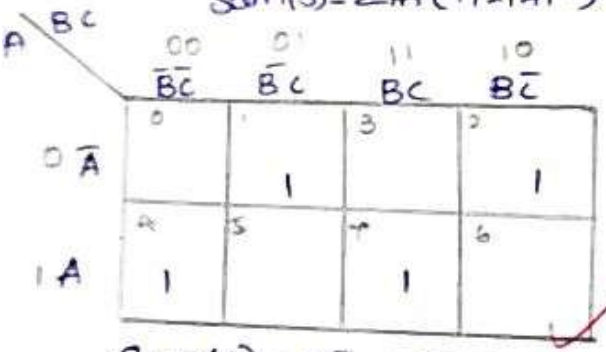
1. Define Inputs/outputs: Inputs are three bits (A, B
   Outputs are sum(S) and carry(C)
2. Logic expression: $sum(S) = A \oplus B \oplus C$
   $carry(C) = AC + BC + AB$
3. Implement Gates: Use two XOR gates for S
   two AND gates along with one OR gate fo

TRUTH TABLE:

Inputs			Outputs	
A	B	C	Sum(S)	Carry(C)
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

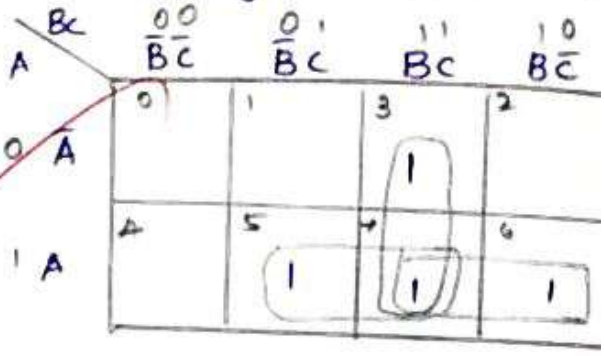
K MAP:

$Sum(S) = \Sigma m(1,2,4,7)$



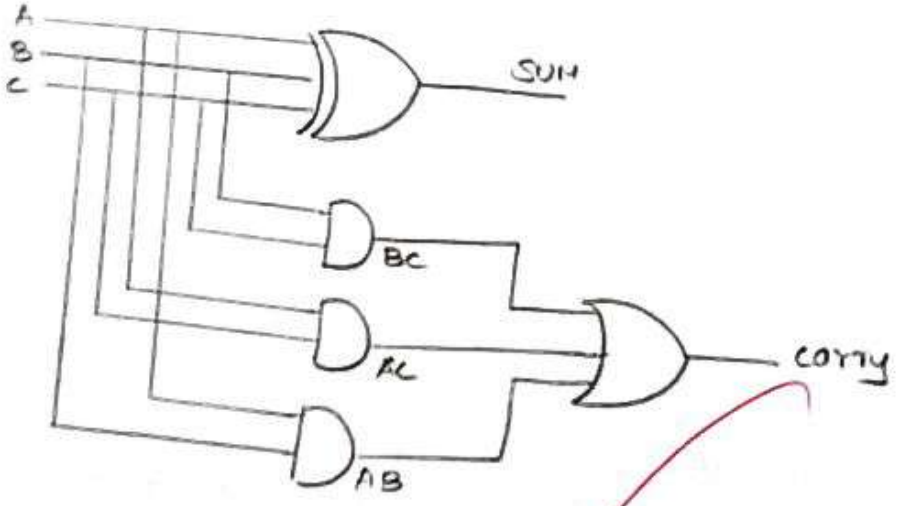
$Sum(S) = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC = A \oplus B \oplus C$

$Carry(C) = \Sigma m(3,5,6,$



$Carry(C) = AC + BC + A$

CIRCUIT DIAGRAM:

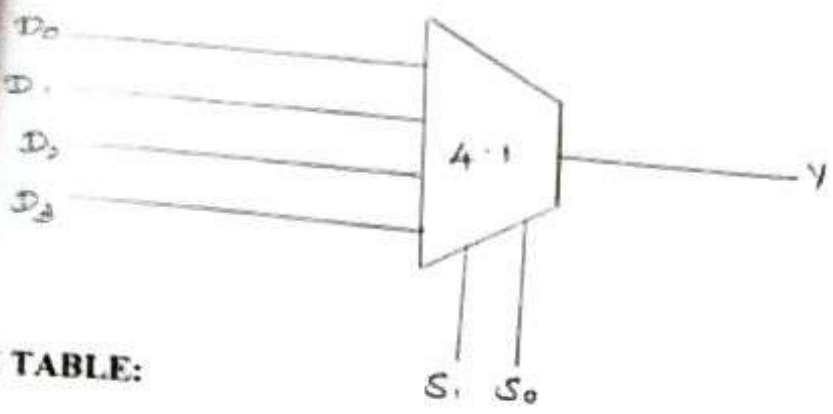


INFERENCE:

Thus, implementation of half-adder and full-adder using logic gates is verified using logisim success

MULTIPLEXER:

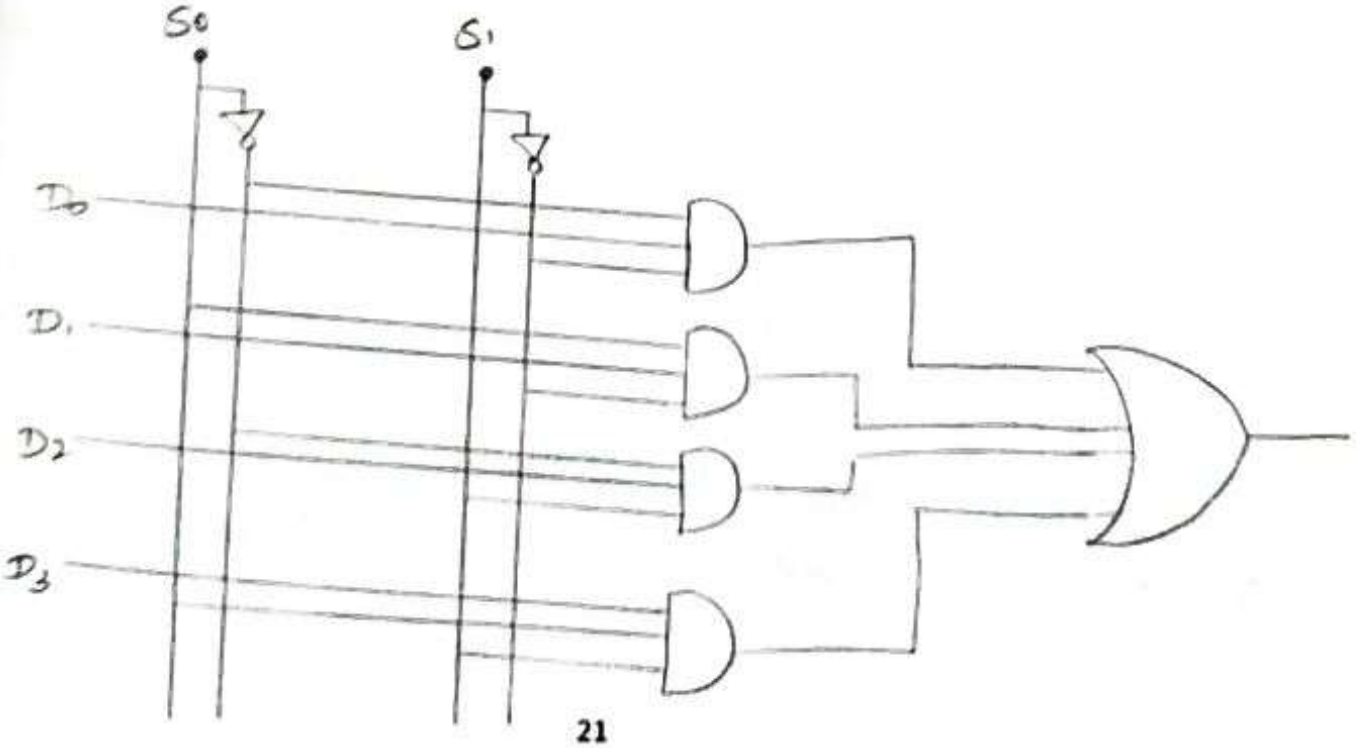
BLOCK DIAGRAM:



TRUTH TABLE:

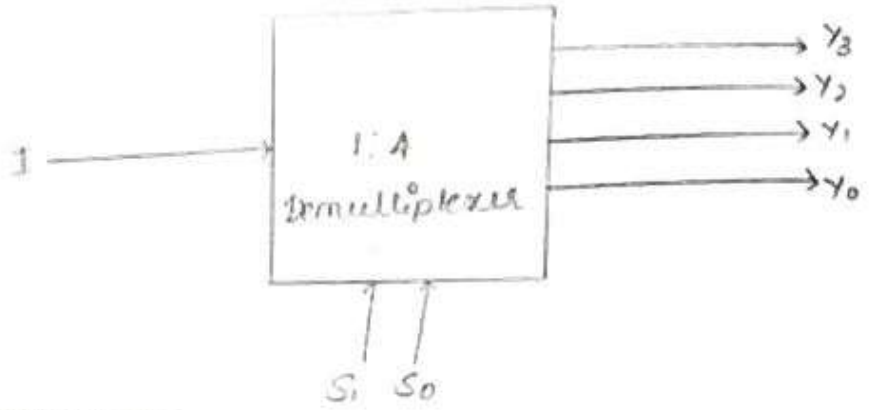
Input		Output	
S_1	S_0	Y	$\bar{S}_1\bar{S}_0D_0$
0	0	D_0	$\bar{S}_0S_0D_1$
0	1	D_1	$S_1\bar{S}_0D_2$
1	0	D_2	$S_1S_0D_3$
1	1	D_3	

CIRCUIT DIAGRAM:



DEMULTIPLEXER:

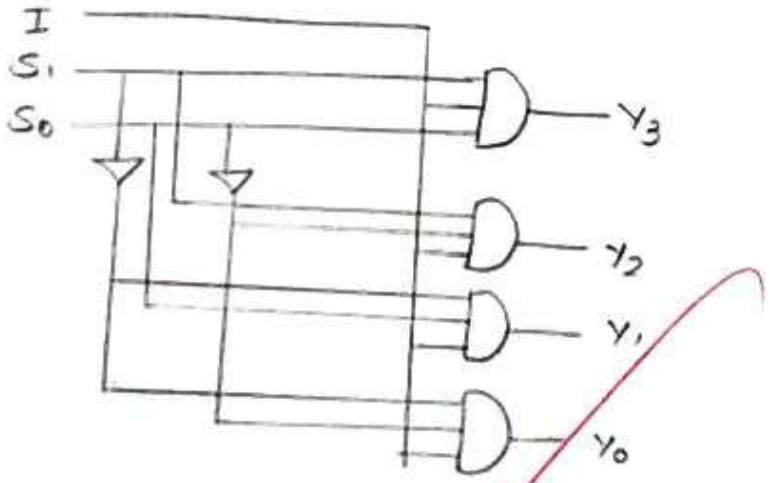
BLOCK DIAGRAM:



TRUTH TABLE:

Inputs		Outputs			
S_1	S_0	Y_3	Y_2	Y_1	Y_0
0	0	0	0	0	1
0	1	0	0	1	0
1	0	0	1	0	0
1	1	1	0	0	0

CIRCUIT DIAGRAM:



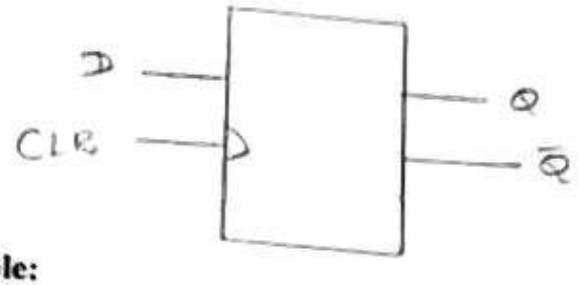
INFERENCE:

Thus, implementation of multiplexer and multiplexer verified successfully using logisim

3/10/

Realization of D flip-flop using NAND gates:

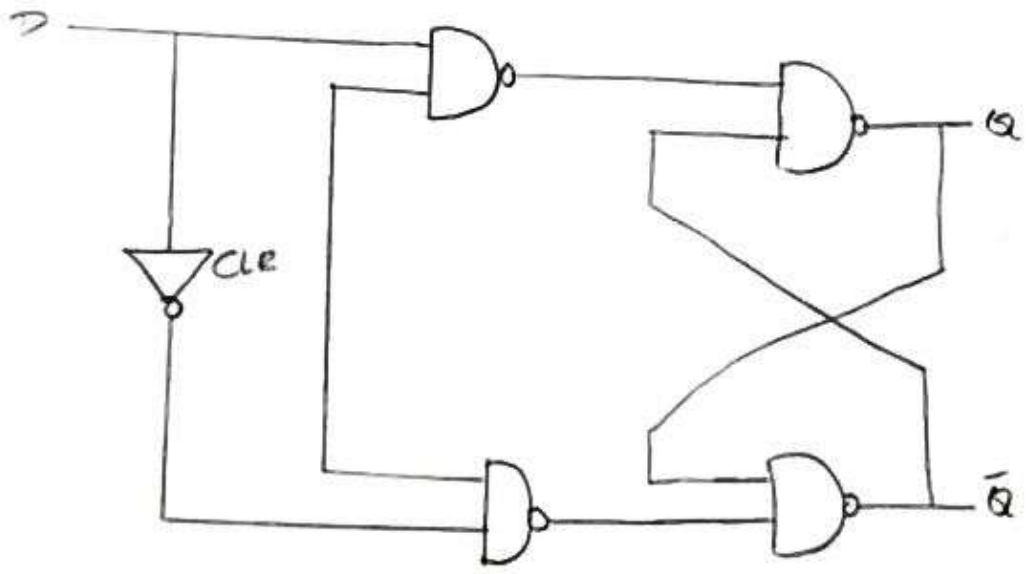
Logical Symbol:



Truth Table:

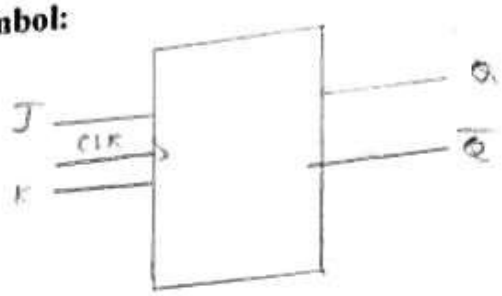
D	Q	$\bar{Q}$
0	0	0
0	1	0
1	0	1
1	1	1

Circuit Diagram:



Realization of JK flip-flop using NAND gates:

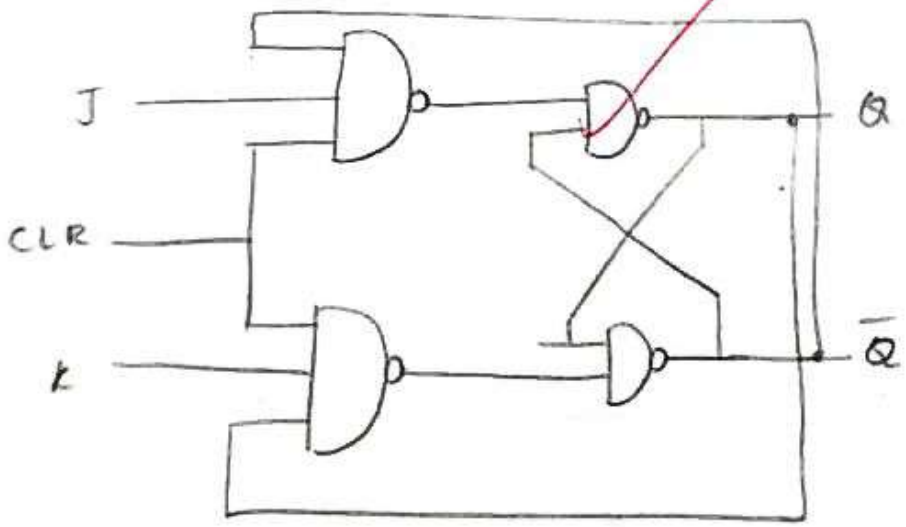
Logical Symbol:



Truth Table:

J	K	Q	$\overline{Q}$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Circuit Diagram:

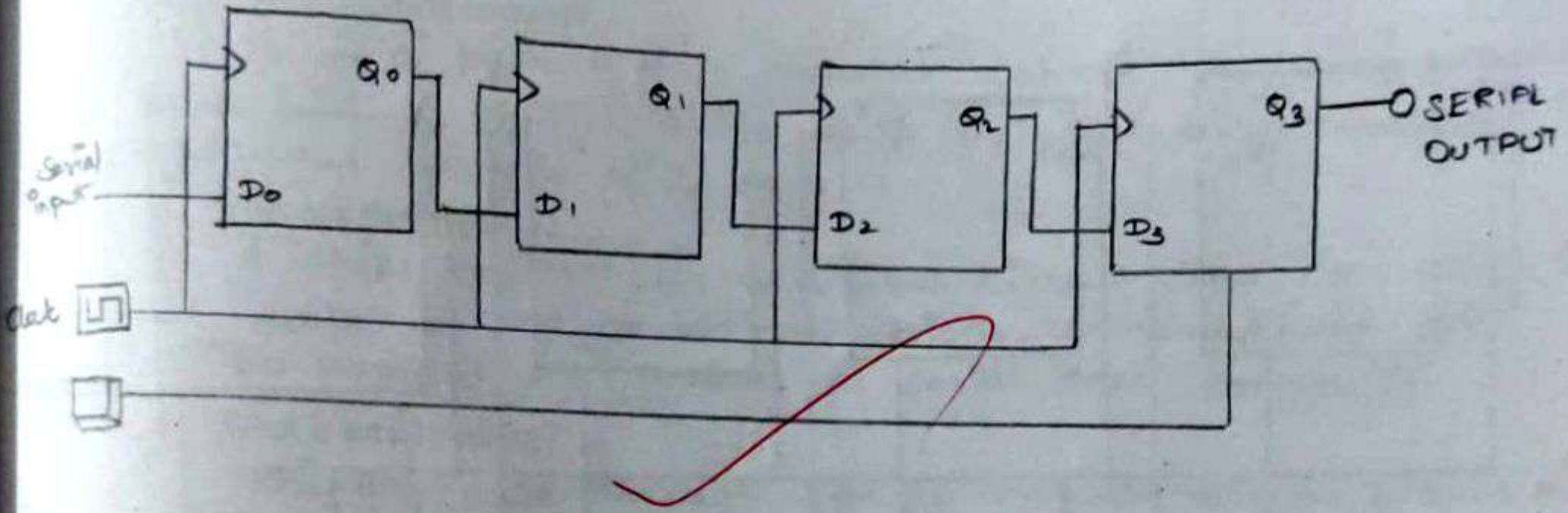


INFERENCE:

Thus, the designing and implementation of flip-flop using NAND gates using logisim successfully.

SERIAL IN AND SERIAL OUT:

CIRCUIT DIAGRAM:

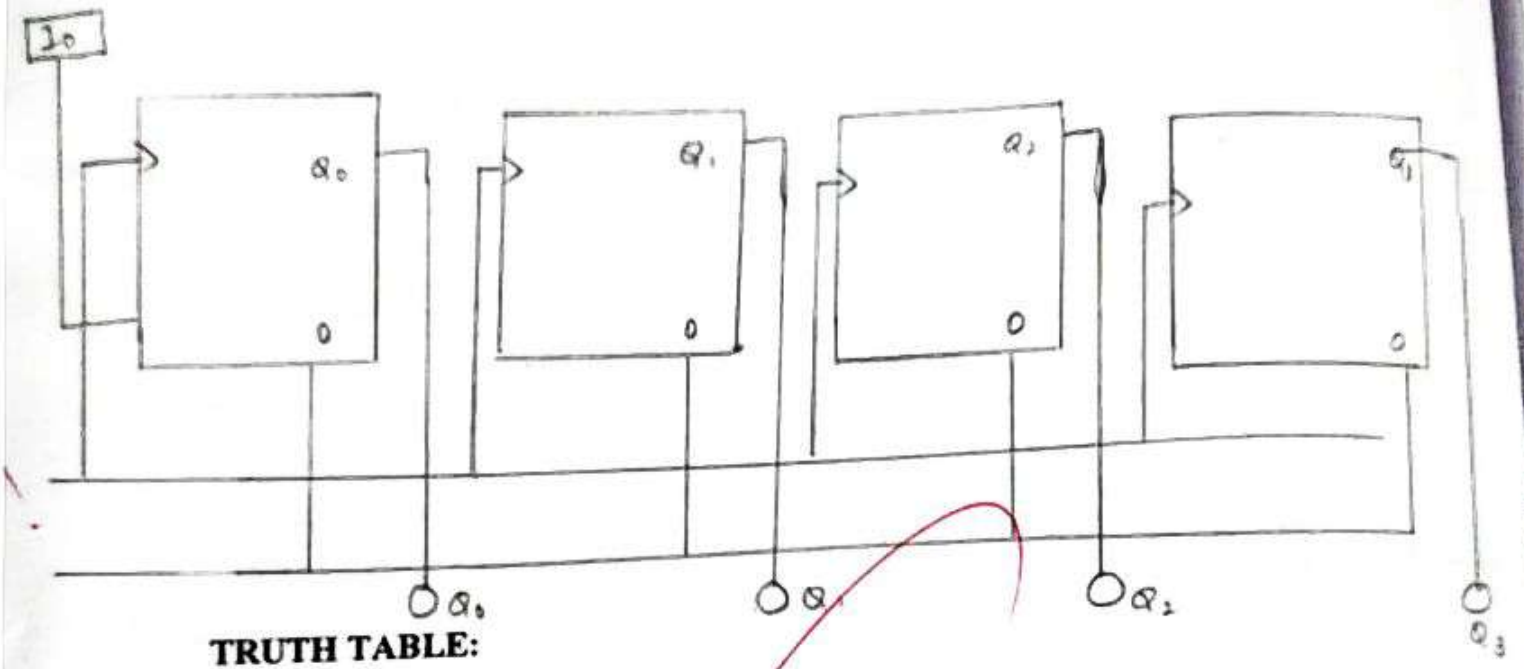


TRUTH TABLE:

clock	Q_0	Q_1	Q_2	Q_3
Initially				

PARALLEL IN AND PARALLEL OUT:

CIRCUIT DIAGRAM:



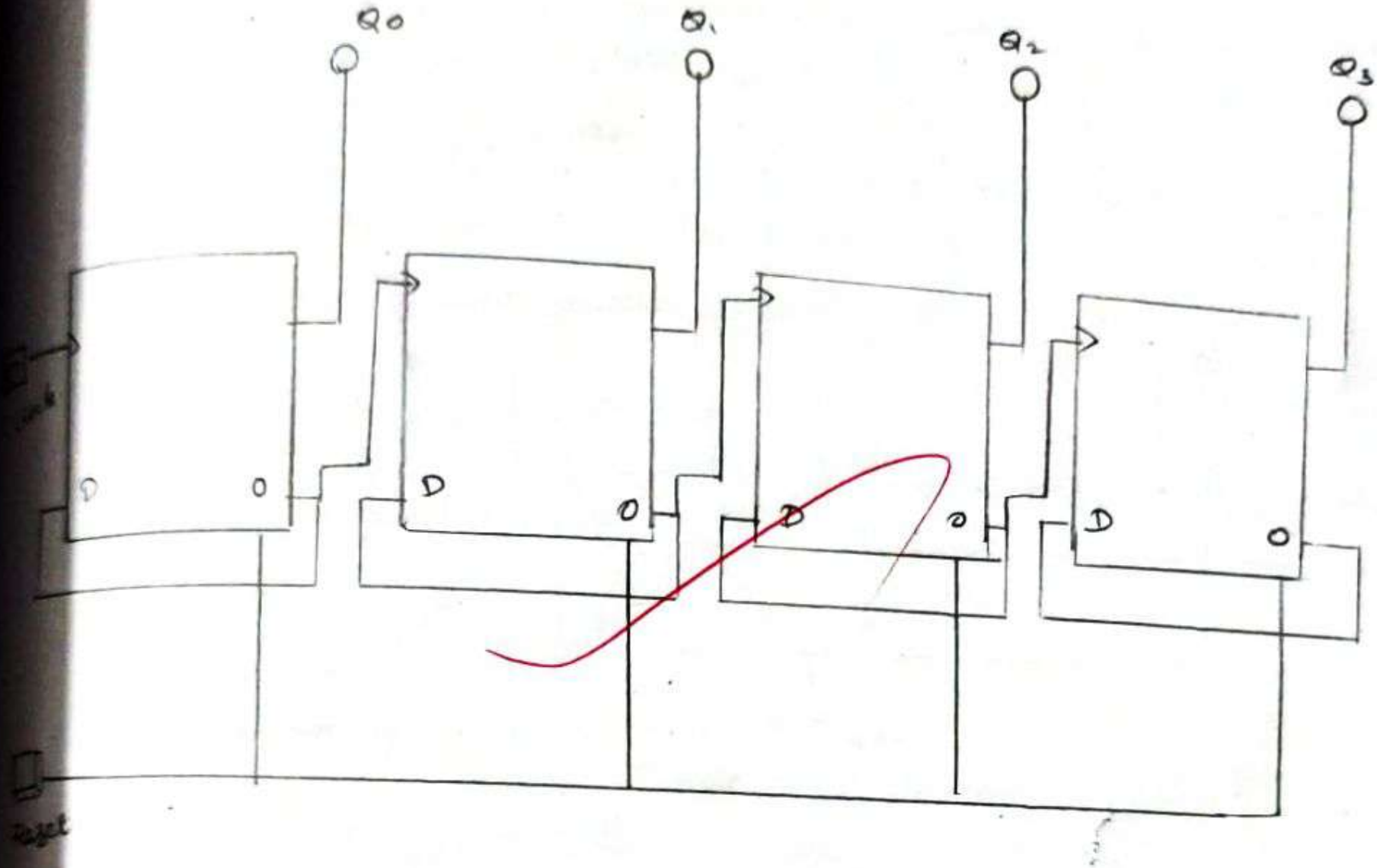
TRUTH TABLE:

Clock	Data Input				Output			
	D_0	D_1	D_2	D_3	Q_0	Q_1	Q_2	Q_3
1	1	0	0	1	1	0	0	1
2	1	0	1	0	1	0	1	0

INFERENCE: The SISO - (Serial in-Serial Out), PIPO (Parallel in - parallel Out) shift register are implemented and verified successfully using LOGISIM

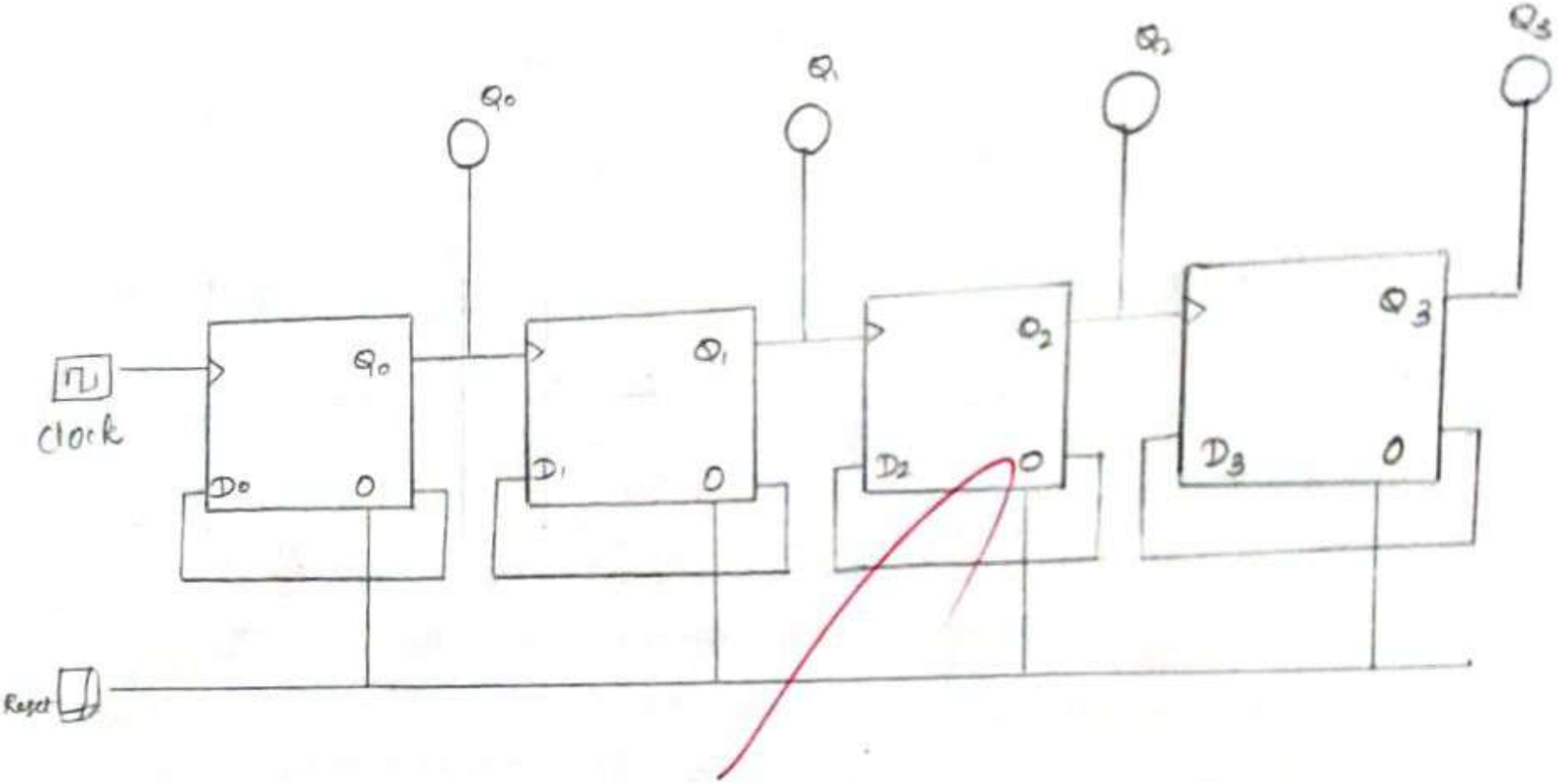
4 BIT RIPPLE UP COUNTER:

CIRCUIT DIAGRAM:



4 BIT RIPPLE DOWN COUNTER:

CIRCUIT DIAGRAM:



CIRCUIT DIAGRAM: