Digital Logic Practical Lab Report

Realization of Boolean Functions Using NAND Gate

Student:

Abhiroop Mukherjee

Teacher:

Prof. Surajeet Ghosh

Prof. Sekhar Mandal

Course:

Digital Logic Laboratory (CS351)

Experiment 1(a)

Objective

Realization Of AND, OR, NOR, XOR, XNOR Gate using NAND Gate

Theory

1. The AND gate is a basic digital logic gate that implements logical conjunction
    - Its Truth Table is as follows

A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

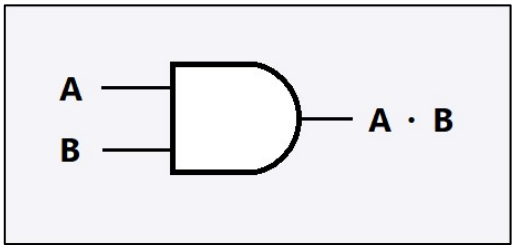

Figure 1: Symbolic Representation of AND Gate

2. The OR gate is a digital logic gate that implements logical disjunction
    - Its Truth Table is as follows

A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

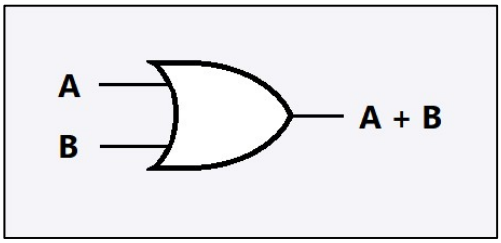

Figure 2: Symbolic Representation of OR Gate

3. NOR is the result of the negation of the OR operator.
    - Its Truth Table is as follows

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

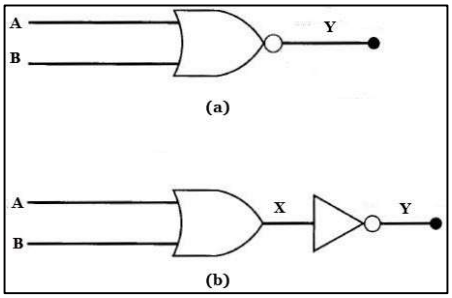

Figure 3: Symbolic Representations of NOR Gate, Both Figure (a) and (b) are Equivalent

4. XOR is a digital logic gate that gives a true (1 or HIGH) output when the number of true inputs is odd.
    - The algebraic expressions $A.B' + A'.B$ or $(A + B).(A' + B')$ or $A \oplus \mathbf{B}$ all represent the XOR gate with inputs A and B.
    - Its Truth Table is as follows

A	B	Y =A $\oplus$ B
0	0	0
0	1	1
1	0	1
1	1	0

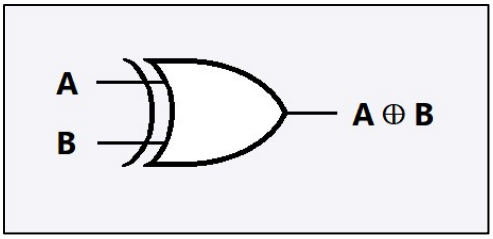

Figure 4: Symbolic Representation of XOR Gate

5. The XNOR gate (sometimes ENOR, EXNOR or NXOR and pronounced as Exclusive NOR) is a digital logic gate whose function is the logical complement of the exclusive OR (XOR) gate.
    - Its Truth Table is as follows

A	B	Y = A XNOR B
0	0	0
0	1	1
1	0	1
1	1	0

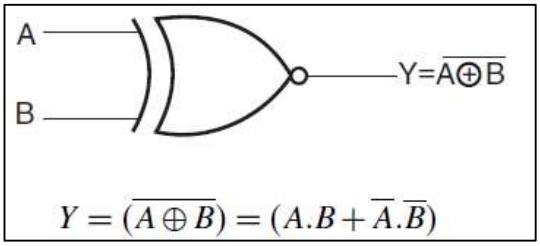

Figure 5: Symbolic Representation of XNOR Gate

6. A NAND gate (NOT-AND) is a logic gate which produces an output which is false only if all its inputs are true. thus, its output is complementing to that of an AND gate.
    - Its Truth Table is as Follows

A	B	Y = $(\mathbf{A}.\mathbf{B})'$
0	0	0
0	1	1
1	0	1
1	1	0

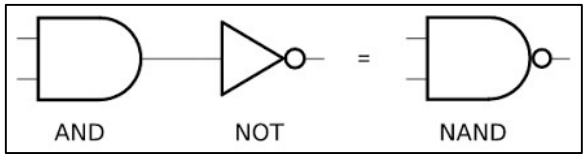

Figure 6: Symbolic Representations of NAND Gate

Circuit Diagrams

The Following Circuits were made for the following Gates (using NAND Gate)

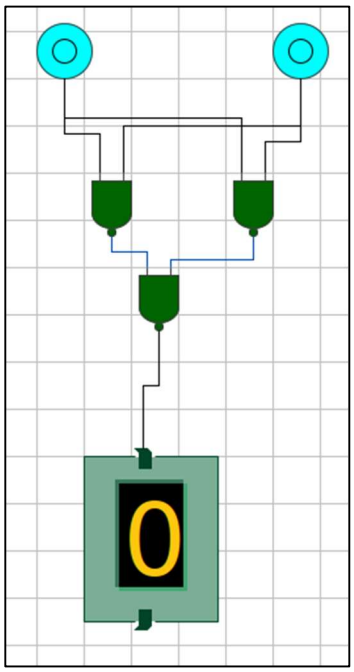

Figure 7: AND Gate using NAND Gates

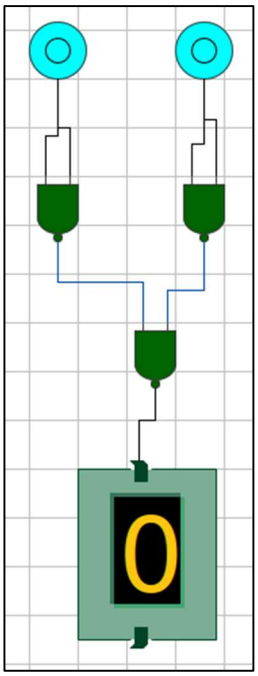

Figure 8: OR Gate using NAND Gates

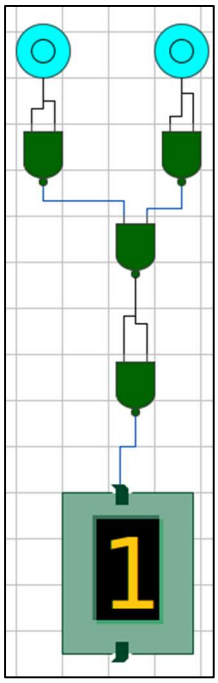

Figure 9: NOR Gate using NAND Gates

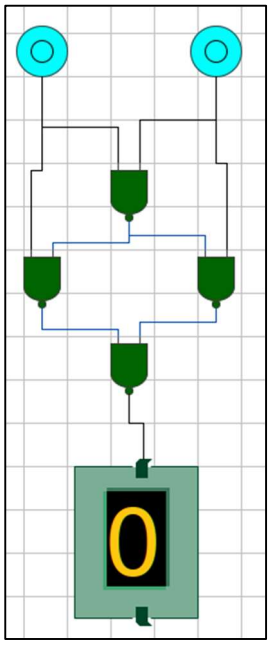

Figure 10: XOR Gate using NAND Gates

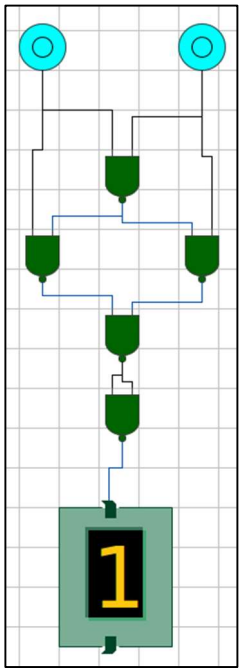

Figure 11: XNOR Gate using NAND Gates

Input/Output

1. AND Gate from NAND Gates

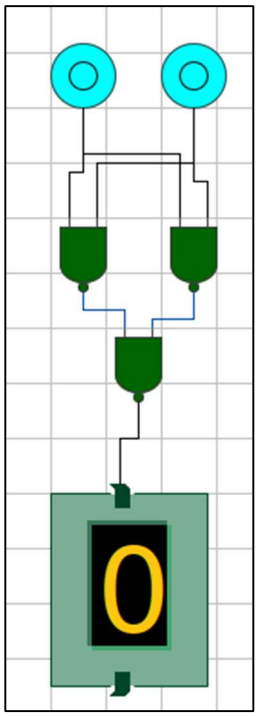

Figure 12: Output for 0 0

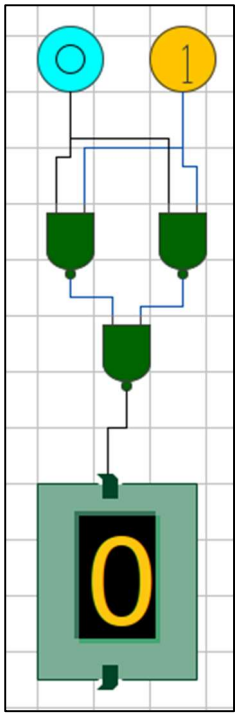

Figure 13: Output for 0 1

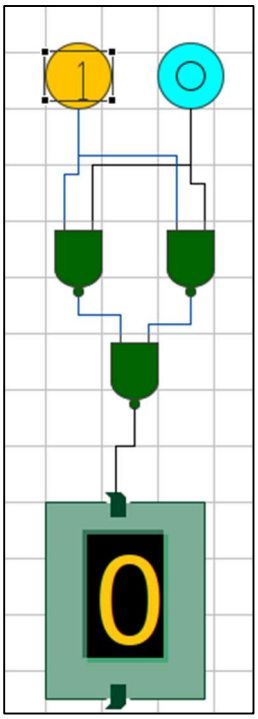

Figure 14: Output for 1 0

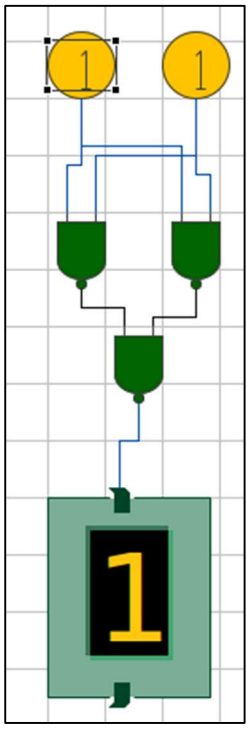

Figure 15: Output for 1 1

2. OR Gate using NAND Gates

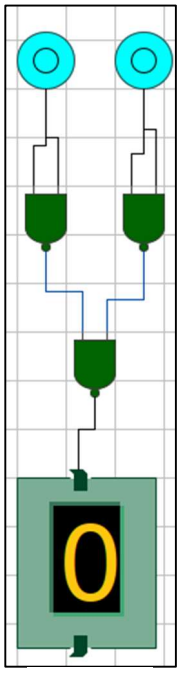

Figure 16: Output for 0 0

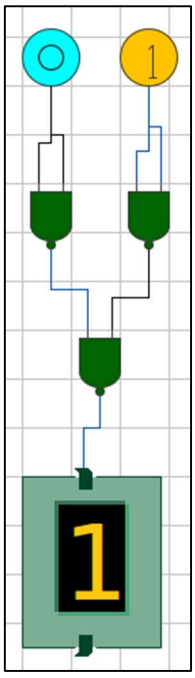

Figure 17: Output for 0 1

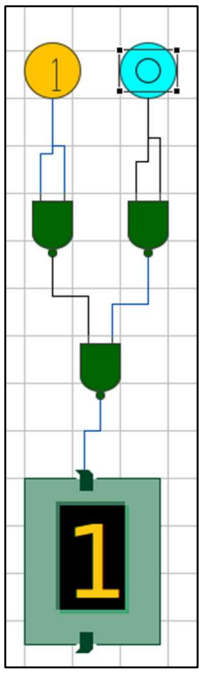

Figure 18: Output for 1 0

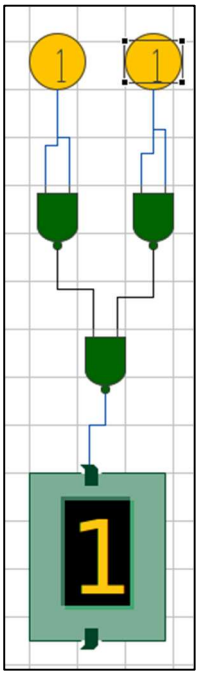

Figure 19: Output for 1 1

3. NOR Gate using NAND Gates

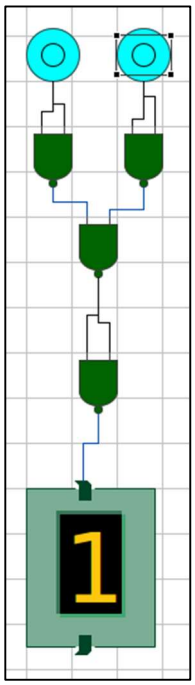

Figure 20: Output for 0 0

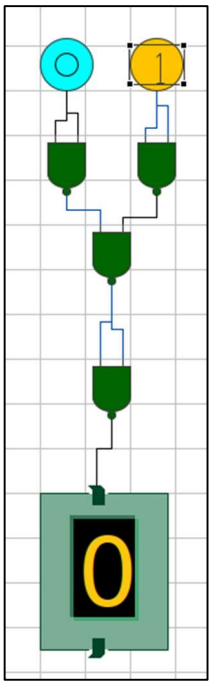

Figure 21: Output for 0 1

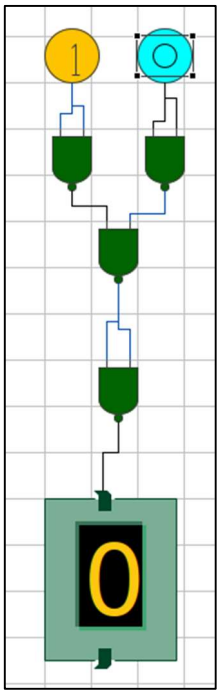

Figure 22: Output for 1 0

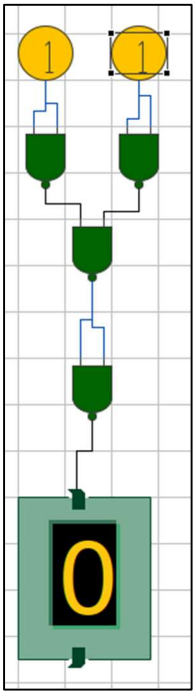

Figure 23: Output for 1 1

4. XOR Gate using NAND Gates

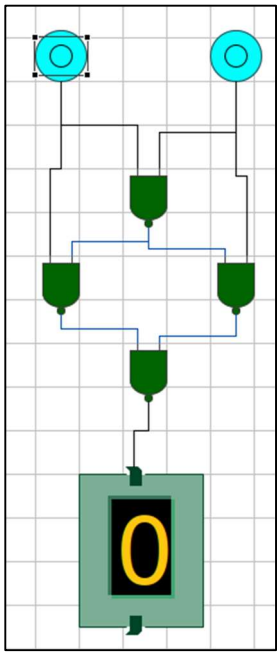

Figure 24: Output for 0 0

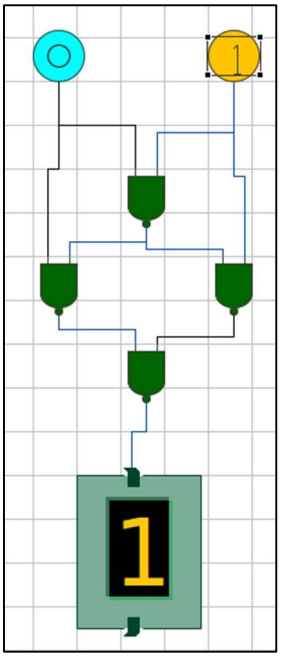

Figure 25: Output for 0 1

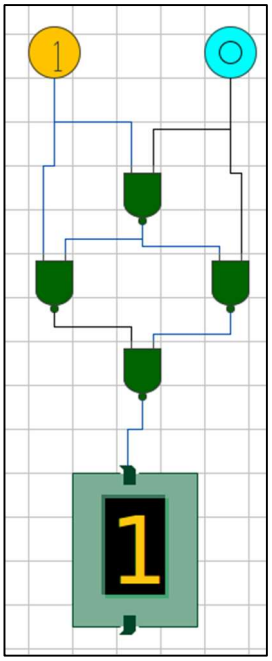

Figure 26: Output for 1 0

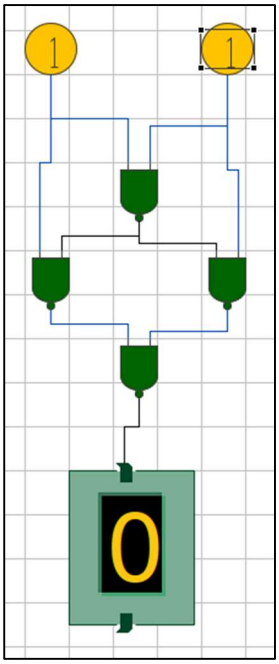

Figure 27: Output for 1 1

5. XNOR Gate using NAND Gates

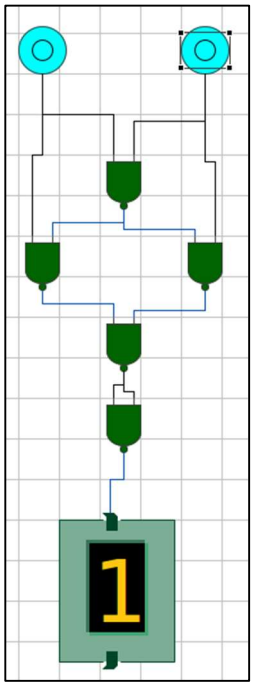

Figure 28: Output for 0 0

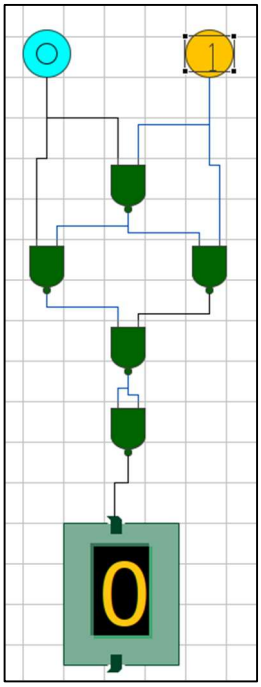

Figure 29: Output for 0 1

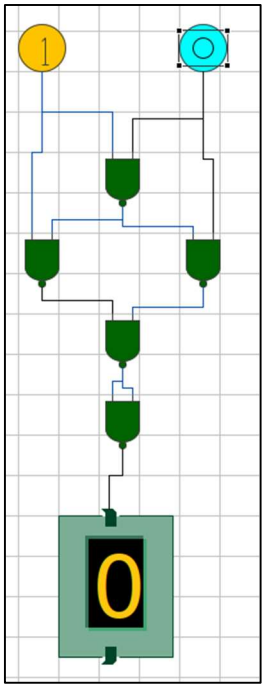

Figure 30: Output for 1 0

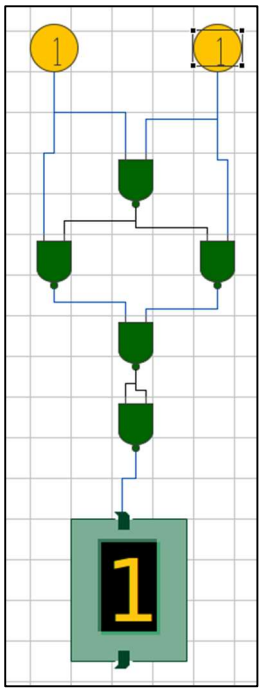

Figure31: Output for 1 1

Experiment 1(b)

Objective

Realization of Half Adder, Full Adder, Half Subtractor, and Full Subtractor using two input NAND Gates

Theory

1. Half adder is a combinational arithmetic circuit that adds two numbers and produces a sum bit (S) and carry bit (C) as the output.

Inputs		Outputs	
x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

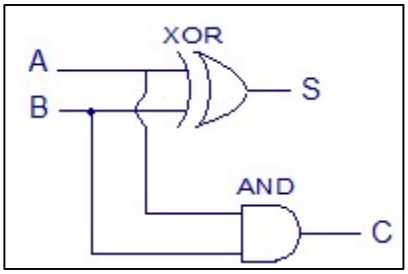

Figure 2: Realization of Half Adder

   - From Truth Table, we observe that
     - $C = x.y$
     - $S = x'.y + x.y' = x \oplus \mathbf{y}$

2. Full Adder is the adder which adds three inputs and produces two outputs. The first two inputs are A and B and the third input is an input carry as C-IN. The output carry is designated as C-OUT and the normal output is designated as S which is SUM.

Inputs			Outputs	
A	B	C - In	Sum	C - 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

- As Full Adder is used in Binary Addition, we can cascade two Half Adders to form a Full Adder

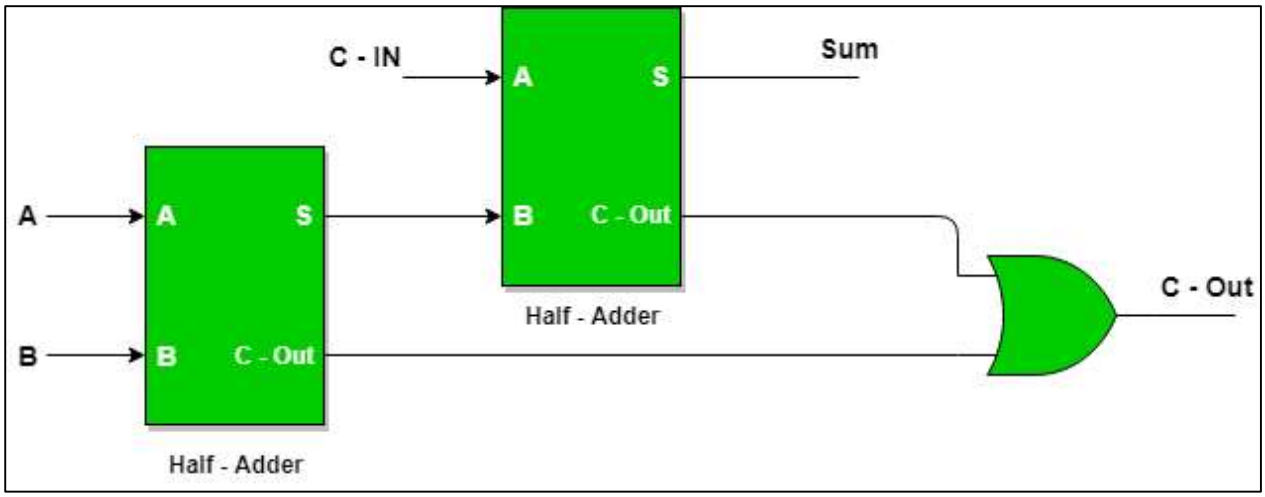

Figure 3: Realization of Full Adder as the cascaded form of Two Half Adders

3. A half subtractor is a logical circuit that performs a subtraction operation on two binary digits. The half subtractor produces a sum and a borrow bit for the next stage.

Input		Output	
x	y	Borrow(B)	Difference(D)
0	0	0	0
0	1	1	1
1	0	0	1
1	1	0	0

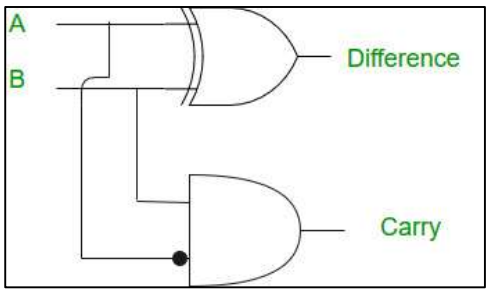

Figure 4: Realization of Half Subtractor

- From Truth Table, we observe that
  - $B = x'.y$
  - $D = x'.y + x.y' = x \oplus \mathbf{y}$

4. A full subtractor is a combinational circuit that performs subtraction of two bits, one is minuend and other is subtrahend, considering borrow of the previous adjacent lower minuend bit.

Inputs			Outputs	
A	B	B - In	Difference (D)	B - Out
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

- As Full Subtractor is used in Binary Subtraction, we can cascade two Half Subtractor to form a Full Adder

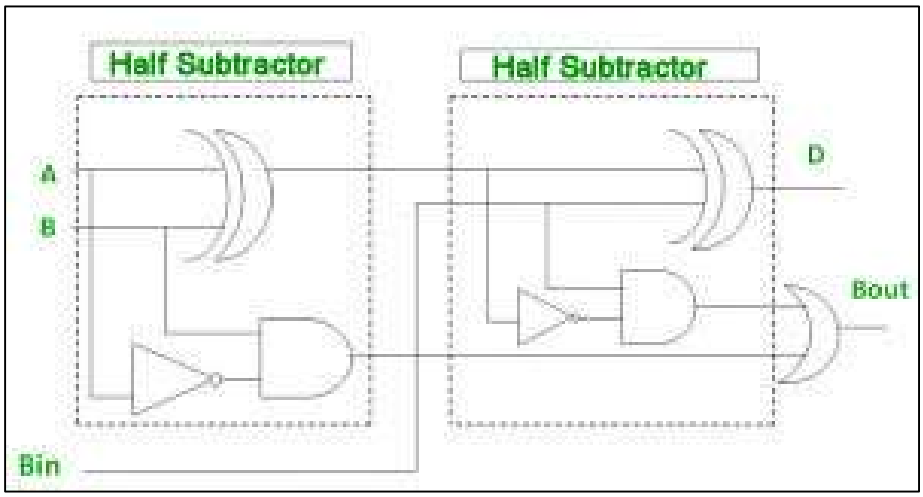

Figure 5: Realization of Full Subtractor as the cascaded form of Two Half Subtractors

Circuits

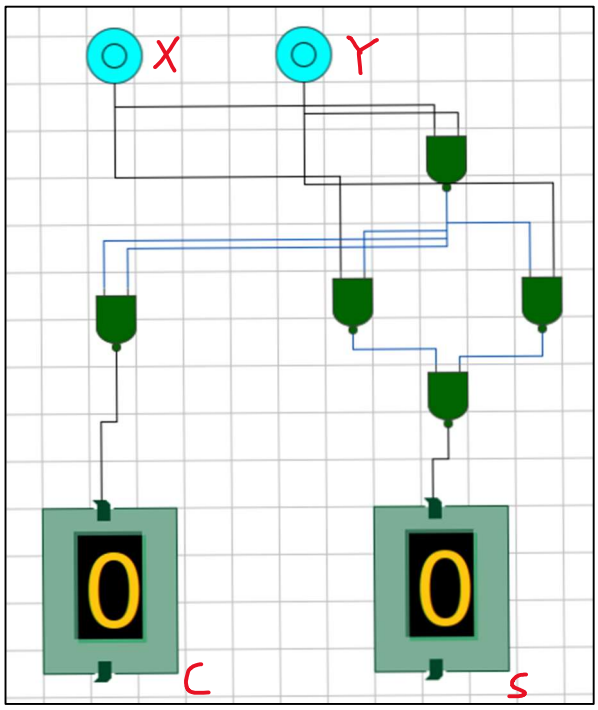

Figure 6: Half Adder Circuit. The Left Display is Carry(C) and the Right Display is Sum(S)

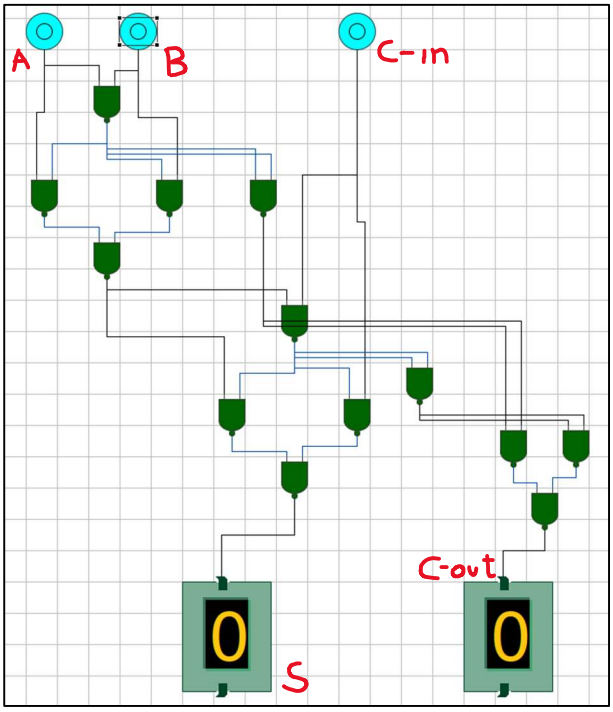

Figure 7: Full Adder Circuit. Here the Left Display is Sum(S) and right Display is Carry(C)

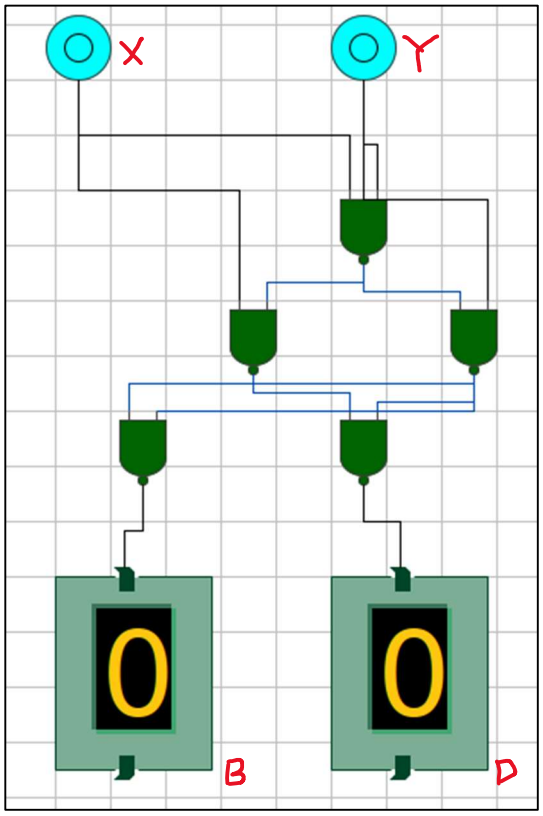

Figure 8: Half Subtractor Circuit. Here the Left Display is Borrow(B) and Right Display is Difference(D)

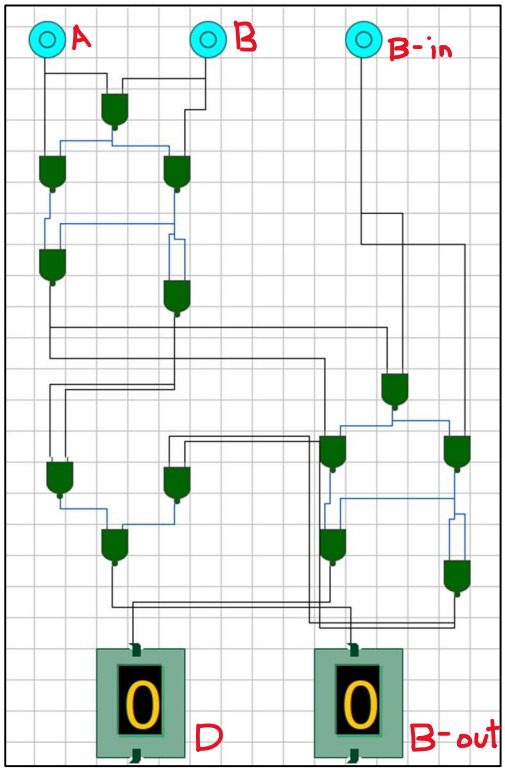

Figure 9: Full Subtractor Circuit. Here Left Display is Difference(D) and Right Display is B-out

Outputs

1. Half Adder

    - This circuit take left input bit as ‘X’ and right input bit as ‘Y’, and outputs the sum ‘X+Y’ in right output display and Carry in left output display

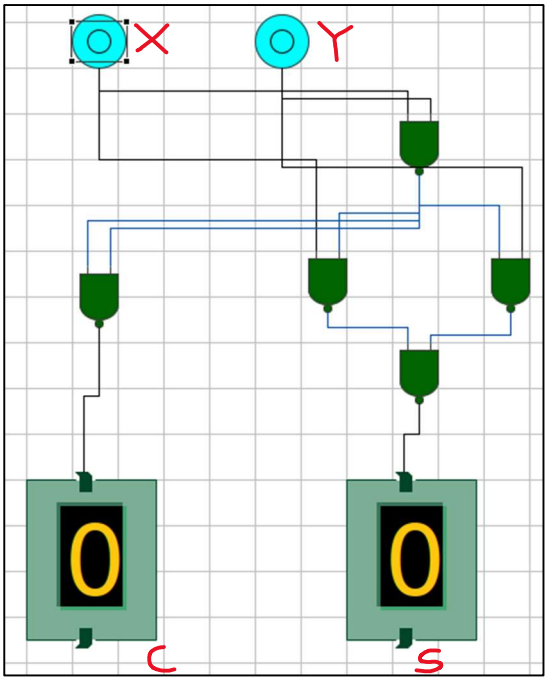

Figure 10: X=0, Y=0; C=0, S=0

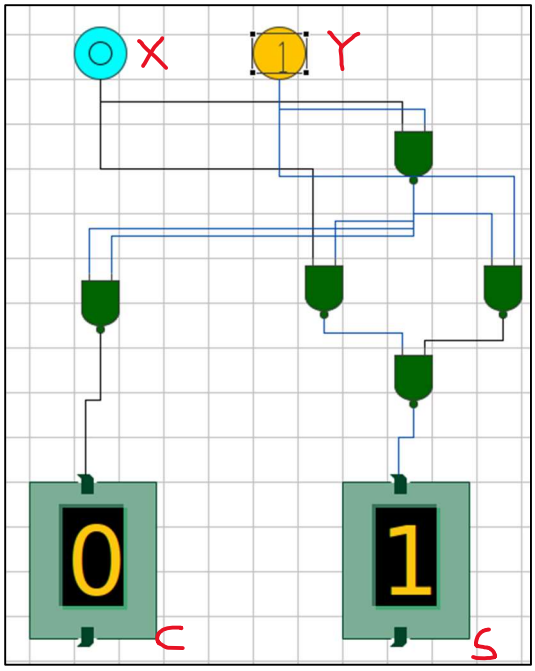

Figure 11:X=0, Y=1; C=0, S=1

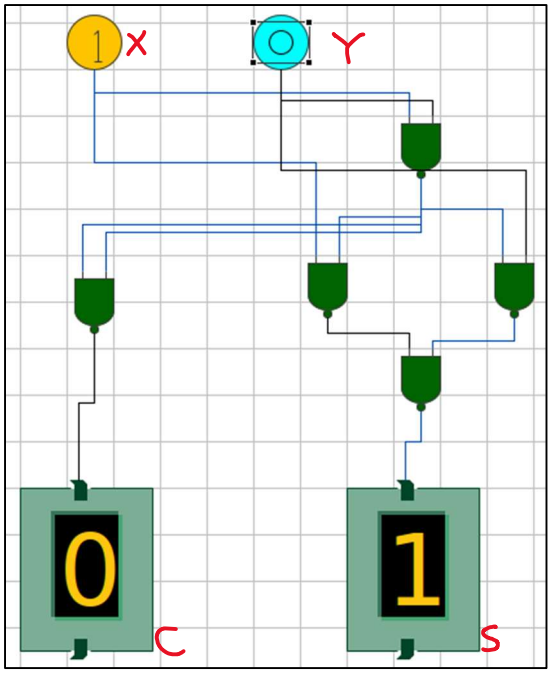

Figure 12:X=1, Y=0; C=0, S=1

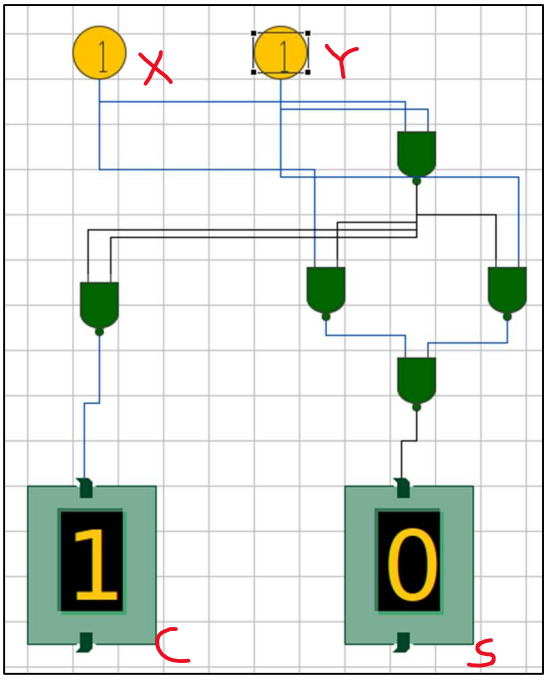

Figure 13:X=1, Y=1; C=1, S=0

2. Full Adder

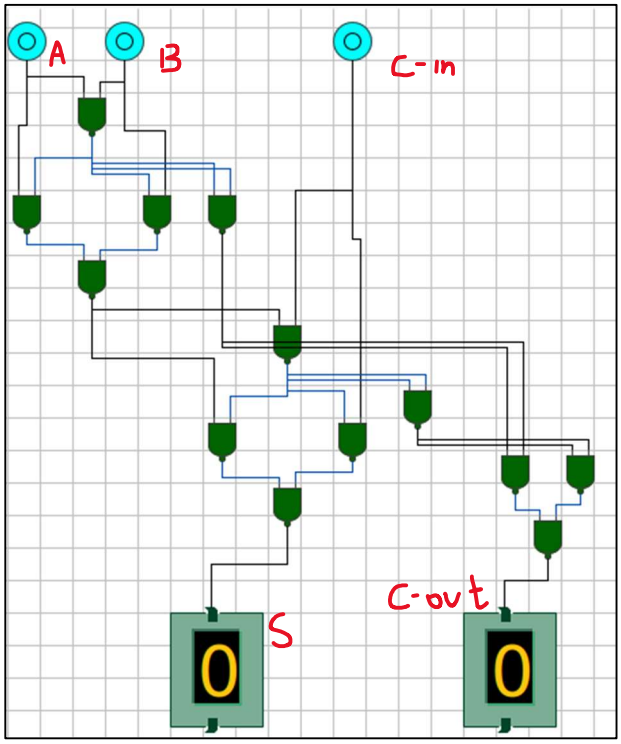

Figure 14: X=0, Y=0, C-in=0; S=0, C-out=0

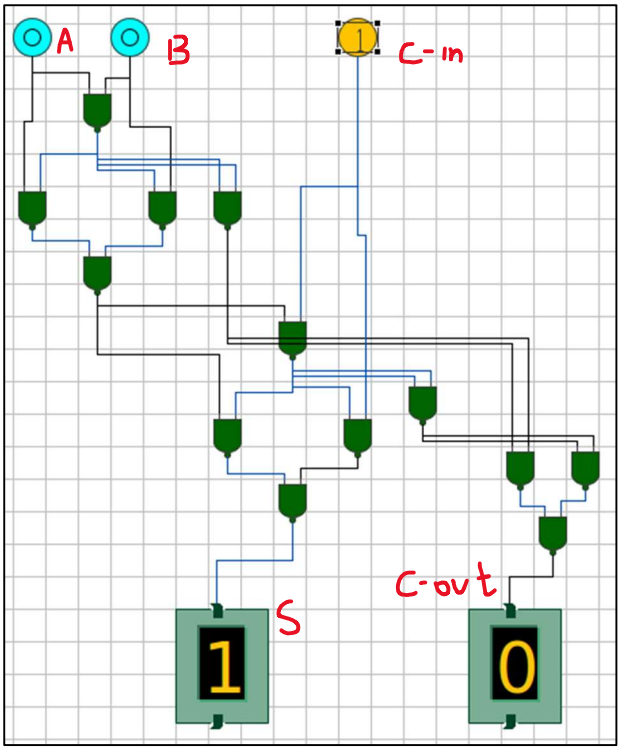

Figure 15: X=0, Y=0, C-in=1; S=1, C-out=0

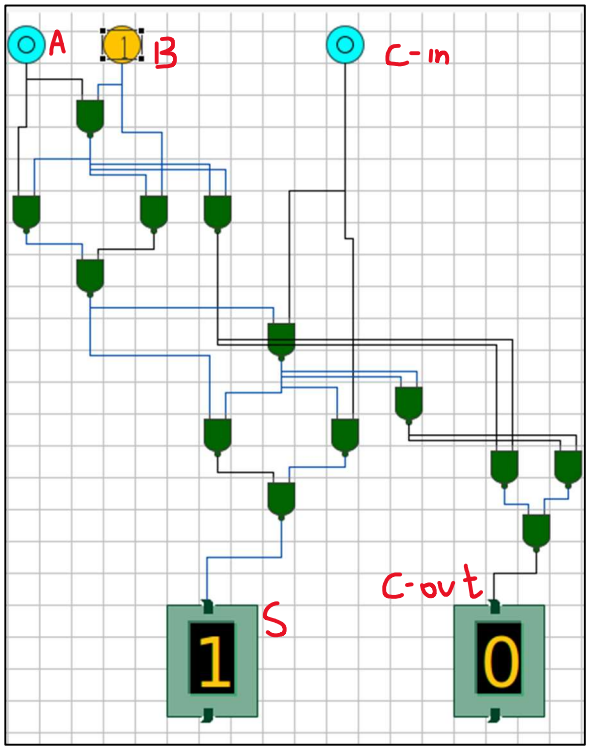

Figure 16: X=0, Y=1, C-in=0; S=1, C-out=0

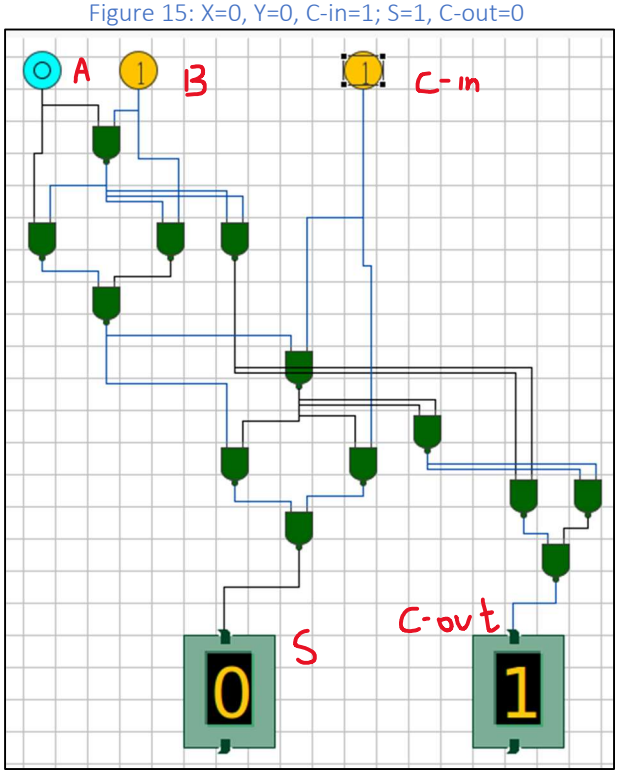

Figure 17: X=0, Y=1, C-in=1; S=0, C-out=1

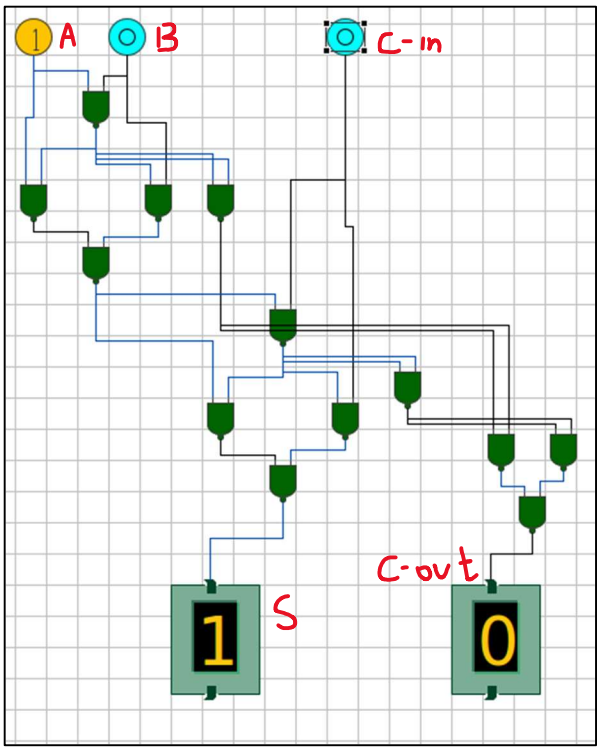

Figure 18: X=1, Y=0, C-in=0; S=1, C-out=0

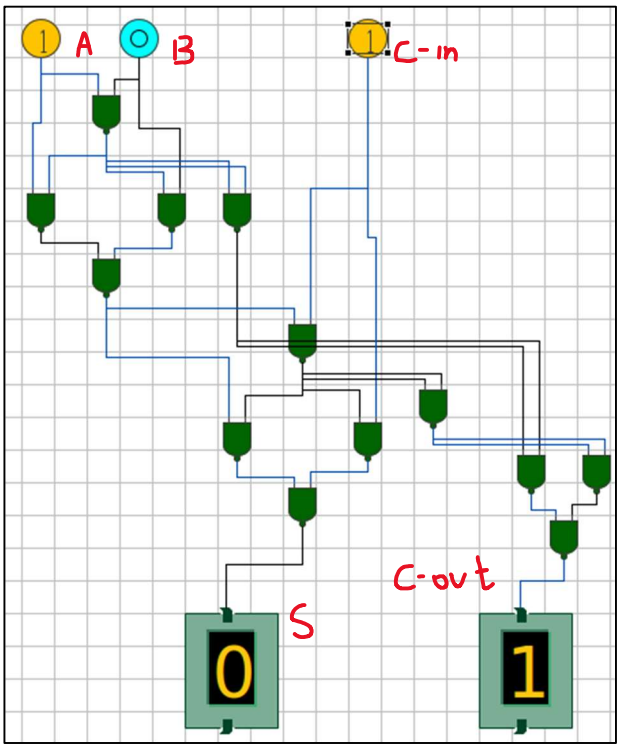

Figure 19: X=1, Y=0, C-in=1; S=0, C-out=1

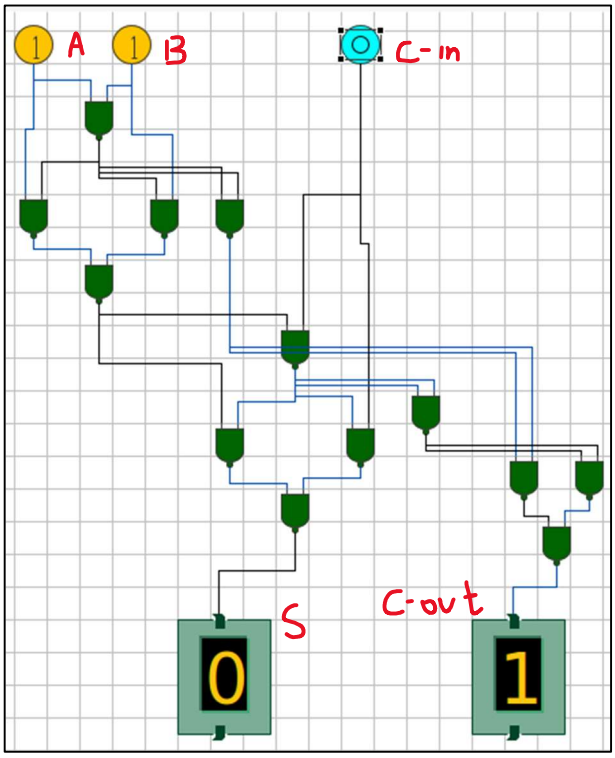

Figure 20: X=1, Y=0, C-in=1; S=0, C-out=1

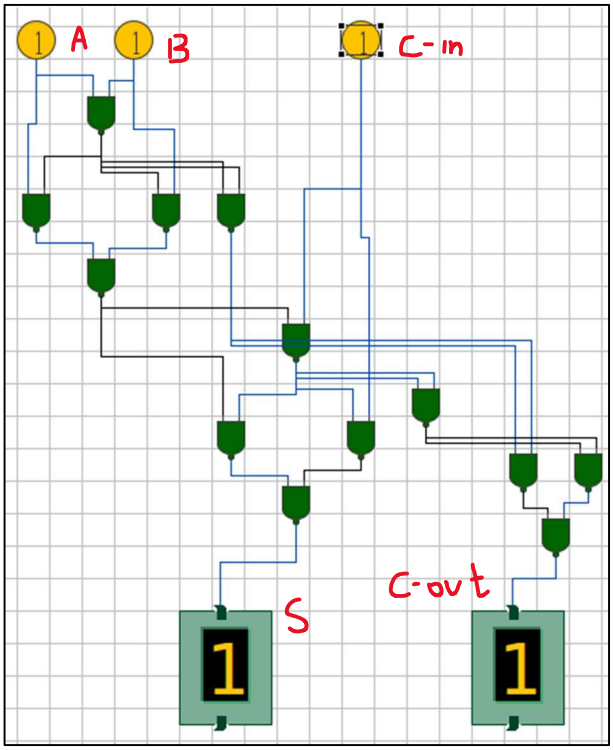

Figure 21: X=1, Y=1, C-in=1; S=1, C-out=1

3. Half Subtractor

- This Circuit give Difference as X-Y

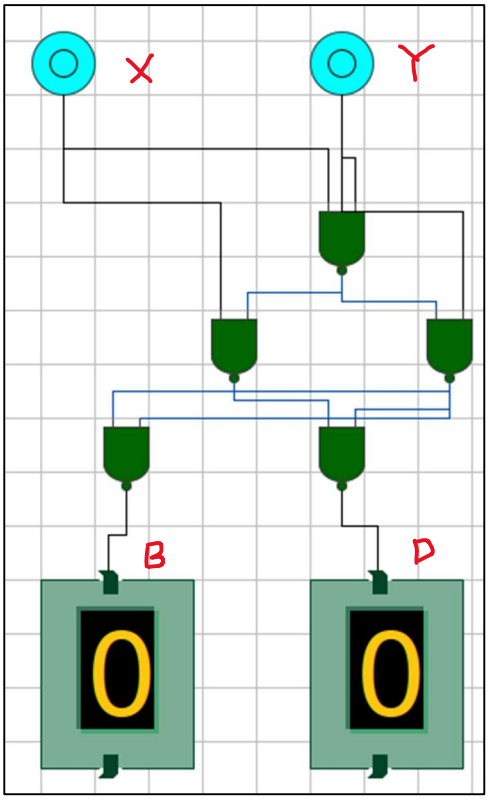

Figure 22: X=0, Y=0; B=0, D=0

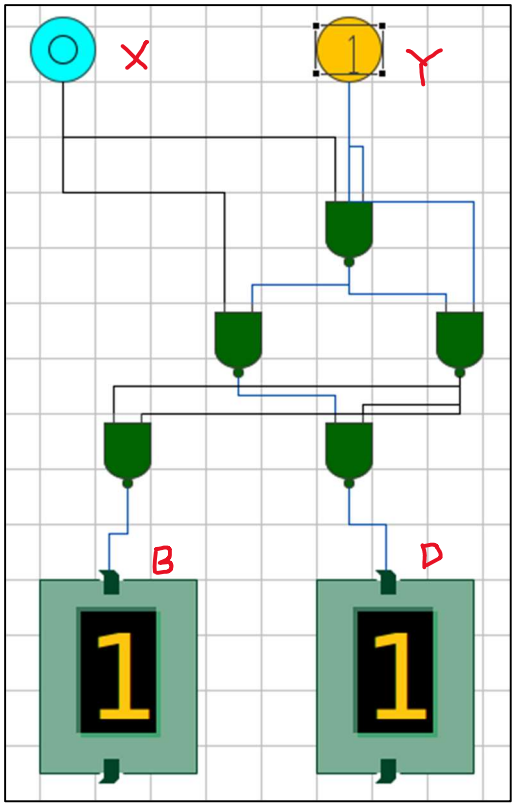

Figure 23: X=0, Y=1; B=1, D=1

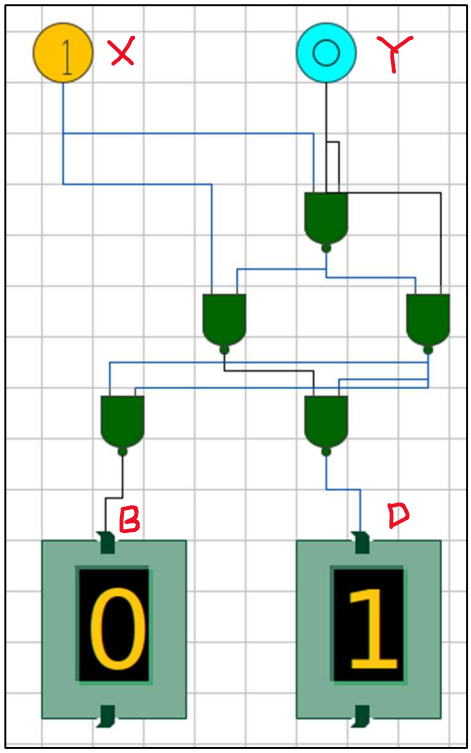

Figure 23: X=1, Y=0; B=0, D=1

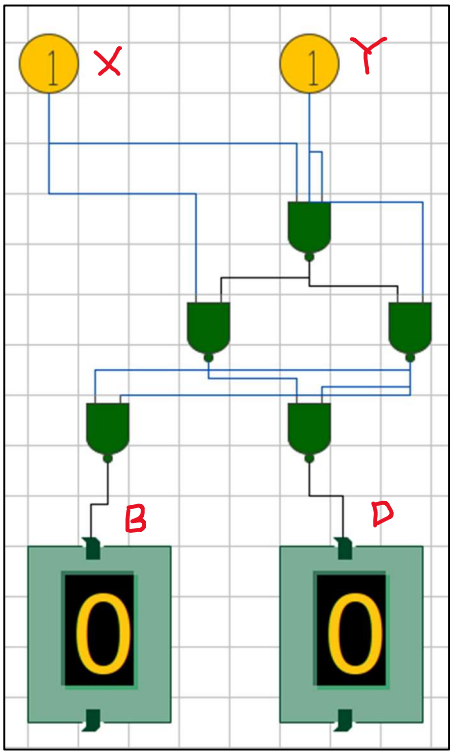

Figure 24: X=1, Y=1; B=0, D=0

4. Full Subtractor

- This Circuit Give Difference as A - B - (B-in)

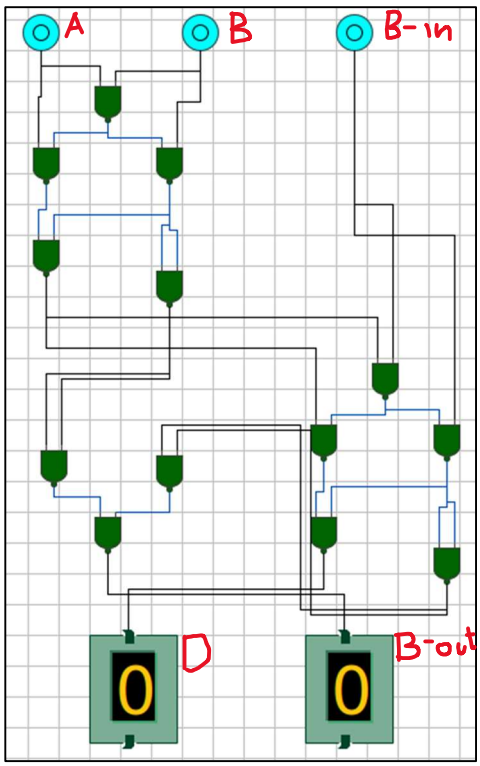

Figure 25: A=0, B=0, B-in=0; D=0, B-out=0

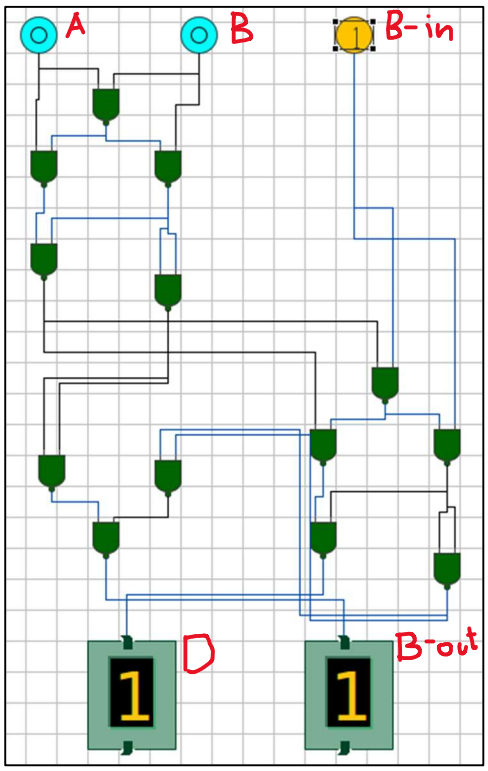

Figure 26: A=0, B=0, B-in=1; D=1, B-out=1

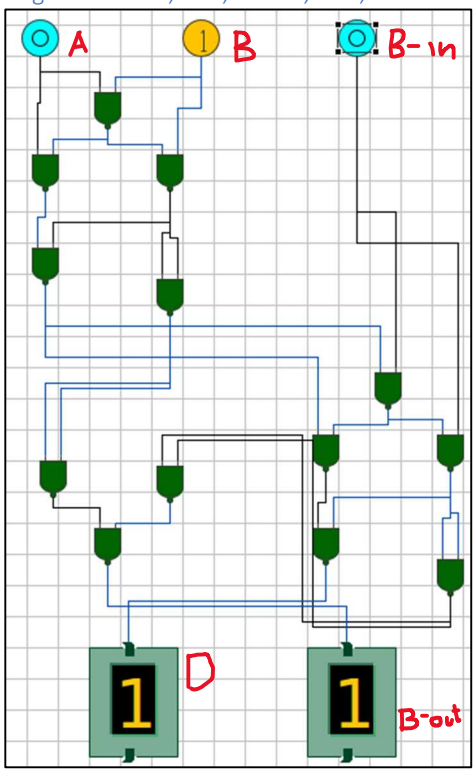

Figure 27: A=0, B=1, B-in=0; D=1, B-out=1

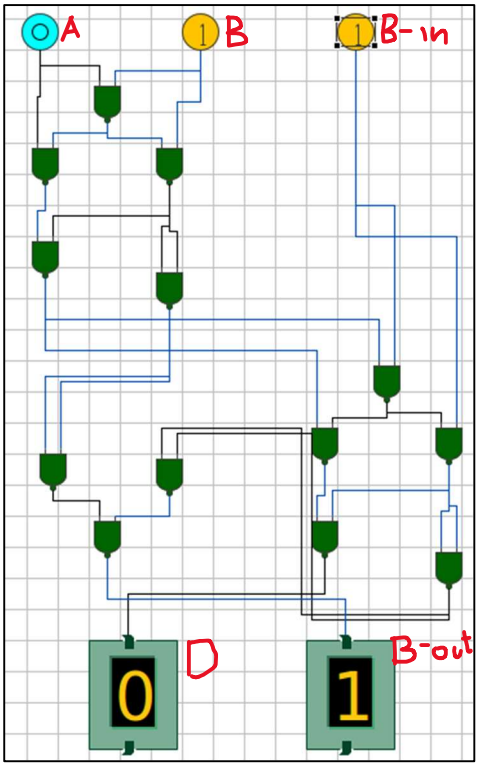

Figure 28: A=0, B=1, B-in=1; D=0, B-out=1

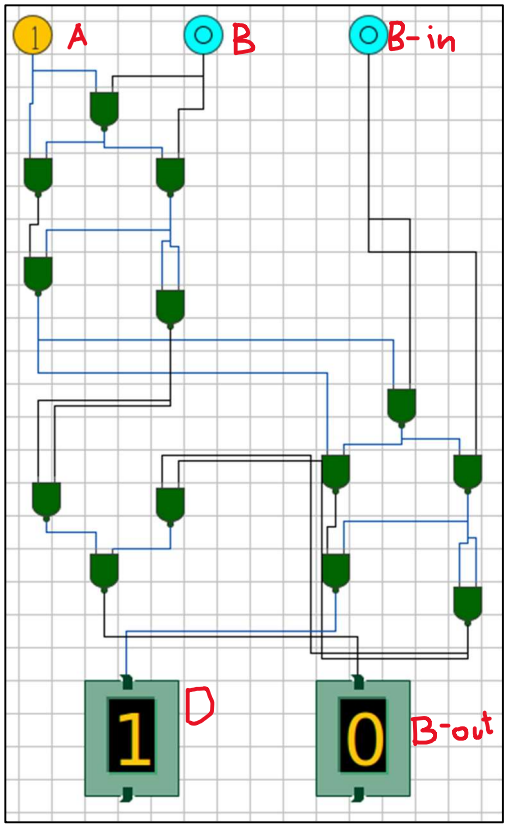

Figure 29: A=1, B=0, B-in=0; D=1, B-out=0

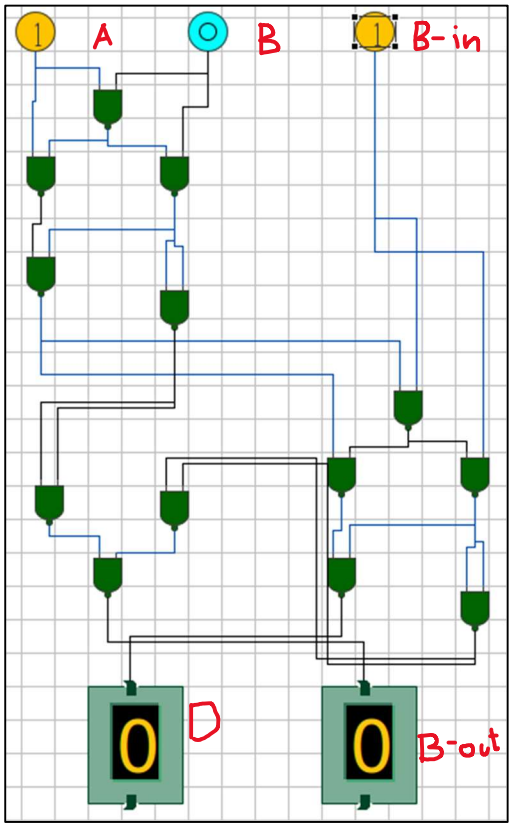

Figure 30: A=1, B=0, B-in=1; D=0, B-out=0

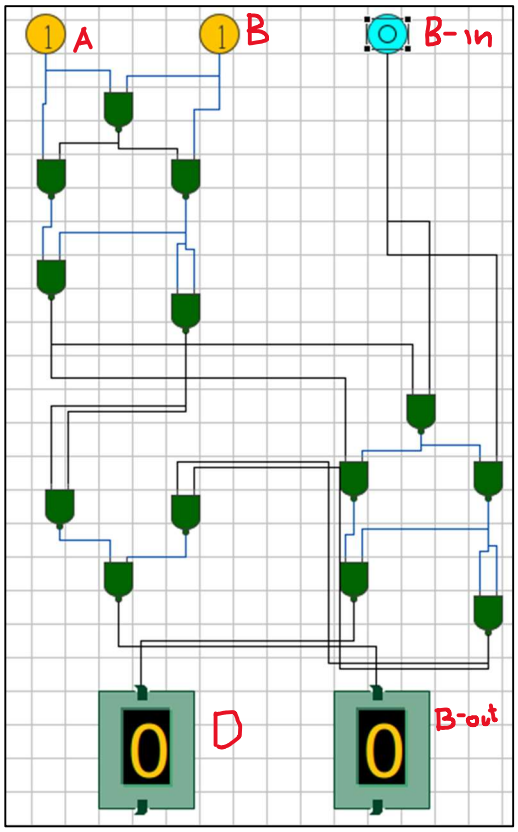

Figure 31: A=1, B=1, B-in=0; D=0, B-out=0

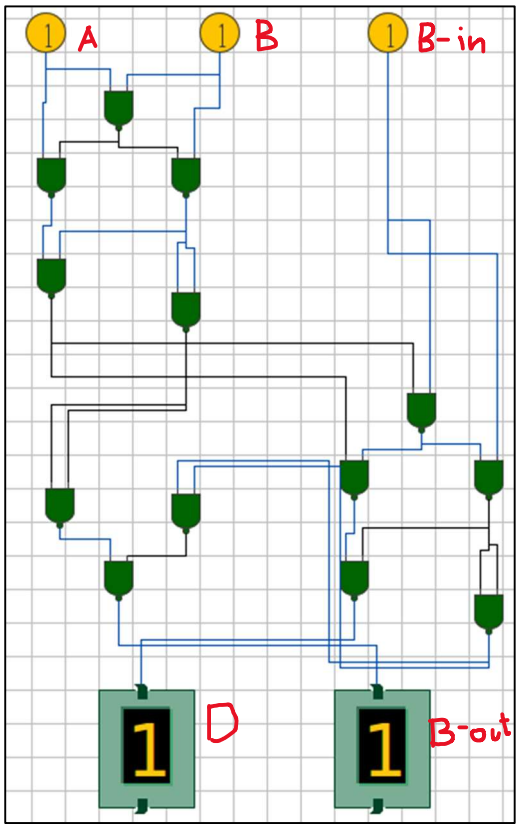

Figure 32: A=1, B=1, B-in=1; D=1, B-out=1

Experiment 1(c)

Objective

Realization of Half Adder, Full Adder, Half Subtractor, and Full Subtractor using two input NAND Gates and XOR gates

Theory

In every circuit made in Experiment 1(b), we can substitute the following arrangement of NAND Gates with XOR Gate, as they are logically equivalent.

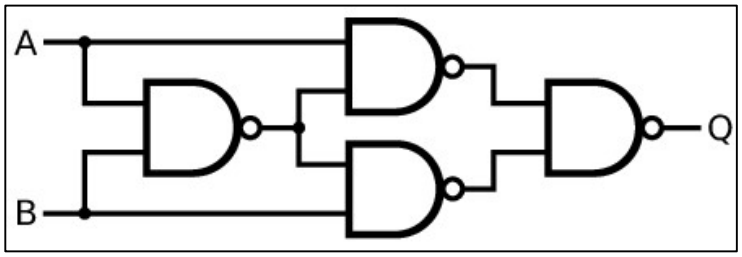

Figure 1: Equivalent Circuit of XOR Gate made by NAND Gates

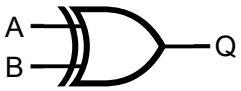

Figure 2: XOR Gate

Circuits

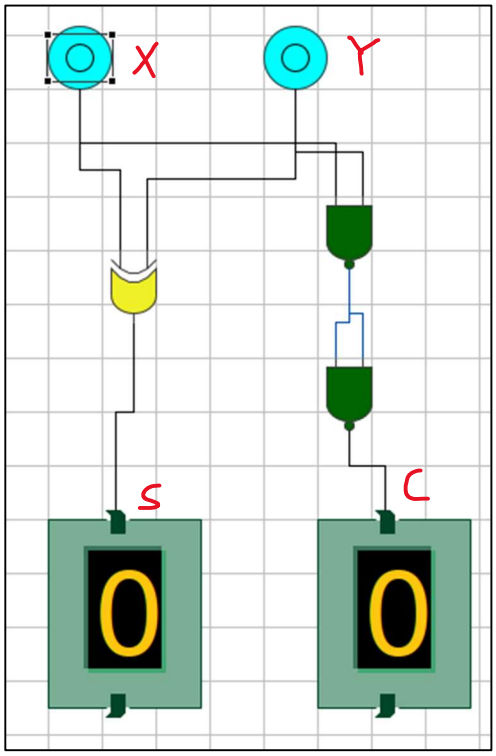

Figure 3: Half Adder Circuit

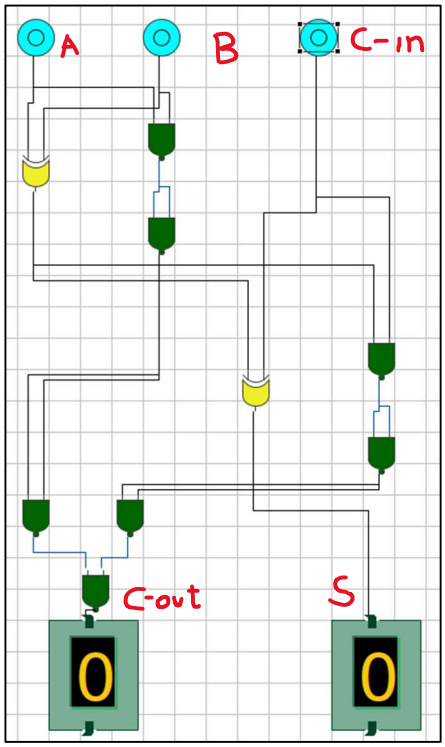

Figure 3: Full Adder Circuit

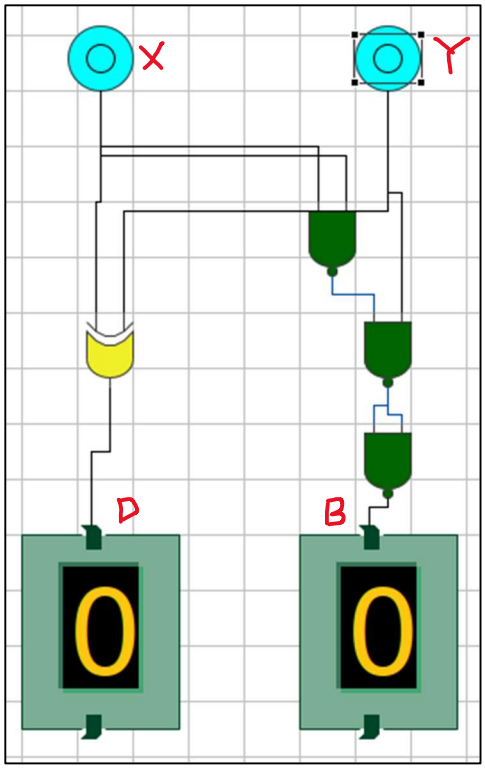

Figure 4: Half Subtractor Circuit

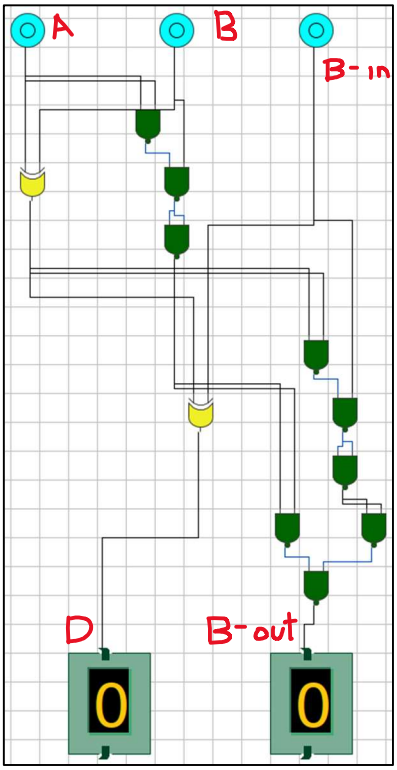

Figure 5: Full Subtractor Circuit

Outputs

1. Half Adder

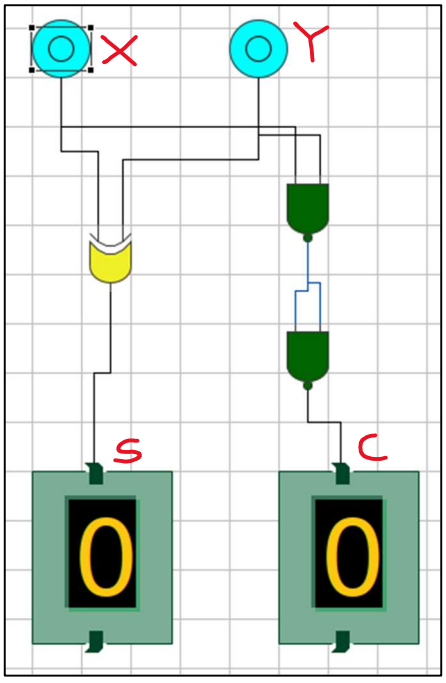

Figure 6: X=0, Y=0; C=0, S=0

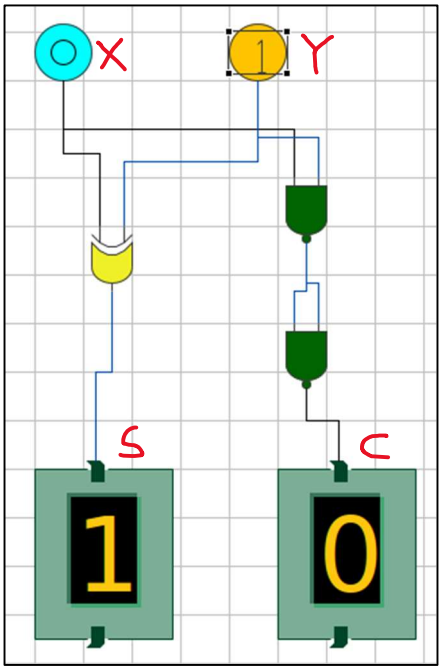

Figure 7:X=0, Y=1; C=0, S=1

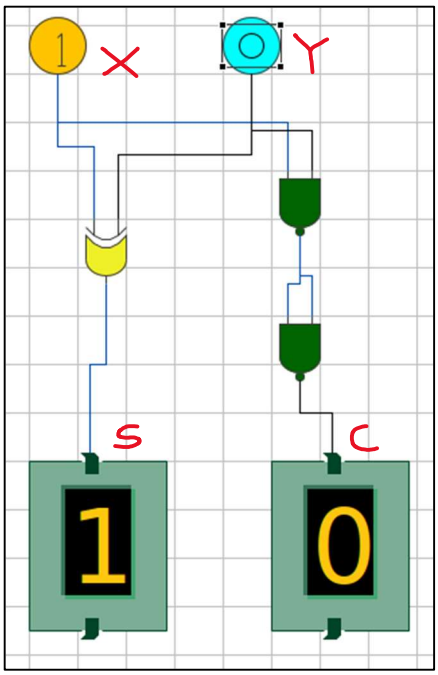

Figure 8:X=1, Y=0; C=0, S=1

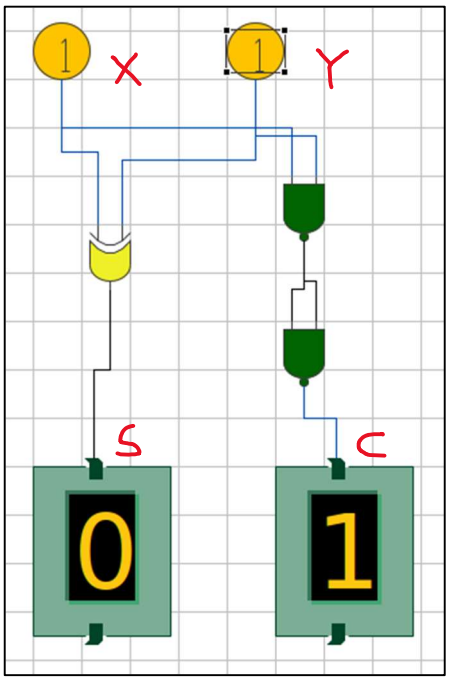

Figure 9:X=1, Y=1; C=1, S=0

2. Full Adder

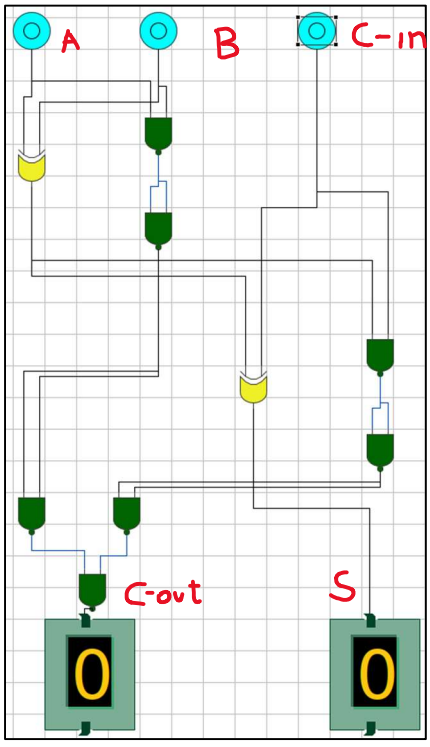

Figure 10: X=0, Y=0, C-in=0; S=0, C-out=0

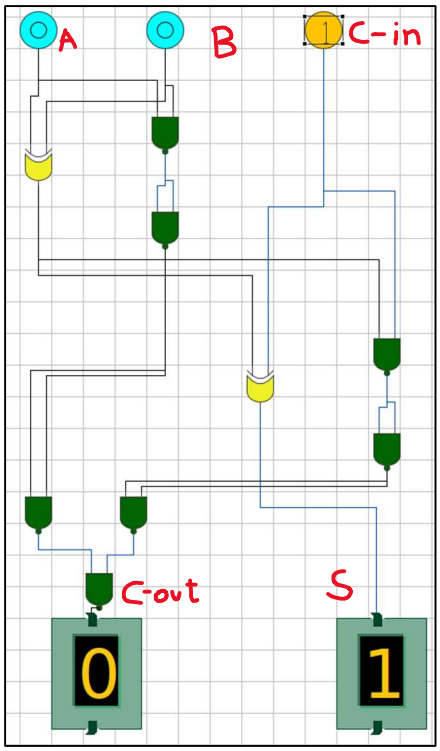

Figure 11: X=0, Y=0, C-in=1; S=1, C-out=0

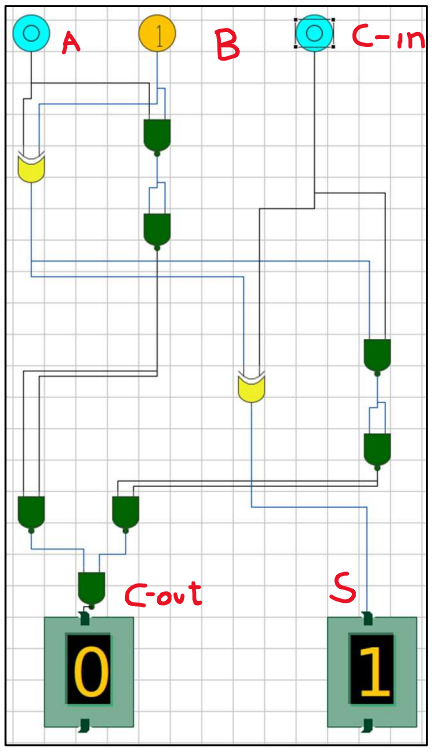

Figure 12: X=0, Y=1, C-in=0; S=1, C-out=0

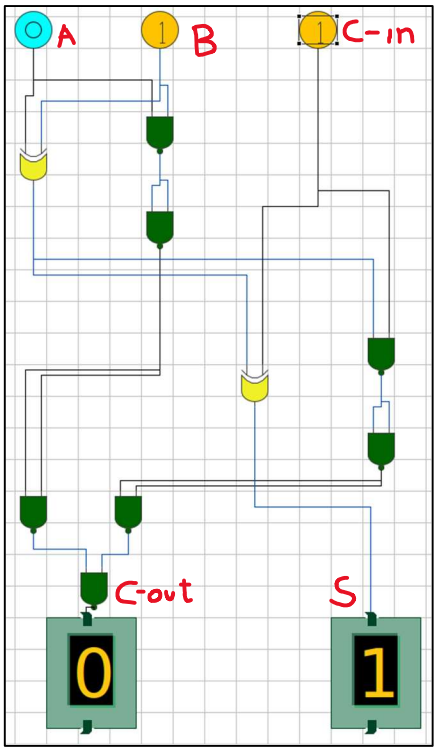

Figure 13: X=0, Y=1, C-in=1; S=0, C-out=1

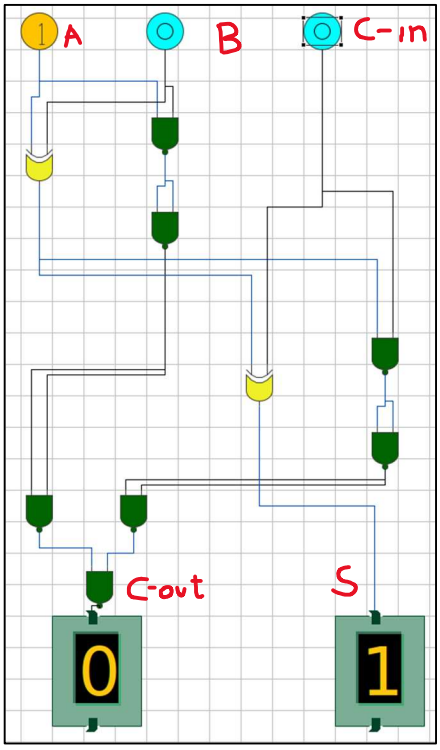

Figure 14: X=1, Y=0, C-in=0; S=1, C-out=0

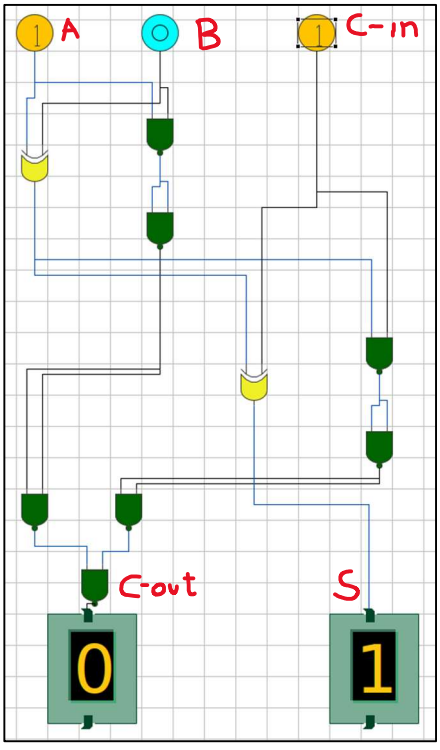

Figure 15: X=1, Y=0, C-in=1; S=0, C-out=1

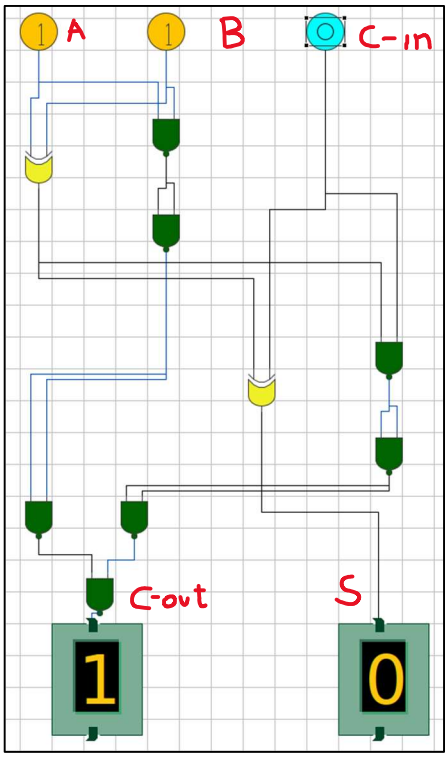

Figure 16: X=1, Y=0, C-in=1; S=0, C-out=1

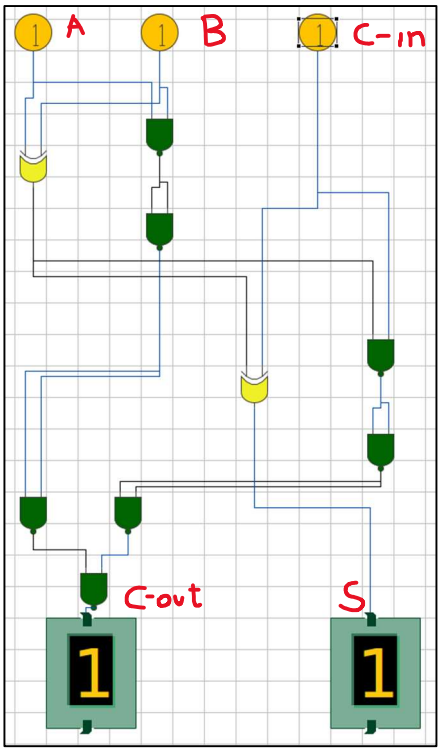

Figure 17: X=1, Y=1, C-in=1; S=1, C-out=1

3. Half Subtractor

- This Circuit give Difference as X-Y

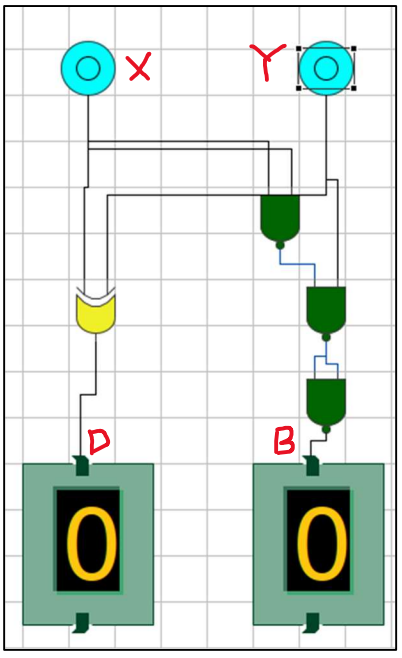

Figure 18: X=0, Y=0; B=0, D=0

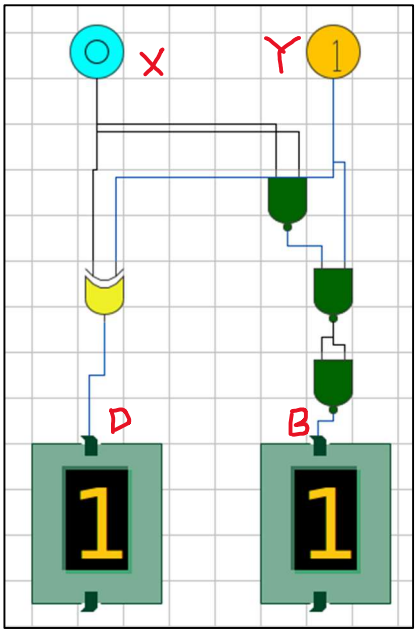

Figure 19: X=0, Y=1; B=1, D=1

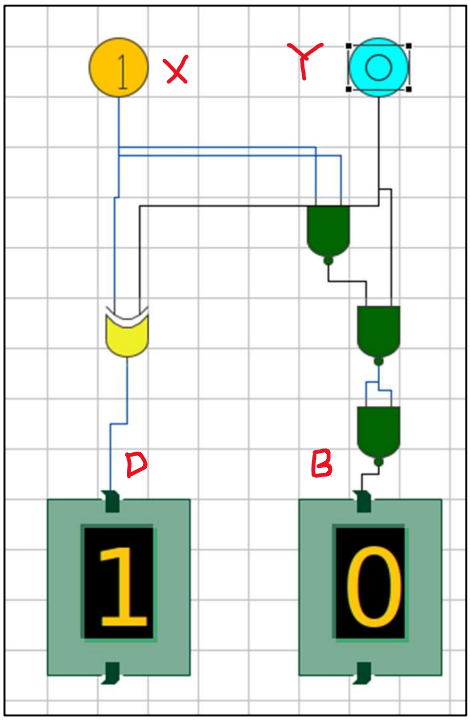

Figure 20: X=1, Y=0; B=0, D=1

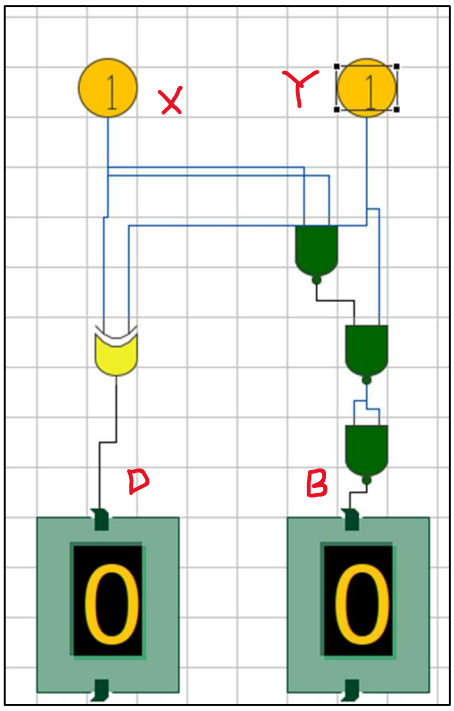

Figure 21: X=1, Y=1; B=0, D=0

1. Full Subtractor

- This Circuit Give Difference as A - B - (B-in)

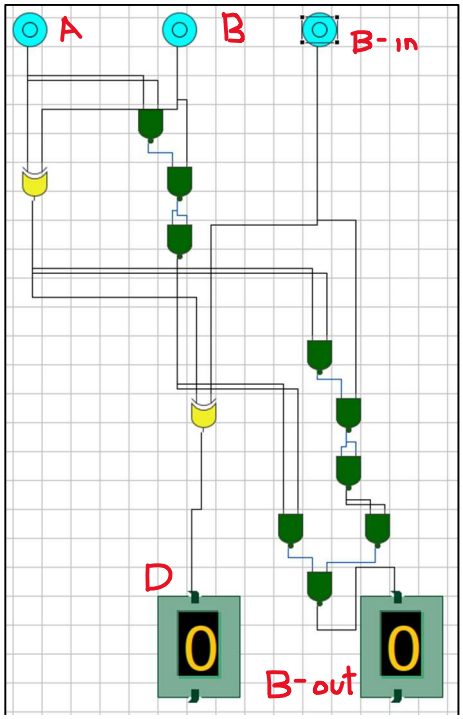

Figure 22: A=0, B=0, B-in=0; D=0, B-out=0

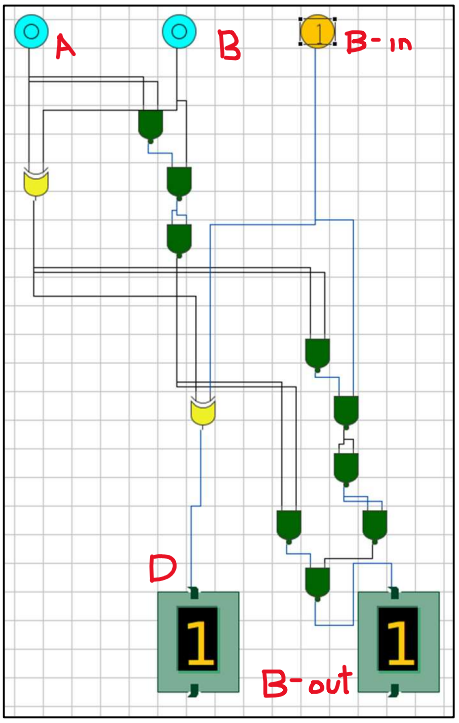

Figure 23: A=0, B=0, B-in=1; D=1, B-out=1

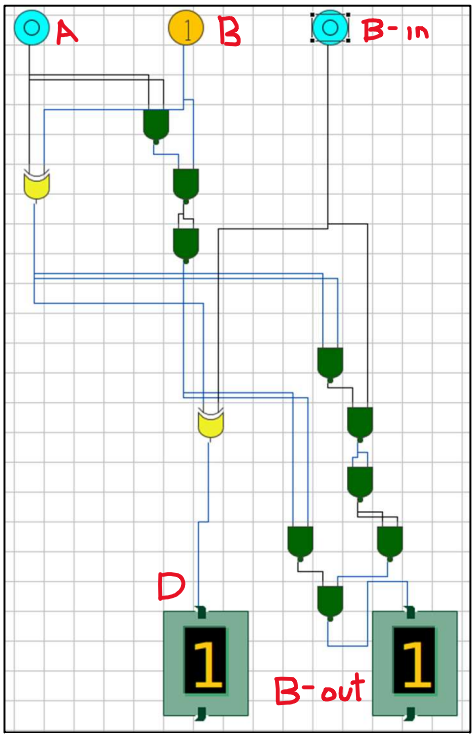

Figure 24: A=0, B=1, B-in=0; D=1, B-out=1

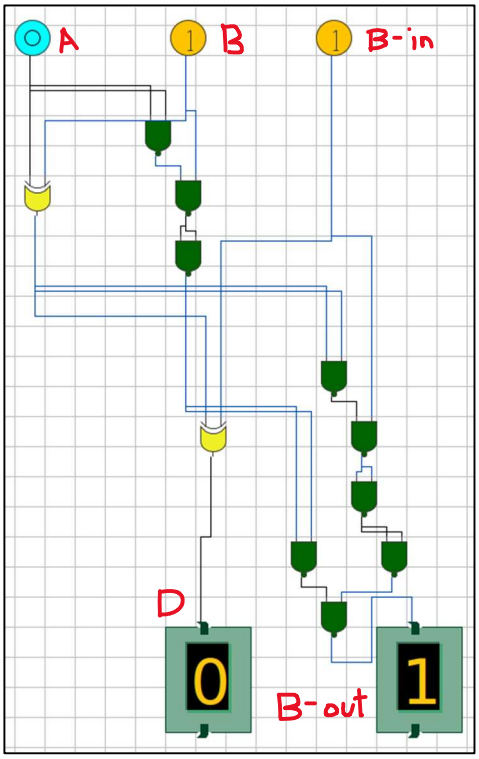

Figure 25: A=0, B=1, B-in=1; D=0, B-out=1

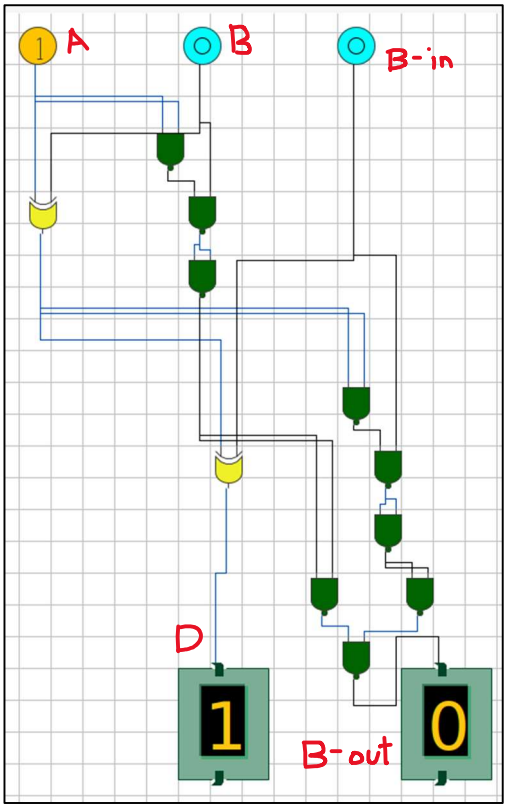

Figure 26: A=1, B=0, B-in=0; D=1, B-out=0

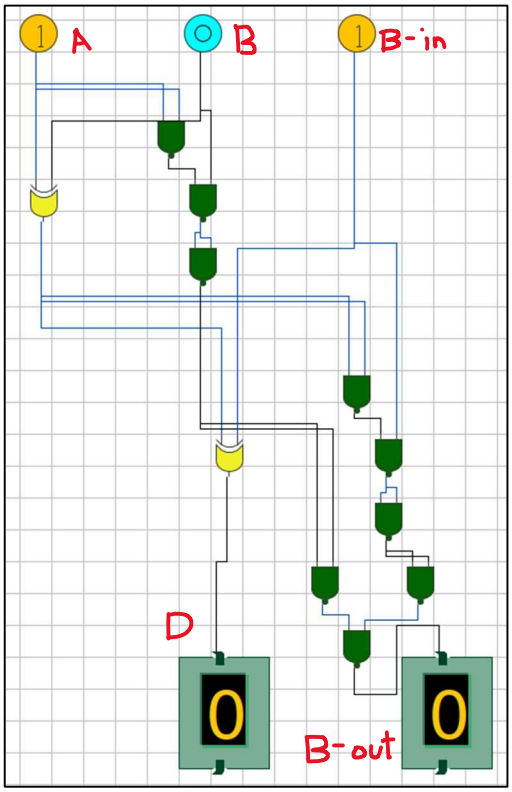

Figure 27: A=1, B=0, B-in=1; D=0, B-out=0

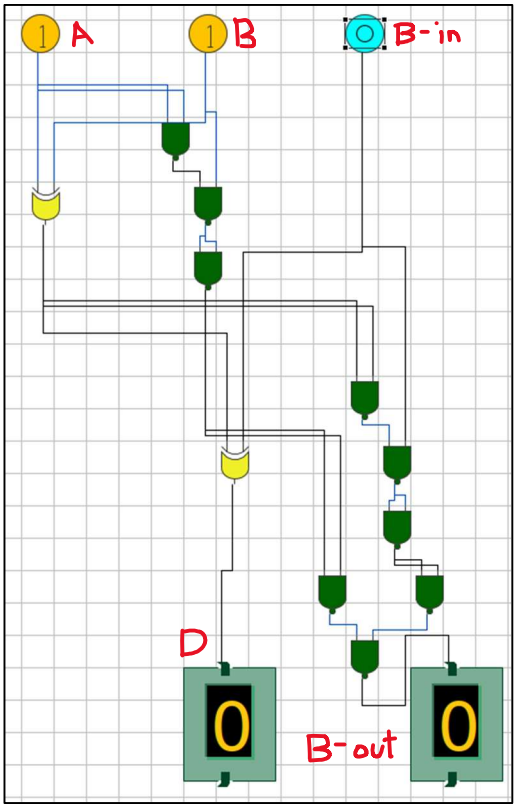

Figure 31: A=1, B=1, B-in=0; D=0, B-out=0

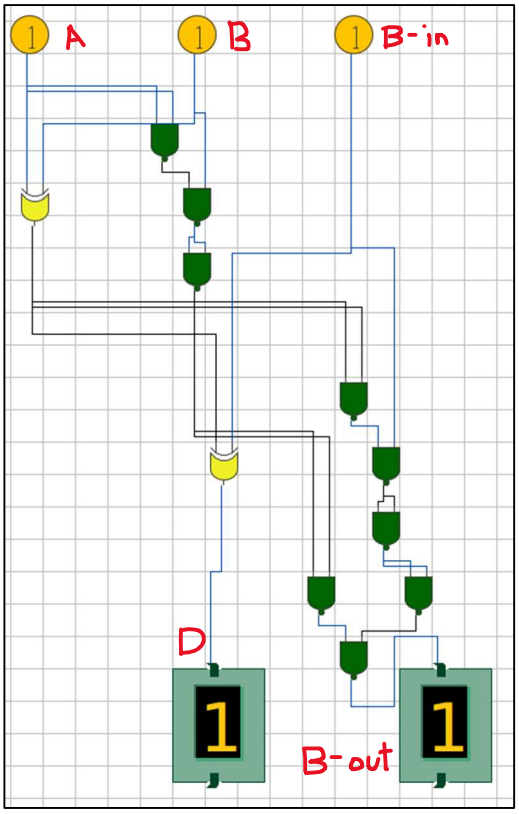

Figure 32: A=1, B=1, B-in=1; D=1, B-out=1

Experiment 2

Objective

Implement BCD-to-Gray code convertor using minimum numbers of two inputs NAND gates.

Theory

- Following Truth Table Exists for BCD to Gray Conversion

Decimal	BCD Input				Gray Code Output			
	B3	B2	B1	B0	D3	D2	D1	D0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1

- Using this Truth Table, we can write that
  - D3 = ∑ (8,9) + ∑d (10,11,12,13,14,15)
  - D2 = ∑ (4,5,6,7,8,9) + ∑d (10,11,12,13,14,15)
  - D1 = ∑ (2,3,4,5) + ∑d (10,11,12,13,14,15)
  - D0 = ∑ (1,2,5,6,9) + ∑d (10,11,12,13,14,15)

- Following K-Maps are made by these expression

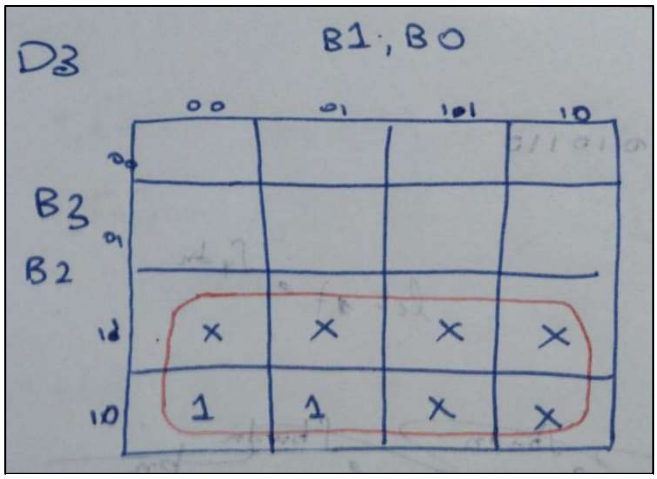

Figure 1: K-Map for D3

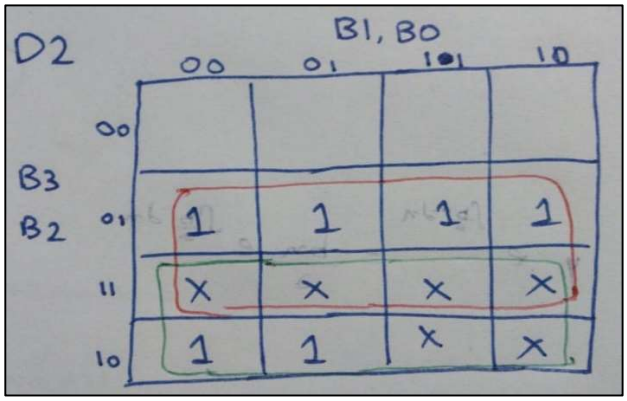

Figure 2: K-Map for D2

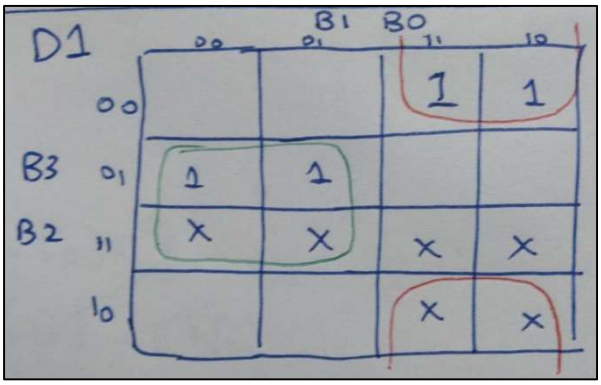

Figure 3: K-Map for D1

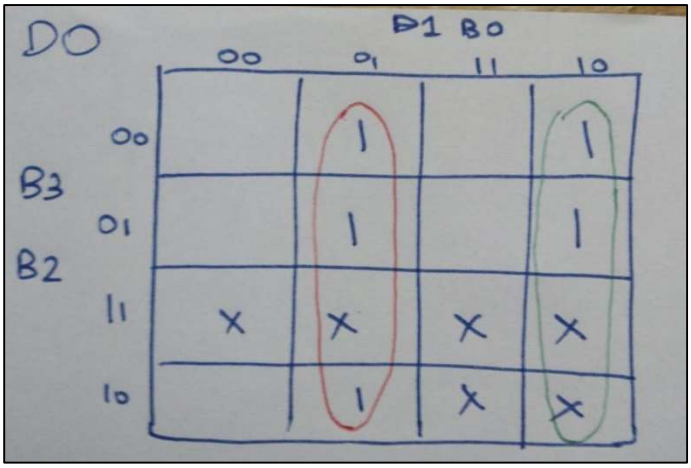

Figure 4: K-Map for D0

- From these K-Maps, we find the following minterms
  - D3 = B3
  - D2 = B2 + B3
  - D1 = B2'.B1 + B2.B1' = B2 XOR B1
  - D0 = B1'.B0 + B1.B0' = B1 XOR B0

Circuit

Based on the minterms found, following circuit was made

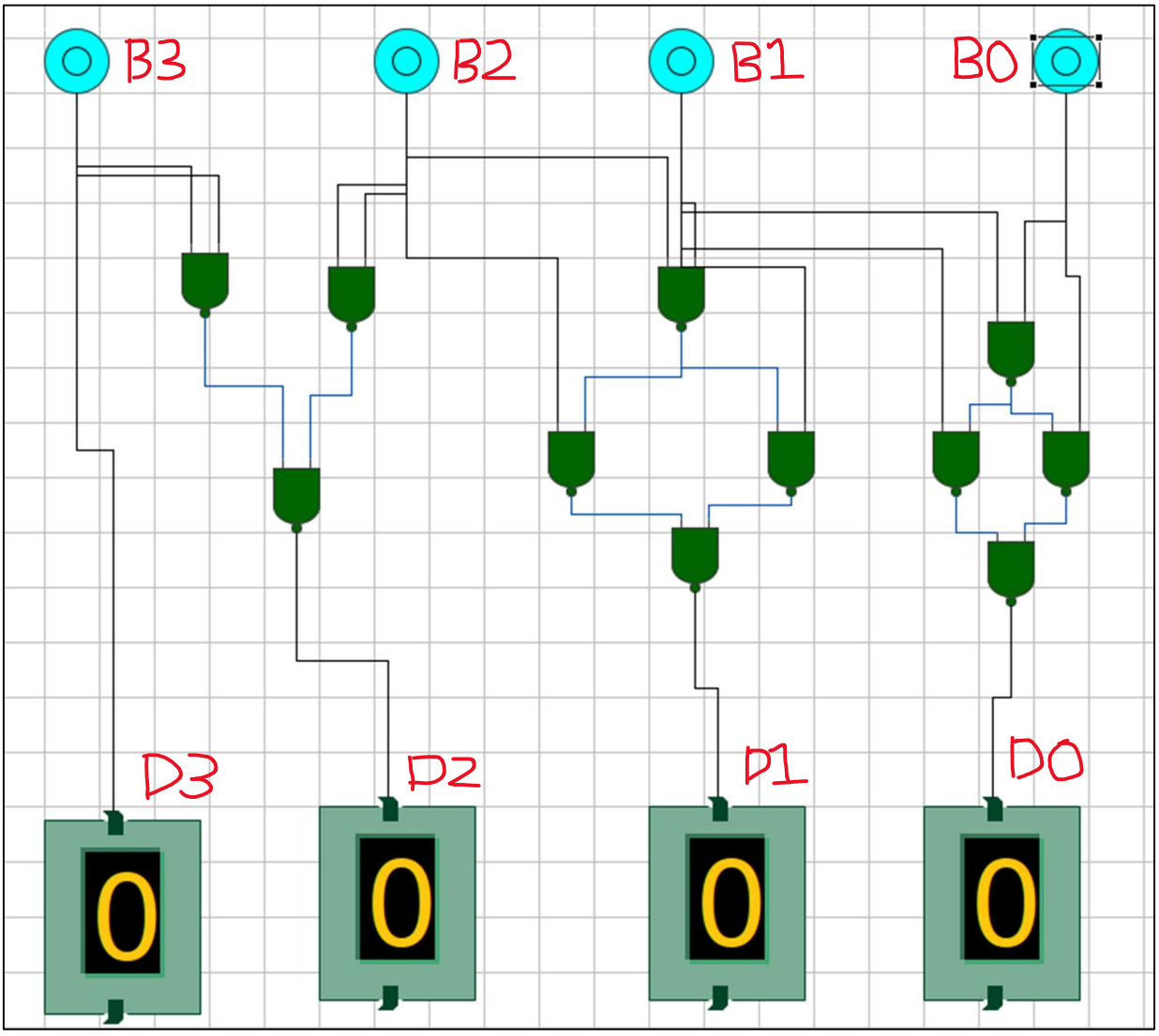

Figure 5: BCD-to-GRAY Code Converter Circuit

Outputs

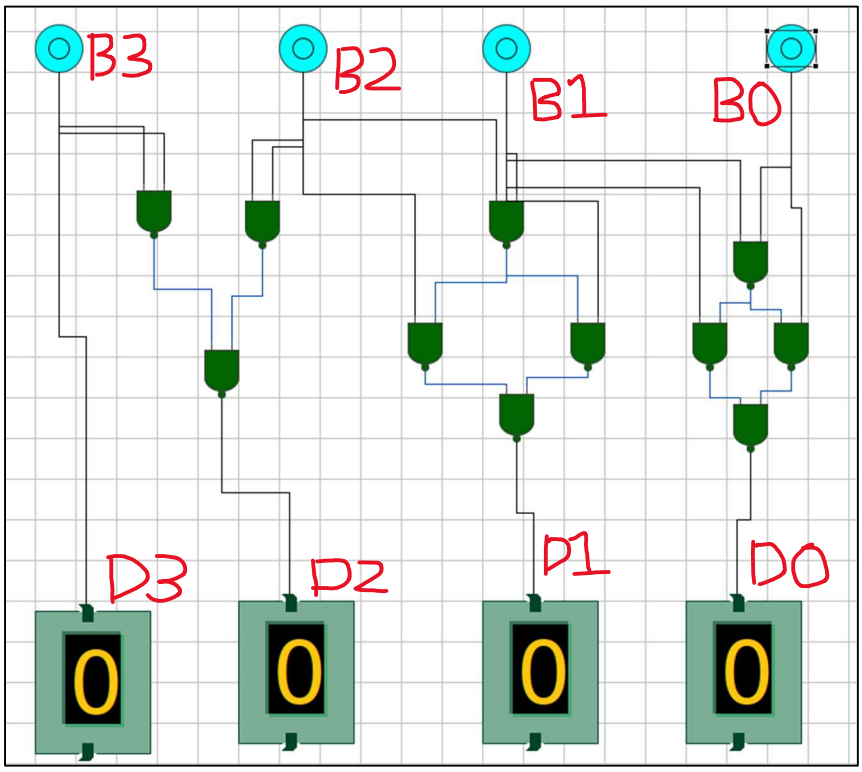

Figure 6: Output for Decimal 0

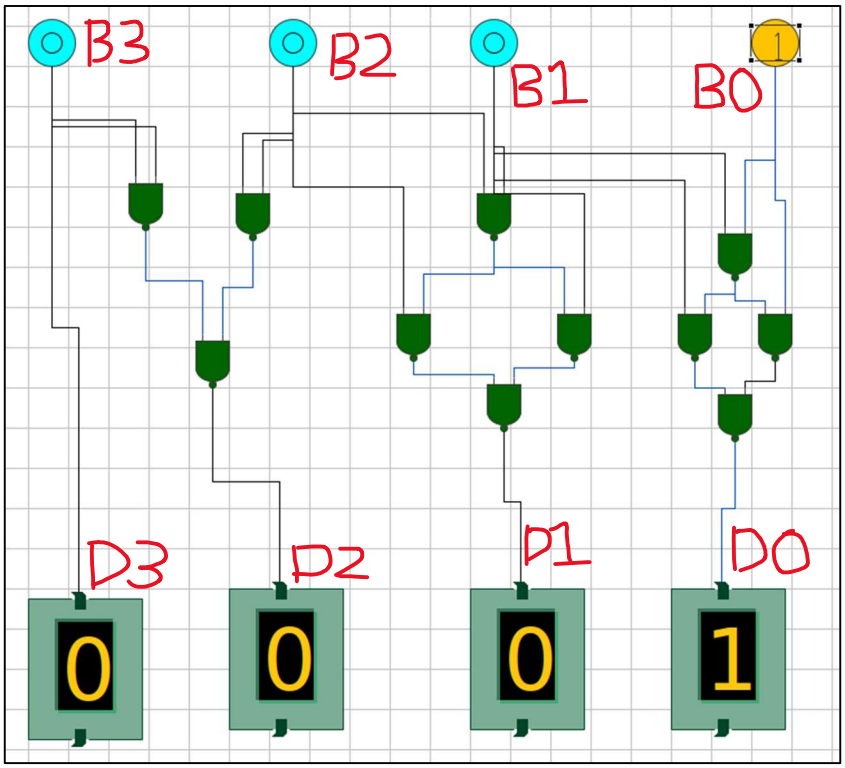

Figure 7: Output for Decimal 1

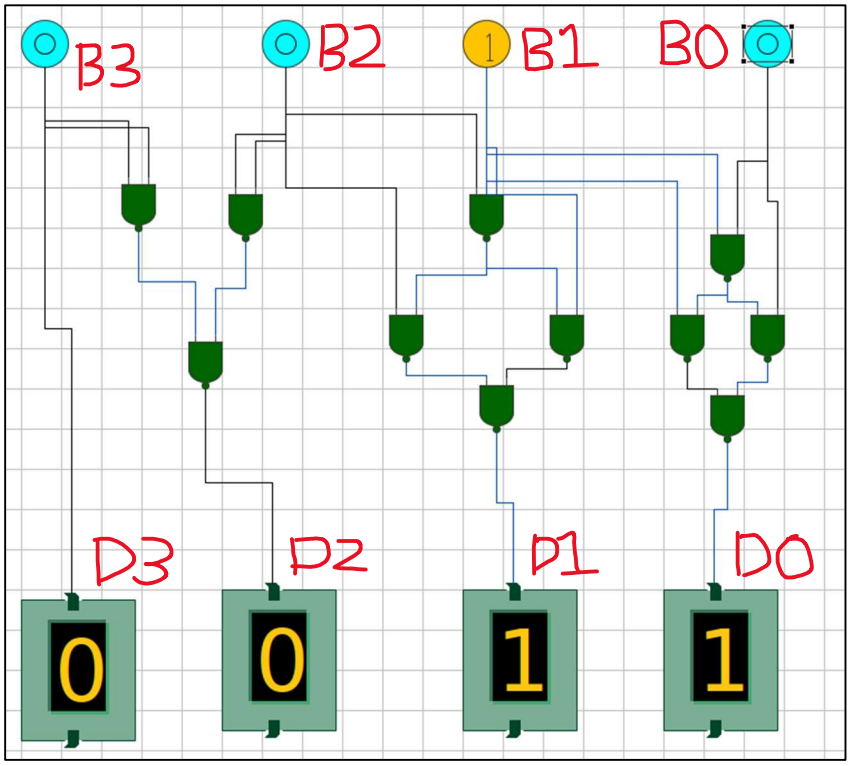

Figure 8: Output for Decimal 2

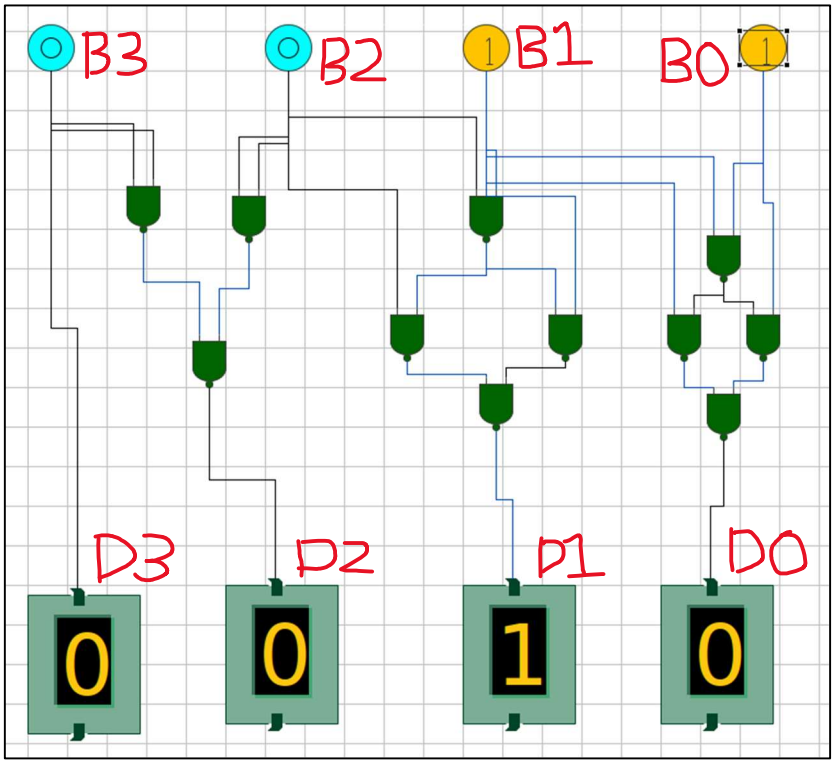

Figure 9: Output for Decimal 3

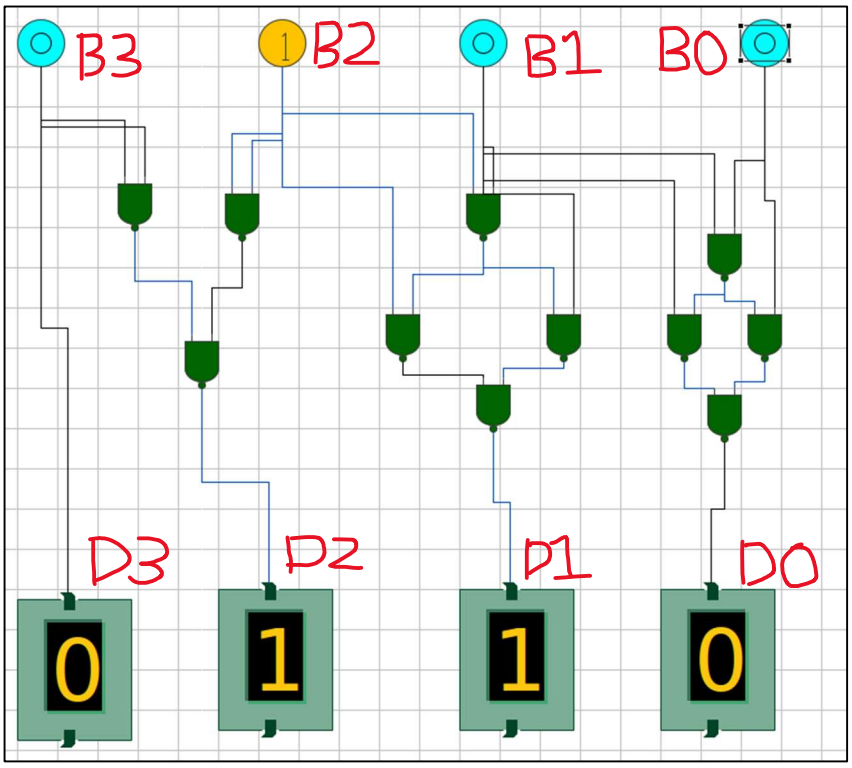

Figure 10: Output for Decimal 4

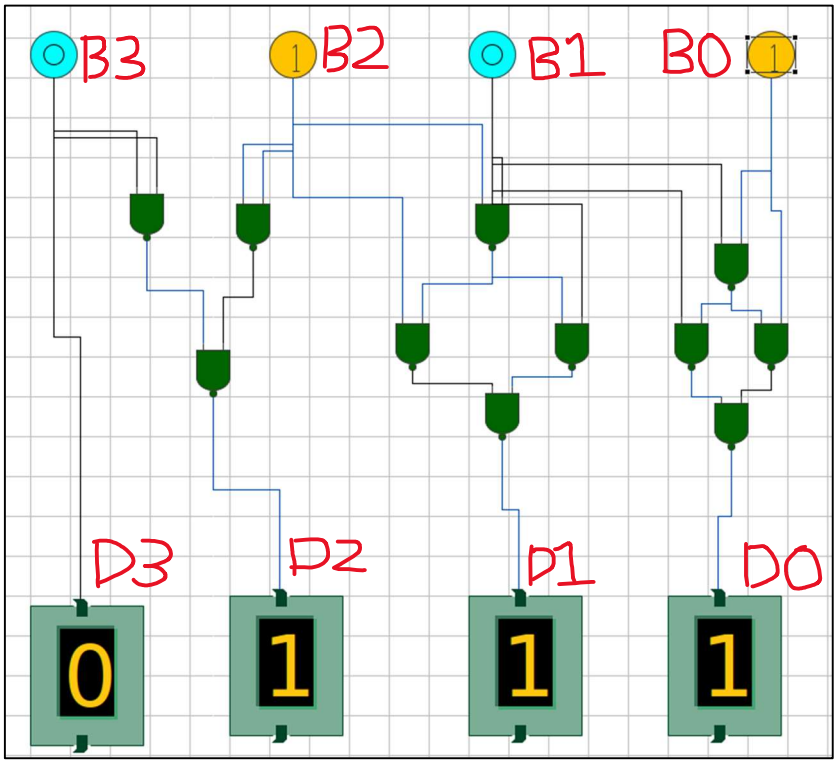

Figure 11: Output for Decimal 5

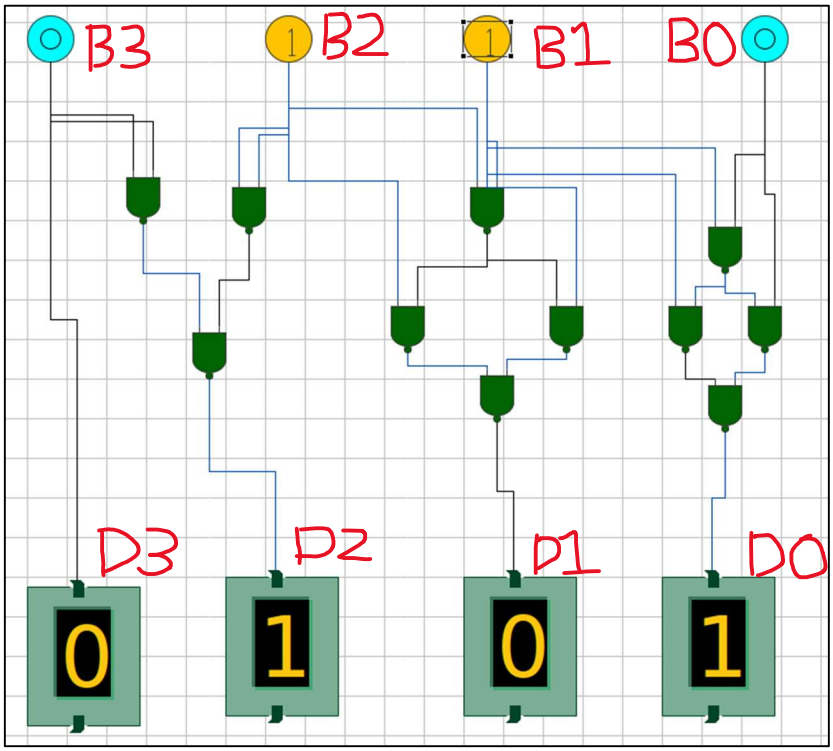

Figure 12: Output for Decimal 6

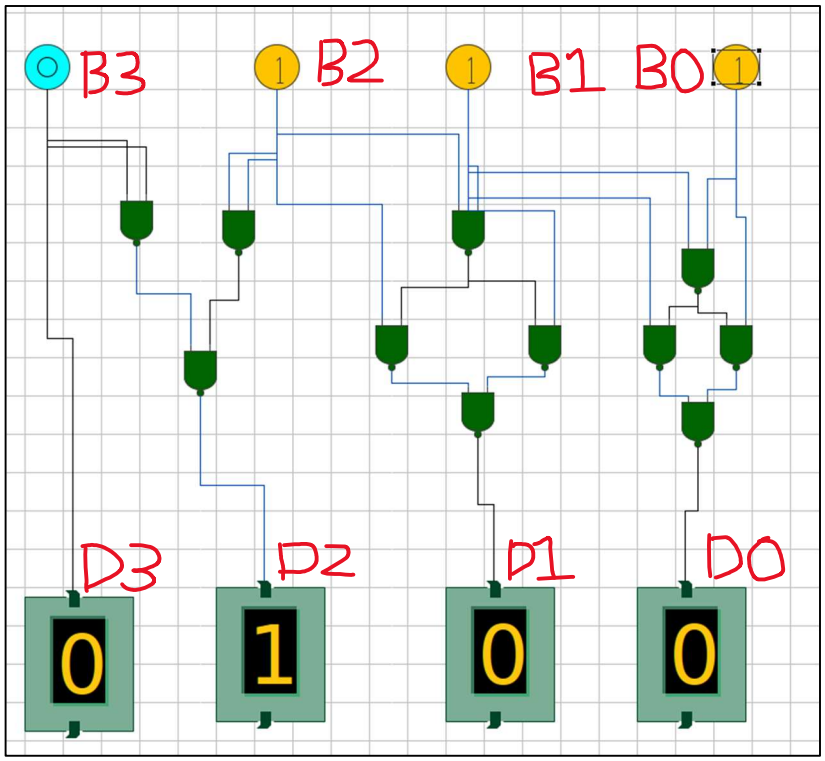

Figure 13: Output for Decimal 7

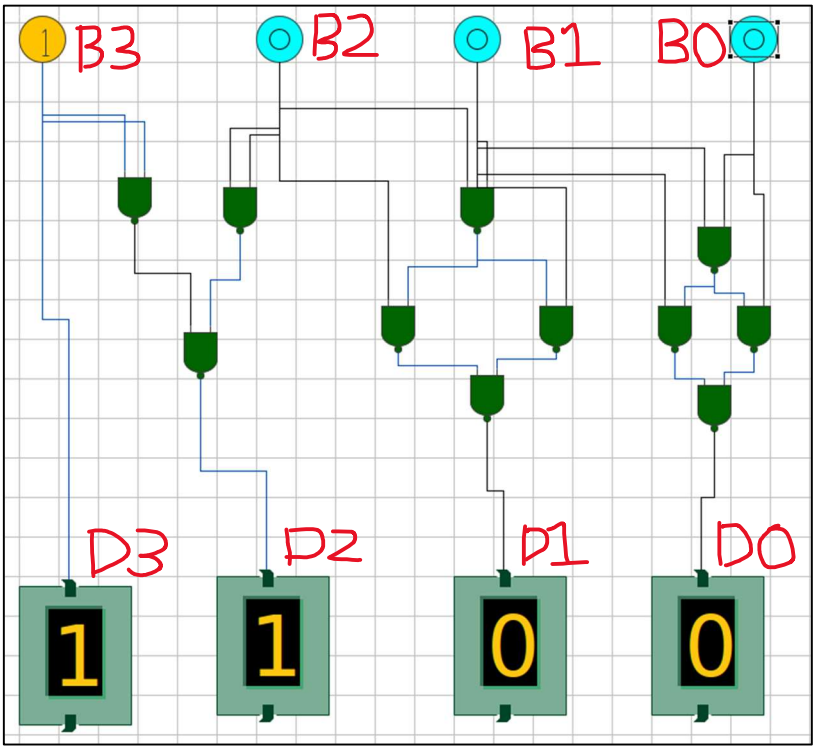

Figure 13: Output for Decimal 8

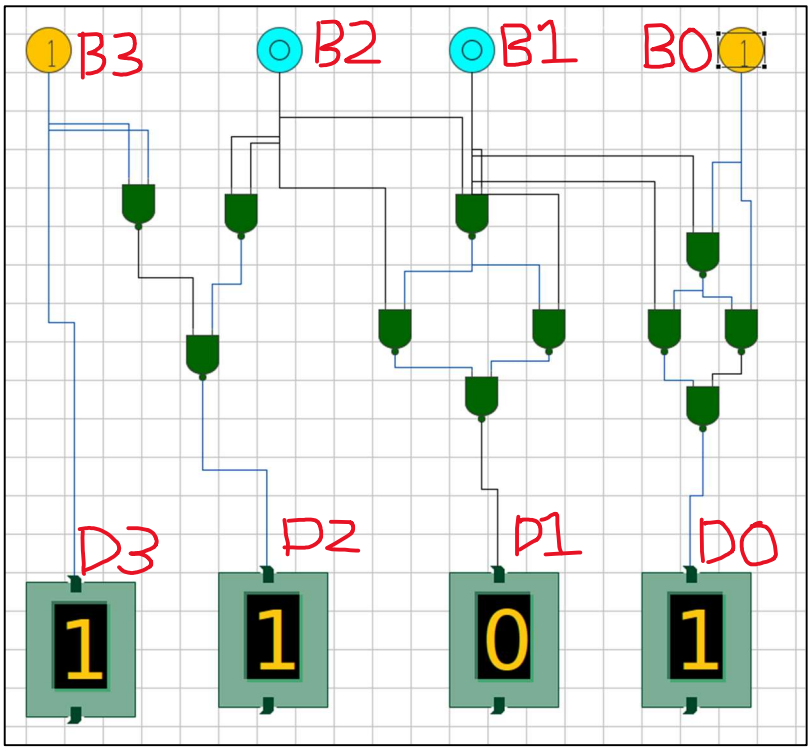

Figure 14: Output for Decimal 9