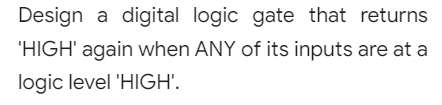
1. **LOGIC GATES**

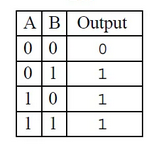
**1. PROBLEM DESCRIPTION:**



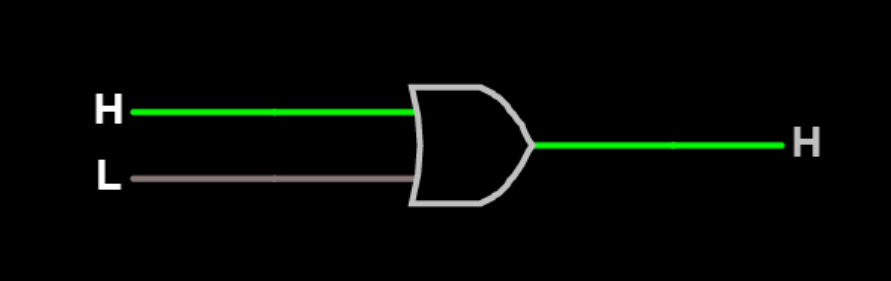
**OR GATE:**

OR gate performs logical addition. If anyone input signal is high, the output signal goes high. The output is low only when all the inputs are low.

**TRUTH TABLE:**

****

**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

0 1

**Expected Output**

1

**Test Case 2**

**Input**

1 1

**Expected Output**

1

**Test Case 3**

**Input**

0 0

**Expected Output**

0

**Test Case 4**

**Input**

1 0

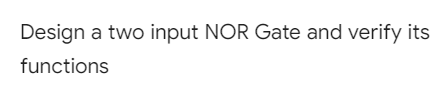
**Expected Output**

1

**RESULT:**

The circuit is successfully verified using all test cases.

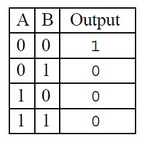
**2. PROBLEM DESCRIPTION:**



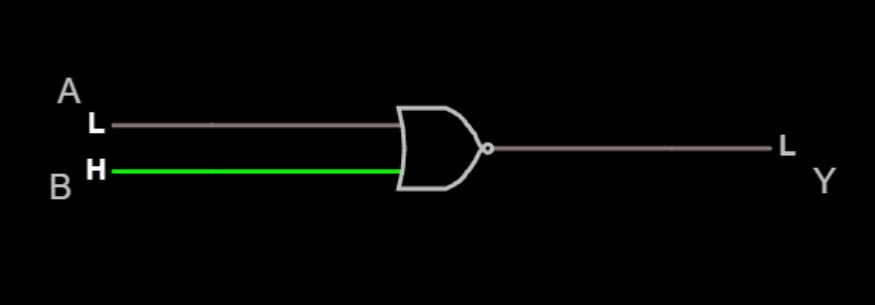
**NOR GATE:**

NOR is complement of OR gate. The output of NOR gate is low if any of the inputs are high. The output is high only when all its inputs are low. The symbol is an OR gate with a small circle on the output.

**TRUTH TABLE:**



**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

0 0

**Expected Output**

1

**Test Case 2**

**Input**

0 1

**Expected Output**

0

**Test Case 3**

**Input**

1 0

**Expected Output**

0

**Test Case 4**

**Input**

1 1

**Expected Output**

0

**RESULT:**

The circuit is successfully verified using all test cases.

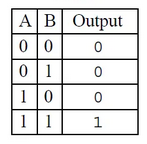
**3. PROBLEM DESCRIPTION:**



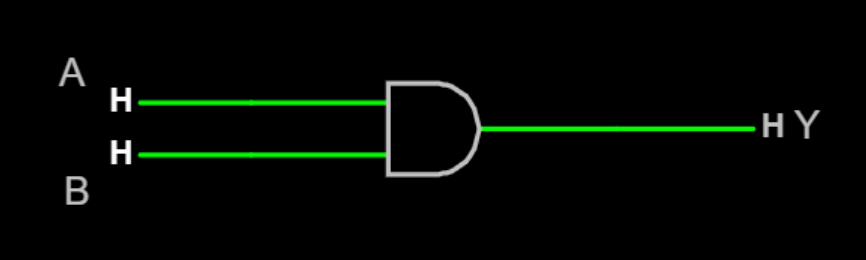
**AND GATE:**

NOR is complement of OR gate. The output of NOR gate is low if any of the inputs are high. The output is high only when all its inputs are low. The symbol is an OR gate with a small circle on the output.

**TRUTH TABLE:**



**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

0 0

**Expected Output**

0

**Test Case 2**

**Input**

0 1

**Expected Output**

0

**Test Case 3**

**Input**

1 0

**Expected Output**

0

**Test Case 4**

**Input**

1 1

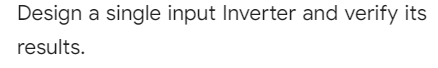
**Expected Output**

1

**RESULT:**

The circuit is successfully verified using all test cases.

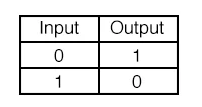
**4. PROBLEM DESCRIPTION:**



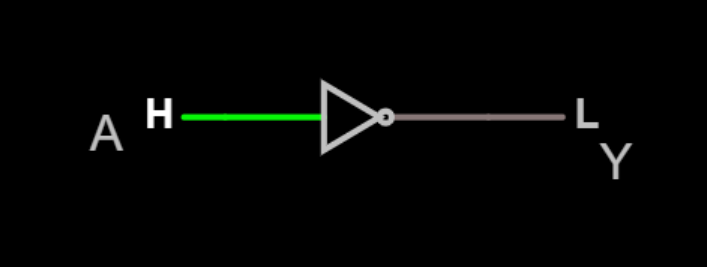
**NOT GATE (INVERTER):**

A NOT gate is basically a single input device. The output of a NOT gate is high when the input is low and the output is low when the input is high. That is, the output is the complement or inverse of the input. Hence it is also known as inverter.

**TRUTH TABLE:**



**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

0

**Expected Output**

1

**Test Case 2**

**Input**

1

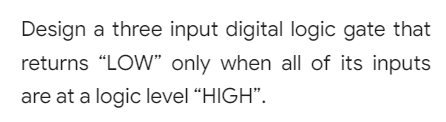
**Expected Output**

0

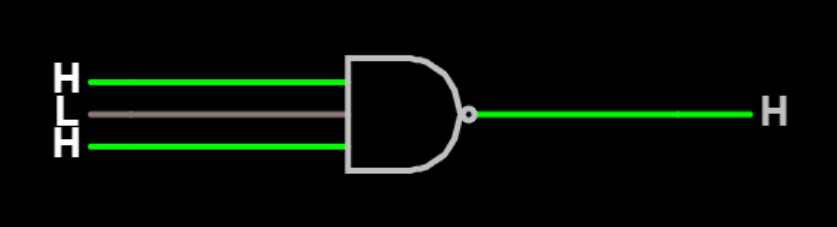
**RESULT:**

The circuit is successfully verified using all test cases.

**5. PROBLEM DESCRIPTION:**



**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

101

**Expected Output**

1

**Test Case 2**

**Input**

1 1 1

**Expected Output**

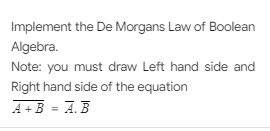
0

**RESULT:**

The circuit is successfully verified using all test cases.

1. **MINIMIZATION**

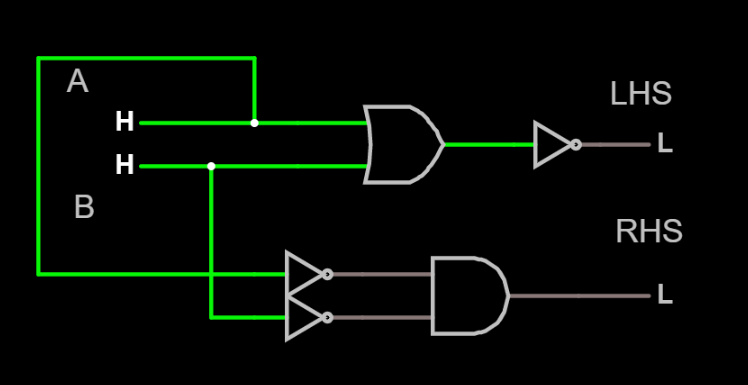
**1. PROBLEM DESCRIPTION:**



**De Morgans Law**

De Morgans Law states that the complements of the sums of all the terms are equal to the products of the complements of each and every term.

**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

0 0

**Expected Output**

1

**Test Case 2**

**Input**

0 1

**Expected Output**

0

**Test Case 3**

**Input**

1 0

**Expected Output**

0

**Test Case 4**

**Input**

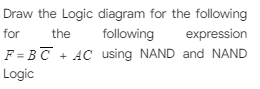
1 1

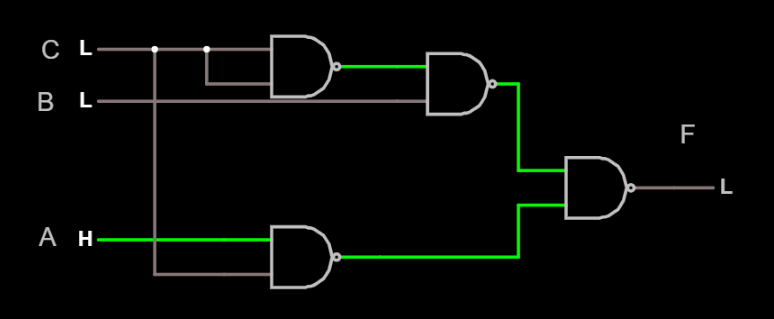
**Expected Output**

0

**RESULT:**

The circuit is successfully verified using all test cases.

**2. PROBLEM DESCRIPTION:**

**CIRCUIT DIAGRAM:**

**TEST CASES:**

**Test Case 1**

**Input**

0 0 0

**Expected Output**

0

**Test Case 2**

**Input**

0 1 0

**Expected Output**

1

**Test Case 3**

**Input**

0 0 1

**Expected Output**

0

**Test Case 4**

**Input**

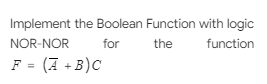
1 1 1

**Expected Output**

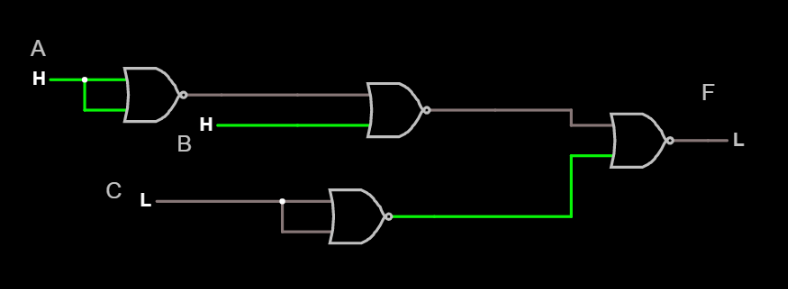
1

**RESULT:**

The circuit is successfully verified using all test cases.

**3. PROBLEM DESCRIPTION:**

**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

0 1 1

**Expected Output**

1

**Test Case 2**

**Input**

1 0 1

**Expected Output**

0

**Test Case 3**

**Input**

0 0 1

**Expected Output**

1

**Test Case 4**

**Input**

1 1 0

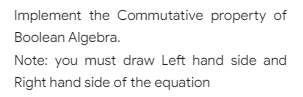
**Expected Output**

0

**RESULT:**

The circuit is successfully verified using all test cases

**4. PROBLEM DESCRIPTION:**

****

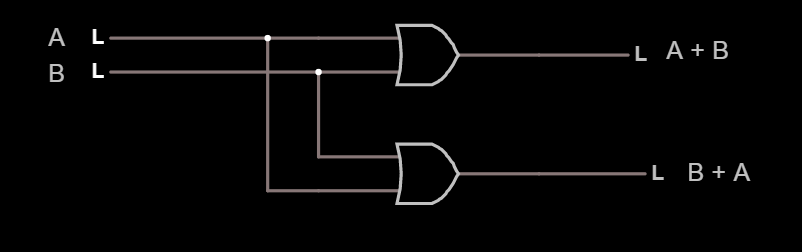
**COMMUTATIVE LAW:**

Commutative law states that changing the sequence of the variables does not have any effect on the output of a logic circuit.

**x + y = y +x**

**x . y = y . x**

**CIRCUIT DIAGRAM:**

****

**TEST CASES:**

**Test Case 1**

**Input**

**0 1**

**Expected Output**

**1**

**Test Case 2**

**Input**

**0 0**

**Expected Output**

**0**

**Test Case 3**

**Input**

**1 0**

**Expected Output**

**1**

**Test Case 4**

**Input**

**1 1**

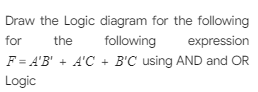
**Expected Output**

**1**

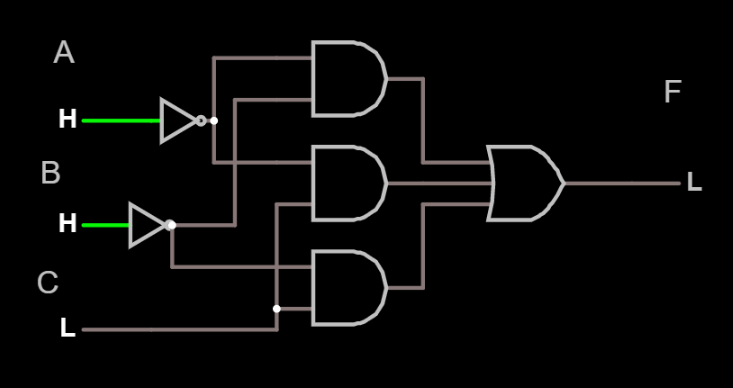
**RESULT:**

The circuit is successfully verified using all test cases.

**5. PROBLEM DESCRIPTION:**



**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

1 0 1

**Expected Output**

1

**Test Case 2**

**Input**

1 1 0

**Expected Output**

0

**Test Case 3**

**Input**

0

**Expected Output**

0

**Test Case 4**

**Input**

0

**Expected Output**

0

**RESULT:**

The circuit is successfully verified using all test cases.

**3. COMBINATIONAL CIRCUIT**

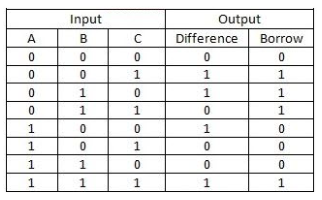
**1. PROBLEM DESCRIPTION:**

Design the subtractor circuit that has 2 half subtractor circuits

**FULL SUBTRACTOR:**

A full subtractor performs subtraction operation on two bits, a minuend ‘A’ and a subtrahend ‘B’, and also takes into consideration whether a 1 has already been borrowed by the previous adjacent lower minuend bit or not ‘C’. Hence, there are three bits to be handled at the input of a full subtractor, namely the two bits to be subtracted and a borrow bit. There are two outputs, namely the DIFFERENCE ‘D’ and the BORROW ‘Bo’. The BORROW output bit tells whether the minuend bit needs to borrow from the next possible higher minuend bit.

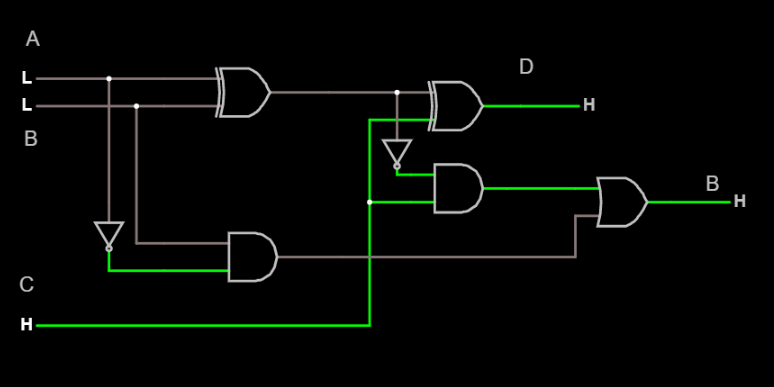
**TRUTH TABLE:**

****

***Difference (D) = A ⊕ B ⊕ C***

***Borrow (Bo) = (A ⊕ B)’ C + A’B***

**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

1 0 0

**Expected Output**

difference = 1

borrow = 0

**Test Case 2**

**Input**

1 1 0

**Expected Output**

difference = 0

borrow = 0

**Test Case 3**

**Input**

0 0 1

**Expected Output**

difference = 1

borrow = 1

**Test Case 4**

**Input**

0 0 0

**Expected Output**

difference = 0

borrow = 0

**RESULT:**

The circuit is successfully verified using all test cases.

**2. PROBLEM DESCRIPTION:**

Implement 2 to 4 binary Decoder using logic gates  
Note :  
Inputs are E(Enable), A, B

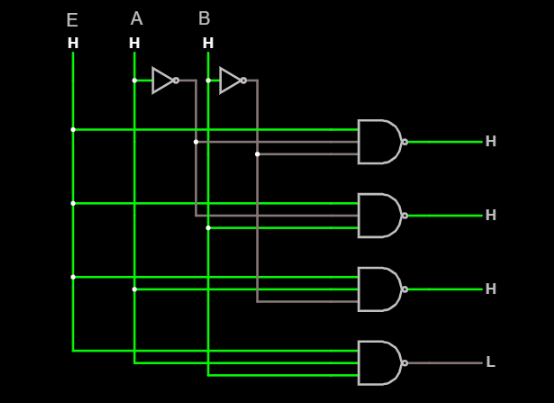
**DECODER:**

A decoder is a combinational circuit that converts binary information from n input lines to a maximum of 2n unique output lines. Given a binary code of n bits, a decoder will tell which code is this out of the 2n possible output lines.

**TRUTH TABLE OF 2-TO-4 LINE DECODER:**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **E** | **A** | **B** | **D0** | **D1** | **D2** | **D3** |
| **0** | **X** | **X** | **1** | **1** | **1** | **1** |
| **1** | **0** | **0** | **0** | **1** | **1** | **1** |
| **1** | **0** | **1** | **1** | **0** | **1** | **1** |
| **1** | **1** | **0** | **1** | **1** | **0** | **1** |
| **1** | **1** | **1** | **1** | **1** | **1** | **0** |

**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

1 0 0

**Expected Output**

0 1 1 1

**Test Case 2**

**Input**

1 1 0

**Expected Output**

1 1 0 1

**Test Case 3**

**Input**

1 0 1

**Expected Output**

1 0 1 1

**Test Case 4**

**Input**

1 1 1

**Expected Output**

1 1 1 0

**RESULT:**

The circuit is successfully verified using all test cases.

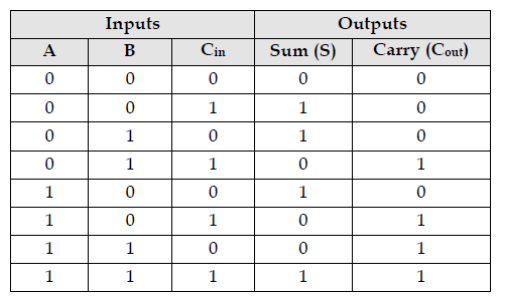
**3. PROBLEM DESCRIPTION:**

Design the Full Adder circuit using two half adder circuits

**FULL ADDER:**

A full adder is a combinational circuit that forms the arithmetic sum of three input bits. It consists of 3 inputs and 2 outputs. Two of the input variables, represent the significant bits to be added. The third input represents the carry from previous lower significant position. The output variables represent Sum and Carry

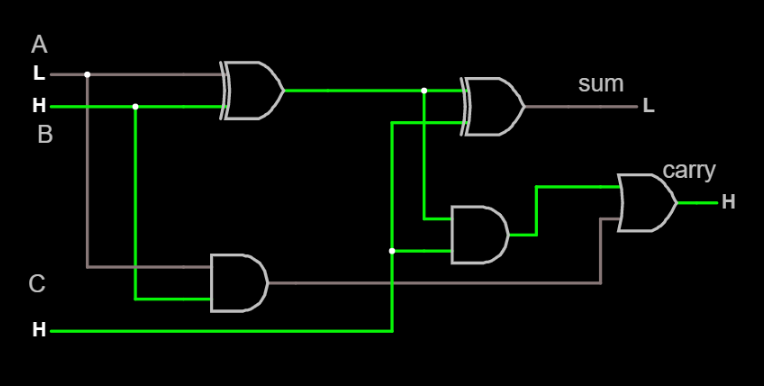
**TRUTH TABLE:**

****

**Sum (S) =  *A ⊕ B ⊕ Cin***

***Carry (Cout) = (A ⊕ B) Cin + AB***

**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

1 0 1

**Expected Output**

sum = 0

carry = 1

**Test Case 2**

**Input**

1 1 1

Expected Output

sum = 1

carry = 1

**Test Case 3**

**Input**

0 0 0

**Expected Output**

sum=0

carry=0

**Test Case 4**

**Input**

0 1 1

**Expected Output**

sum=0

carry=1

**RESULT:**

The circuit is successfully verified using all test cases.

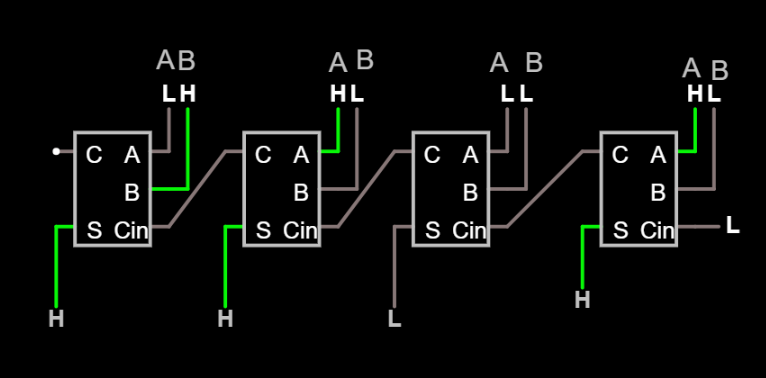
**4. PROBLEM DESCRIPTION:**

Design a combinational circuit of 4 bit Parallel Adder using full adders

**BINARY PARALLEL ADDER:**

A binary parallel adder is a digital circuit that adds two binary numbers and produces the arithmetic sum of those numbers in parallel form. It can be constructed with full adders connected in cascade with the output carry form each full adder connected to the input carry of the next full adder in the chain.

**CIRCUIT DIAGRAM:**



**Test Cases:**

**Test Case 1**

**Input**

A=0 0 0 0

B=0 0 0 1

**Expected Output**

S=0 0 0 1

**Test Case 3**

**Input**

A=1 0 1 0

B=0 1 0 1

**Expected Output**

S=1 1 1 1

**Test Case 2**

**Input**

A=0 0 1 0

B=0 0 0 1

**Expected Output**

S=0 0 1 1

**Test Case 4**

**Input**

A=1 0 1 0

B=0 0 0 1

**Expected Output**

S=1 0 1 1

**RESULT:**

The circuit is successfully verified using all test cases.

**5. PROBLEM DESCRIPTION:**

Design a 4 to 1 Line Multiplexer circuit using logic gates

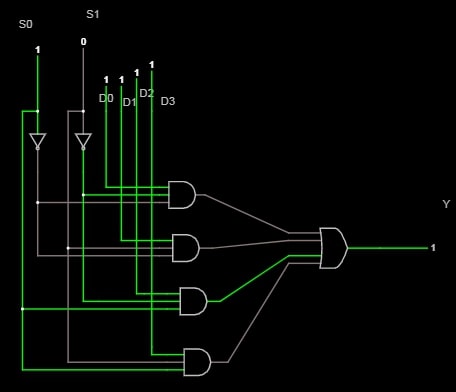
**4 to 1 Line MULTIPLEXER:**

4x1 Multiplexer has four data inputs D3, D2, D1 & D0, two selection lines S1 & S0 and one output Y

**TRUTH TABLE:**

|  |  |  |
| --- | --- | --- |
| **Selection Inputs** | | **Output** |
| **S1** | **S2** | **Y** |
| 0 | 0 | D0 |
| 0 | 1 | D1 |
| 1 | 0 | D2 |
| 1 | 1 | D3 |

**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

0 X X

**Expected Output**

0

**Test Case 2**

**Input**

1 0 0

**Expected Output**

D0

**Test Case 3**

**Input**

1 0 1

**Expected Output**

D1

**Test Case 4**

**Input**

1 1 0

**Expected Output**

D2

**RESULT:**

The circuit is successfully verified using all test cases.

**6. PROBLEM DESCRIPTION:**

Implement 1:4 DeMultiplexer

Use Input as Q, S1, S2 and Outputs are D0,D1,D2,D3

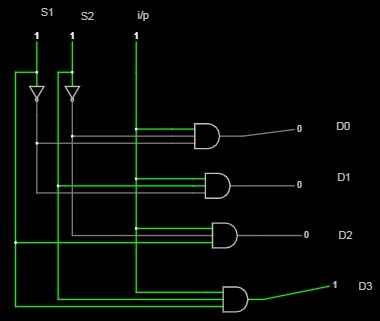
**1:4 DeMULTIPLEXER:**

1:4 DeMultiplexer has single input, ‘2’ selection lines and 4 outputs. The input will be connected to one of these outputs based on the values of selection lines

**TRUTH TABLE:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Selection Inputs** | | **Outputs** | | | |
| **S1** | **S2** | **D3** | **D2** | **D1** | **D0** |
| 0 | 1 | 0 | 0 | 0 | Q |
| 0 | 1 | 0 | 0 | Q | 0 |
| 1 | 0 | 0 | Q | 0 | 0 |
| 1 | 1 | Q | 0 | 0 | 0 |

**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

X X X

**Expected Output**

0 0 0 0

**Test Case 2**

**Input**

0 0 0

**Expected Output**

0 0 0 0

**Test Case 3**

**Input**

1 0 0

**Expected Output**

1 0 0 0

**Test Case 4**

**Input**

1 1 1

**Expected Output**

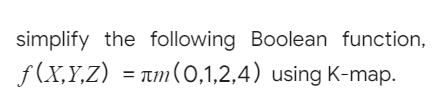
0 0 0 1

**RESULT:**

The circuit is successfully verified using all test cases.

1. **KARNAUGH MAP**

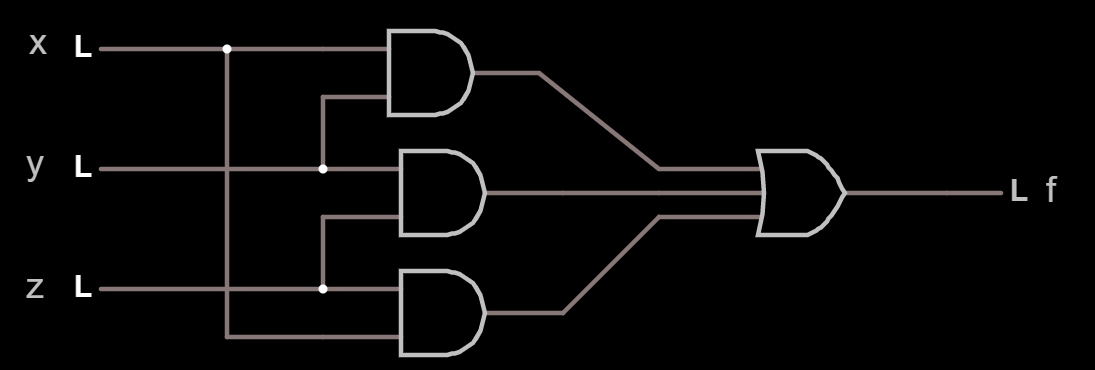
**1. PROBLEM DESCRIPTION:**



**SIMPLIFICATION USING K-MAP:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Y’Z’** | **Y’Z** | **YZ** | **YZ’** |
| **X’** |  |  |  |  |
|  | **0** | **1** | **3** | **2** |
| **X** |  |  |  |  |
|  | **4** | **5** | **7** | **6** |

**CIRCUIT DIAGRAM:**

****

**TEST CASES:**

**Test Case 1**

**Input**

X=0,Y=0,Z=0

**Expected Output**

out=0

**Test Case 2**

**Input**

X=0,Y=1,Z=0

**Expected Output**

out=0

**Test Case 3**

**Input**

X=0,Y=1,Z=1

**Expected Output**

out=1

**Test Case 4**

**Input**

X=1,Y=1,Z=1

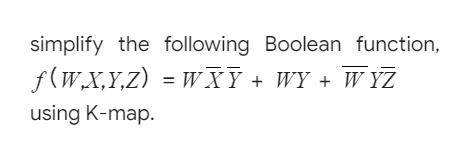
**Expected Output**

**out=1**

**RESULT :**

The circuit is successfully verified using all test cases.

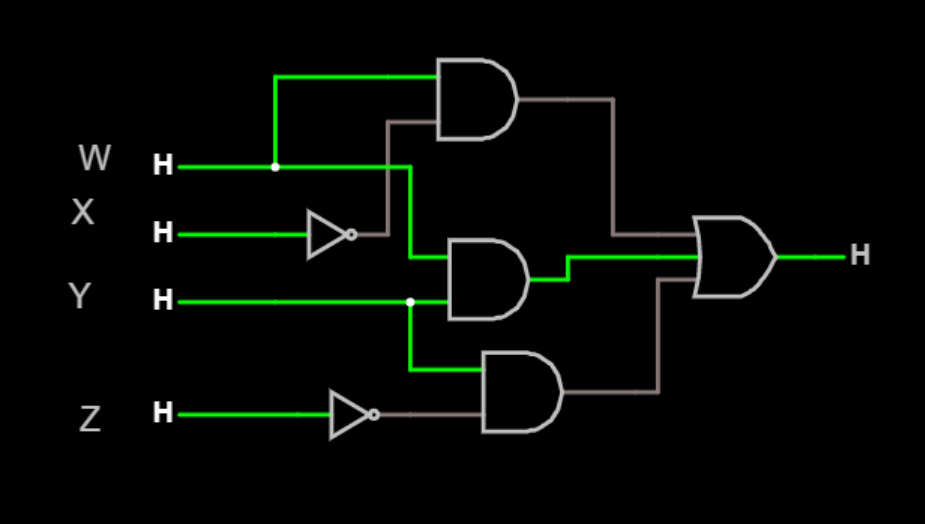
**2. PROBLEM DESCRIPTION:**

****

**SIMPLIFICATION USING K-MAP:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Y’Z’** | **Y’Z** | **YZ** | **YZ’** |
| **W’X’** |  |  |  |  |
|  | **0** | **1** | **3** | **2** |
| **W’X** |  |  |  |  |
|  | **4** | **5** | **7** | **6** |
| **WX** |  |  |  |  |
|  | **12** | **13** | **15** | **14** |
| **WX’** |  |  |  |  |
|  | **8** | **9** | **11** | **10** |

**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

W=L,X=L,Y=L,Z=L

**Expected Output**

OUT=L

**Test Case 2**

**Input**

W=L,X=H,Y=L,Z=H

**Expected Output**

OUT=L

**Test Case 3**

**Input**

W=L,X=H,Y=H,Z=L

**Expected Output**

OUT=H

**Test Case 4**

**Input**

W=H,X=H,Y=H,Z=H

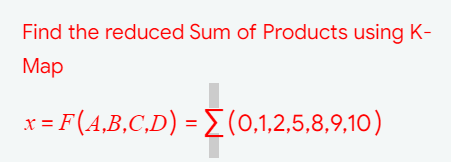
**Expected Output**

OUT=H

**RESULT:**

The circuit is successfully verified using all test cases.

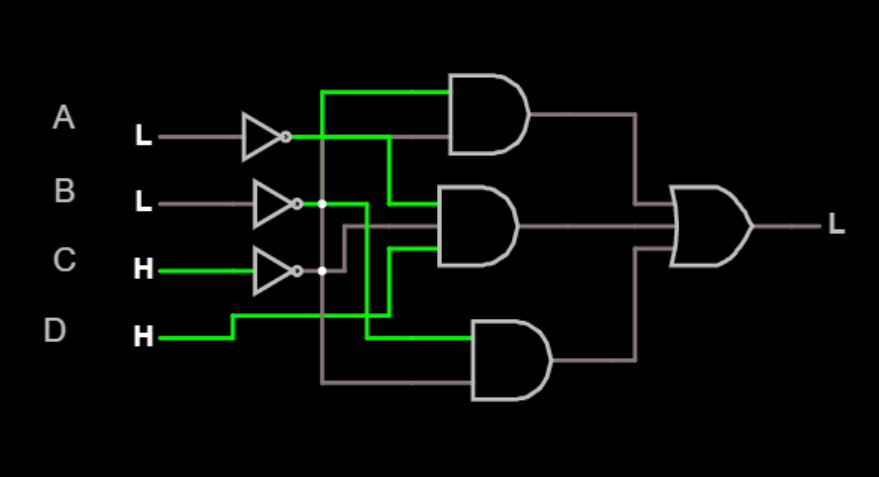
**3. PROBLEM DESCRIPTION:**

****

**SIMPLIFICATION USING K-MAP:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **C’D’** | **C’D** | **CD** | **CD’** |
| **A’B’** |  |  |  |  |
|  | **0** | **1** | **3** | **2** |
| **A’B** |  |  |  |  |
|  | **4** | **5** | **7** | **6** |
| **AB** |  |  |  |  |
|  | **12** | **13** | **15** | **14** |
| **AB’** |  |  |  |  |
|  | **8** | **9** | **11** | **10** |

**CIRCUIT DIAGRAM:**

****

**TEST CASES:**

**Test Case 1**

**Input**

HHHH

**Expected Output**

L

**Test Case 2**

**Input**

LLLL

**Expected Output**

H

**Test Case 3**

**Input**

HHLL

**Expected Output**

L

**Test Case 4**

**Input**

LLHH

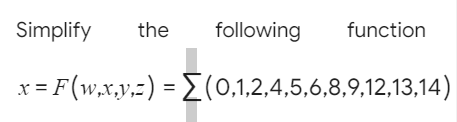
**Expected Output**

L

**RESULT:**

The circuit is successfully verified using all test cases.

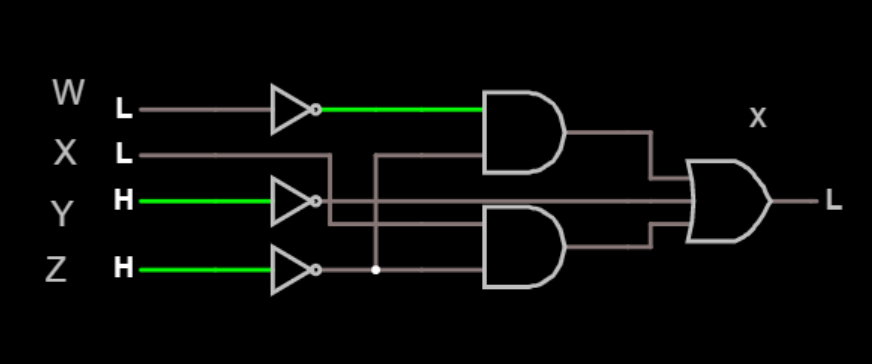
**4. PROBLEM DESCRIPTION:**

****

**SIMPLIFICATION USING K-MAP:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Y’Z’** | **Y’Z** | **YZ** | **YZ’** |
| **W’X’** |  |  |  |  |
|  | **0** | **1** | **3** | **2** |
| **W’X** |  |  |  |  |
|  | **4** | **5** | **7** | **6** |
| **WX** |  |  |  |  |
|  | **12** | **13** | **15** | **14** |
| **WX’** |  |  |  |  |
|  | **8** | **9** | **11** | **10** |

**CIRCUIT DIAGRAM:**

****

**TEST CASES:**

**Test Case 1**

**Input**

HHHH

**Expected Output**

L

**Test Case 2**

**Input**

LLLL

**Expected Output**

H

**Test Case 3**

**Input**

HHLL

**Expected Output**

H

**Test Case 4**

**Input**

LLHH

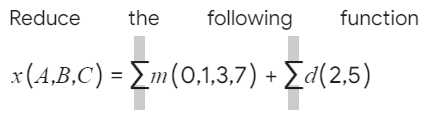
**Expected Output**

L

**RESULT:**

The circuit is successfully verified using all test cases.

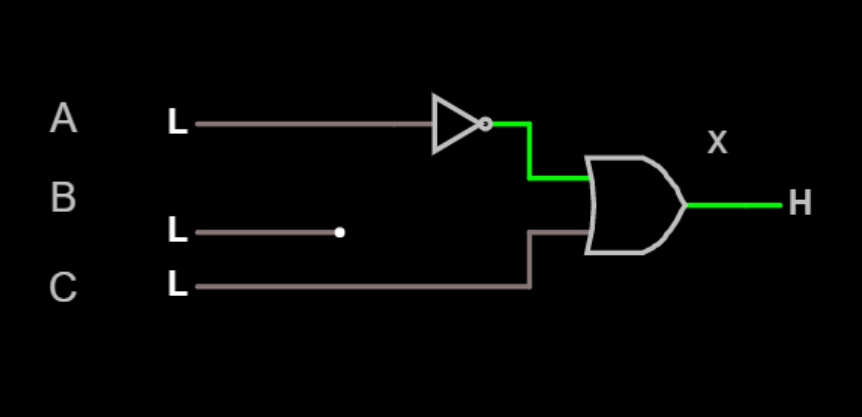
**5. PROBLEM DESCRIPTION:**

****

**SIMPLIFICATION USING K-MAP:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **B’C’** | **B’C** | **BC** | **BC’** |
| **A’** |  |  |  |  |
|  | **0** | **1** | **3** | **2** |
| **A** |  |  |  |  |
|  | **4** | **5** | **7** | **6** |

**CIRCUIT DIAGRAM:**

****

**TEST CASES:**

**Test Case 1**

**Input**

HHH

**Expected Output**

H

**Test Case 2**

**Input**

LLL

**Expected Output**

H

**Test Case 3**

**Input**

HLH

**Expected Output**

H

**Test Case 4**

**Input**

LHL

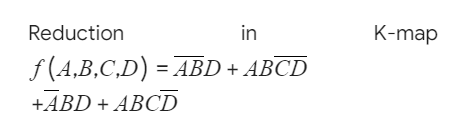
**Expected Output**

H

**RESULT:**

The circuit is successfully verified using all test cases.

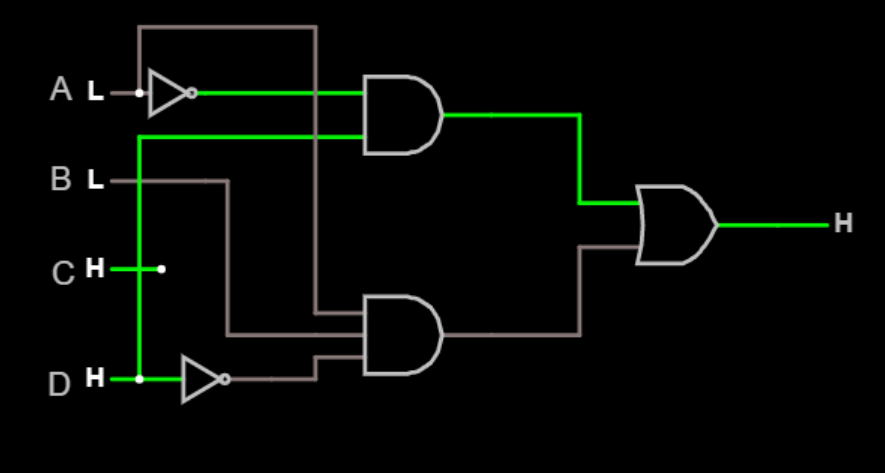
**6. PROBLEM DESCRIPTION:**

****

**SIMPLIFICATION USING K-MAP:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **C’D’** | **C’D** | **CD** | **CD’** |
| **A’B’** |  |  |  |  |
|  | **0** | **1** | **3** | **2** |
| **A’B** |  |  |  |  |
|  | **4** | **5** | **7** | **6** |
| **AB** |  |  |  |  |
|  | **12** | **13** | **15** | **14** |
| **AB’** |  |  |  |  |
|  | **8** | **9** | **11** | **10** |

**CIRCUIT DIAGRAM:**

****

**TEST CASES:**

**Test Case 1**

**Input**

HHHH

**Expected Output**

L

**Test Case 2**

**Input**

LLLL

**Expected Output**

L

**Test Case 3**

**Input**

HHLL

**Expected Output**

H

**Test Case 4**

**Input**

LLHH

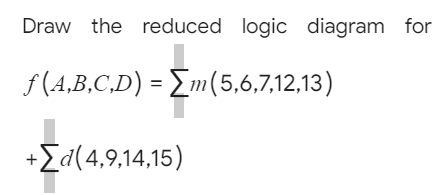
**Expected Output**

H

**RESULT:**

The circuit is successfully verified using all test cases.

**7. PROBLEM DESCRIPTION:**

****

**SIMPLIFICATION USING K-MAP:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **C’D’** | **C’D** | **CD** | **CD’** |
| **A’B’** |  |  |  |  |
|  | **0** | **1** | **3** | **2** |
| **A’B** |  |  |  |  |
|  | **4** | **5** | **7** | **6** |
| **AB** |  |  |  |  |
|  | **12** | **13** | **15** | **14** |
| **AB’** |  |  |  |  |
|  | **8** | **9** | **11** | **10** |

**CIRCUIT DIAGRAM:**

****

**TEST CASES:**

**Test Case 1**

**Input**

L

**Expected Output**

L

**Test Case 2**

**Input**

H

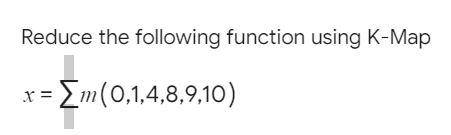
**Expected Output**

H

**RESULT:**

The circuit is successfully verified using all test cases.

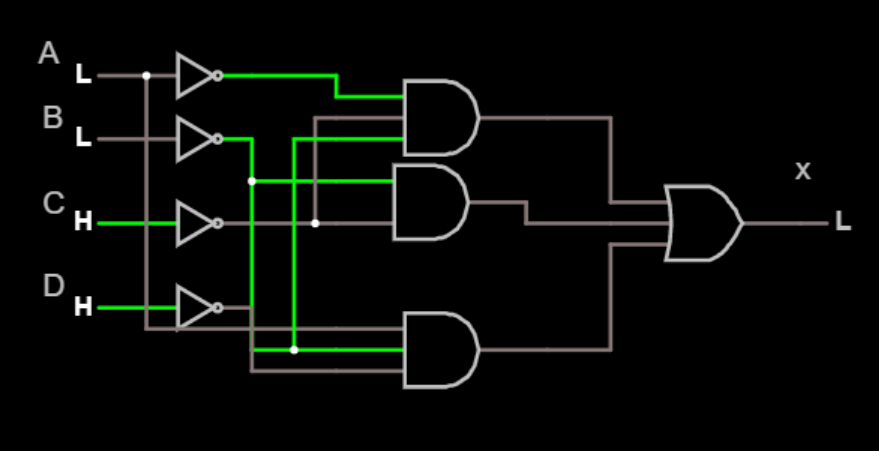
**8. PROBLEM DESCRIPTION:**

****

**SIMPLIFICATION USING K-MAP:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **C’D’** | **C’D** | **CD** | **CD’** |
| **A’B’** |  |  |  |  |
|  | **0** | **1** | **3** | **2** |
| **A’B** |  |  |  |  |
|  | **4** | **5** | **7** | **6** |
| **AB** |  |  |  |  |
|  | **12** | **13** | **15** | **14** |
| **AB’** |  |  |  |  |
|  | **8** | **9** | **11** | **10** |

**CIRCUIT DIAGRAM:**

****

**TEST CASES:**

**Test Case 1**

**Input**

HHHH

**Expected Output**

L

**Test Case 2**

**Input**

LLLL

**Expected Output**

H

**Test Case 3**

**Input**

HHLL

**Expected Output**

L

**Test Case 4**

**Input**

LLHH

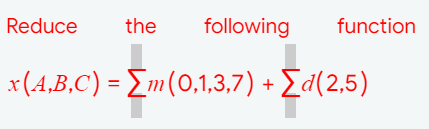
**Expected Output**

L

**RESULT:**

The circuit is successfully verified using all test cases.

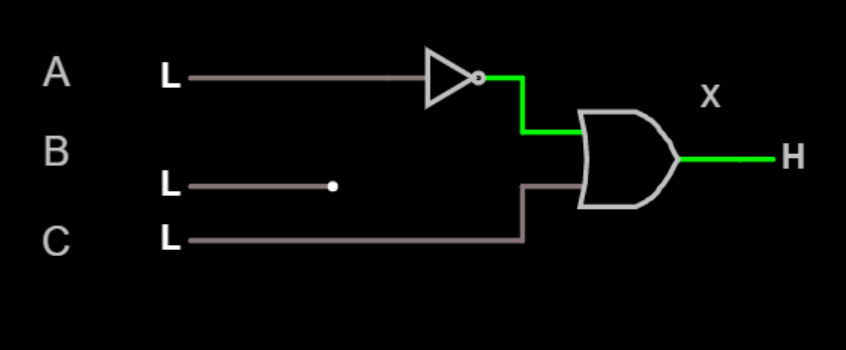
**9. PROBLEM DESCRIPTION:**



**SIMPLIFICATION USING K-MAP:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **B’C’** | **B’C** | **BC** | **BC’** |
| **A’** |  |  |  |  |
|  | **0** | **1** | **3** | **2** |
| **A** |  |  |  |  |
|  | **4** | **5** | **7** | **6** |

**CIRCUIT DIAGRAM:**

****

**TEST CASES:**

**Test Case 1**

**Input**

HHH

**Expected Output**

H

**Test Case 2**

**Input**

LLL

**Expected Output**

H

**Test Case 3**

**Input**

HLH

**Expected Output**

H

**Test Case 4**

**Input**

LHL

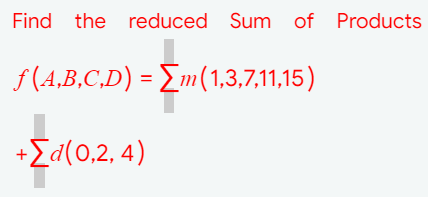
**Expected Output**

H

**RESULT:**

The circuit is successfully verified using all test cases.

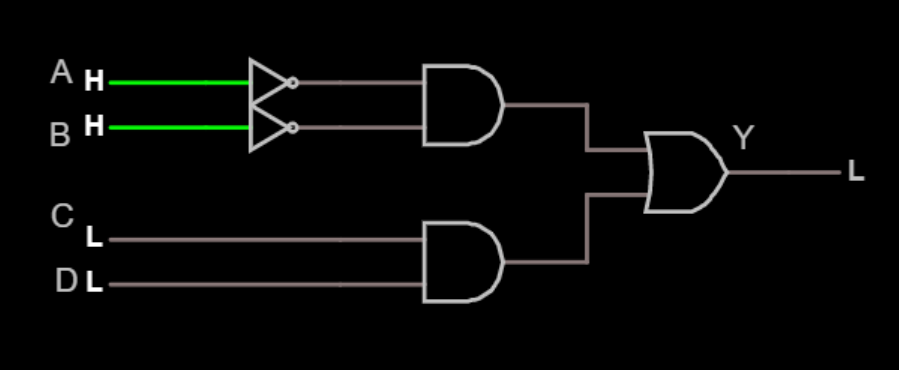
**10. PROBLEM DESCRIPTION:**

****

**SIMPLIFICATION USING K-MAP:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **C’D’** | **C’D** | **CD** | **CD’** |
| **A’B’** |  |  |  |  |
|  | **0** | **1** | **3** | **2** |
| **A’B** |  |  |  |  |
|  | **4** | **5** | **7** | **6** |
| **AB** |  |  |  |  |
|  | **12** | **13** | **15** | **14** |
| **AB’** |  |  |  |  |
|  | **8** | **9** | **11** | **10** |

**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

HHHH

**Expected Output**

H

**Test Case 2**

**Input**

LLLL

**Expected Output**

H

**Test Case 3**

**Input**

HHLL

**Expected Output**

L

**Test Case 4**

**Input**

LLHH

**Expected Output**

H

**RESULT:**

The circuit is successfully verified using all test cases.

**5. FLIP FLOPS**

**1. PROBLEM DESCRIPTION:**

Construct Set-Reset flip flop without using clock.  
  
HINT:

Inputs -S, R  
Outputs - Q, Q'

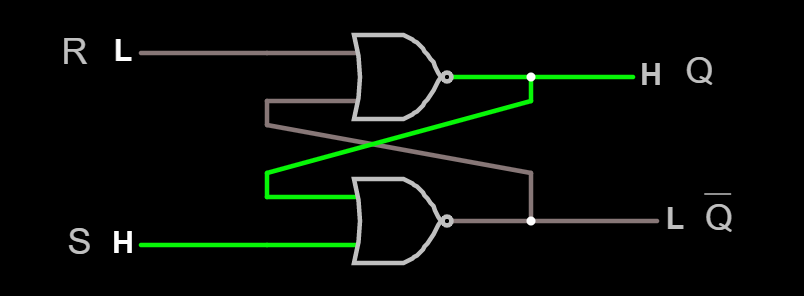
**SR FLIP FLOP WITHOUT CLOCK:**

The SR Flip flop without clock or SR latch is a circuit with two cross-coupled NOR gates or two cross-coupled NAND gates, and two inputs labeled S for set and R for reset.

**TRUTH TABLE:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Inputs** | **Outputs** | | **Remarks** |
| **S** | **R** | **Q(t+1)** | **Q’(t+1)** |
| 0 | 0 | Q(t) | Q’(t) | No Change |
| 0 | 1 | 0 | 1 | Reset |
| 1 | 0 | 1 | 0 | Set |
| 1 | 1 | 0 | 0 | Invalid |

**CIRCUIT DIAGRAM:**

****

**TEST CASES:**

**Test Case 1**

**Input**

1 0

**Expected Output**

0 1

**Test Case 2**

**Input**

1 1

**Expected Output**

0 0

**Test Case 3**

**Input**

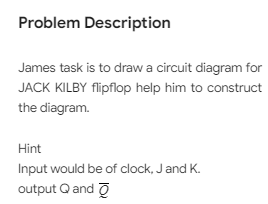
0 1

**Expected Output**

1 0

**RESULT:**

The circuit is successfully verified using all test cases.

**2. PROBLEM DESCRIPTION:**

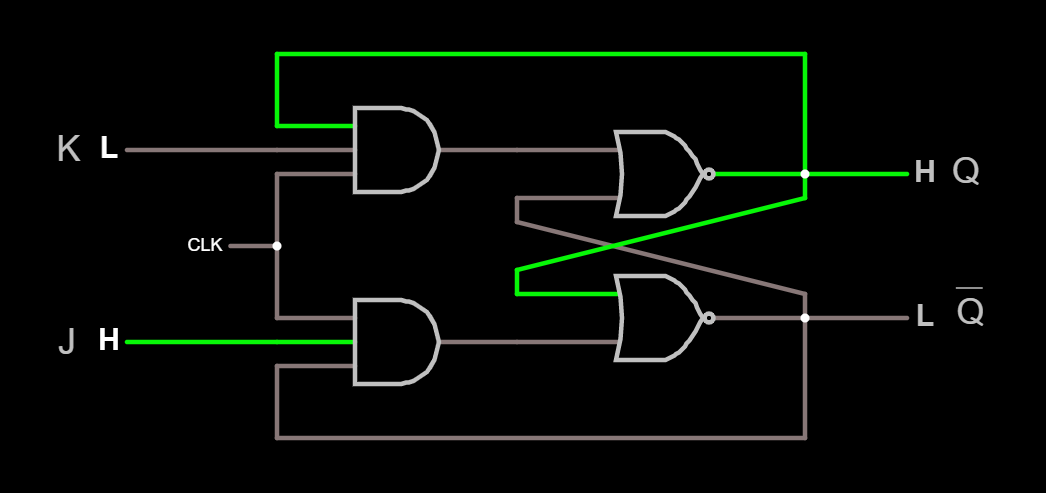
**JK FLIPFLOP:**

JK Flip-flop is a refinement of RS Flip-flop. JK Flip-flop defines the indeterminate state of RS Flip-flop. The inputs are labeled J and K in honor of the inventor of the device, ‘Jack Kilby’. The behavior of inputs J and K is same as the S and R inputs of the S-R flip flop. The JK Flip-flop includes SR latch, two 3-input AND gates and clock pulse. The outputs of the flip flop are returned back as a feedback to the inputs of the AND along with other inputs.

**TRUTH TABLE:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Inputs** | | | **Output** | **Remarks** |
| **CLK** | **J** | **K** | **Q(t+1)** |
| 0 | X | X | Q(t) | No Change |
| 1 | 0 | 0 | Q(t) | No Change |
| 1 | 0 | 1 | 0 | Reset |
| 1 | 1 | 0 | 1 | Set |
| 1 | 1 | 1 | Q’(t) | Toggle |

**CIRCUIT DIAGRAM:**

****

**TEST CASES:**

**Test Case 1**

**Input**

1 0

**Expected Output**

1

**Test Case 2**

**Input**

0 1

**Expected Output**

0

**Test Case 3**

**Input**

0 0

**Expected Output**

NO CHANGE

**Test Case 4**

**Input**

1 1

**Expected Output**

TOGGLE

**RESULT:**

The circuit is successfully verified using all test cases.

**3. PROBLEM DESCRIPTION:**



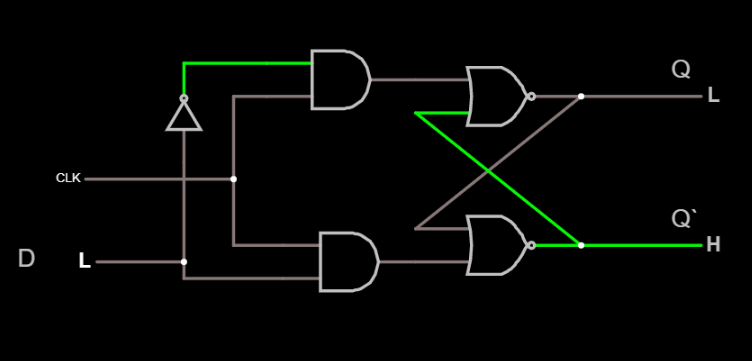
**D FLIPFLOP:**

D Flip-flop is basically an RS flip-flop with an inverter in the R input. D input goes directly to S input and complement of D goes to R input of RS flip-flop. D Flip-flop is known as Delay or Data Flip-flop because of its ability to transfer/delay ‘data’ into flip-flop.

**TRUTH TABLE:**

|  |  |  |
| --- | --- | --- |
| **Inputs** | | **Output** |
| **CLK** | **D** | **Q(t+1)** |
| 0 | X | Q(t) |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

0

**Expected Output**

0

**Test Case 2**

**Input**

1

**Expected Output**

1

**RESULT:**

The circuit is successfully verified using all test cases.

**4. PROBLEM DESCRIPTION:**

Design a Toggle flipflop using gates

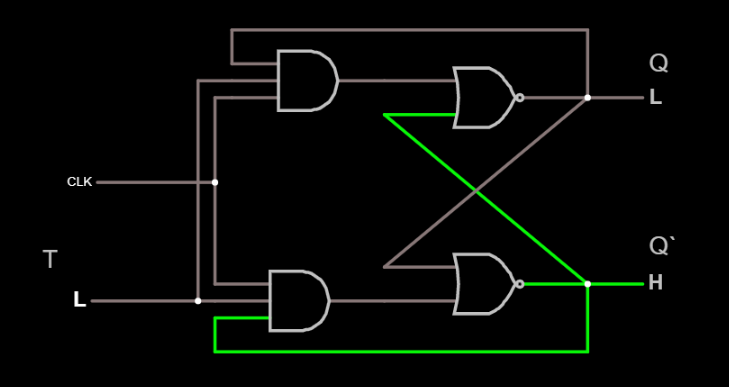
**TOGGLE FLIPFLOP:**

T flip-flop is the simplified version of JK flip-flop. It is obtained by connecting the same input ‘T’ to both inputs of JK flip-flop. It is known as Toggle Flip-flop because of its ability to toggle state.

**TRUTH TABLE:**

|  |  |  |
| --- | --- | --- |
| **Inputs** | | **Output** |
| **CLK** | **T** | **Q(t+1)** |
| 0 | X | Q(t) |
| 1 | 0 | Q(t) |
| 1 | 1 | Q’(t) |

**CIRCUIT DIAGRAM:**



**TEST CASES:**

**Test Case 1**

**Input**

0

**Expected Output**

NO CHANGE

**Test Case 2**

**Input**

1

**Expected Output**

TOGGLE

**RESULT:**

The circuit is successfully verified using all test cases.

**5. PROBLEM DESCRIPTION:**

Construct a Clocked SET-RESET Flipflop.

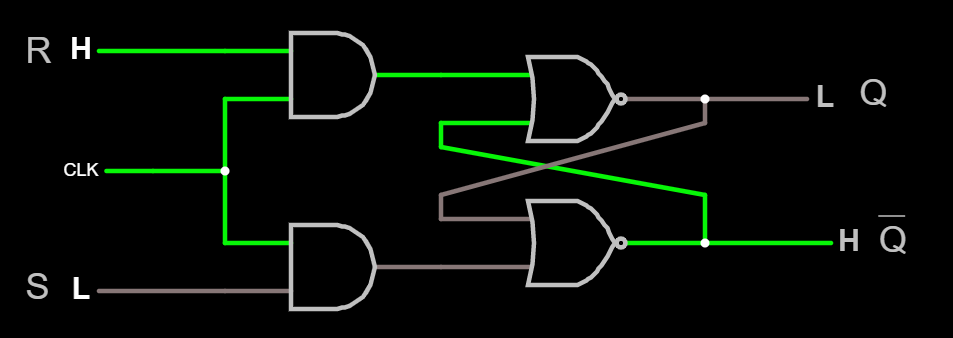
**CLOCKED SET-RESET FLIPFLOP/ SR FLIPFLOP**

The SR stands for SET/RESET. It has two inputs, “SET” which will set the device (Q = 1&Q’ = 0) and “RESET” which will reset the device (Q = 0&Q’ = 1)

**TRUTH TABLE:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Inputs** | | | **Output** | | **Remarks** |
| **CLK** | **S** | **R** | **Q(t+1)** | **Q’(t+1)** |
| 0 | X | X | Q(t) | Q’(t) | No Change |
| 1 | 0 | 0 | Q(t) | Q’(t) | No Change |
| 1 | 0 | 1 | 0 | 1 | Reset |
| 1 | 1 | 0 | 1 | 0 | Set |
| 1 | 1 | 1 | 0 | 0 | Forbidden |

**CIRCUIT DIAGRAM:**

****

**TEST CASES:**

**Test Case 1**

**Input**

0 1

**Expected Output**

0 1

**Test Case 2**

**Input**

1 1

**Expected Output**

0 0

**Test Case 3**

**Input**

1 0

**Expected Output**

1 0

**RESULT:**

The circuit is successfully verified using all test cases.

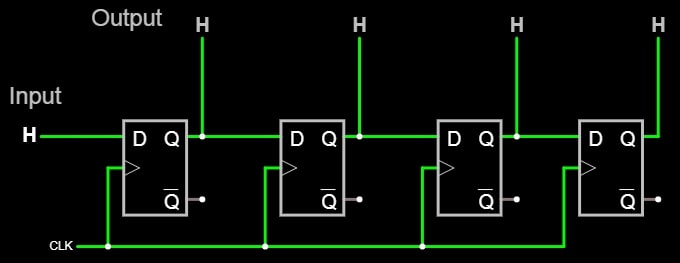
**6. PROBLEM DESCRIPTION:**

Design a Serial-In-Parallel-Out using D-Flipflop.

**SERIAL-IN-PARALLEL-OUT REGISTER**

The shift register, which allows serial input (one bit after the other through a single data line) and produces a parallel output is known as Serial-In Parallel-Out shift register.

**CIRCUIT DIAGRAM:**



**RESULT:**

The circuit is successfully verified.