# ***REPORT ON SUBTRACTOR CIRCUIT***

A **Binary Subtractor** is a decision-making circuit that subtracts two binary numbers from each other, for example, X – Y to find the resulting difference between the two numbers.

Unlike the Binary Adder which produces a SUM and a CARRY bit when two binary numbers are added together, the *binary subtractor* produces a DIFFERENCE, D by using a BORROW bit, B from the previous column.

Binary subtractor circuit are classified into two types***:*** ***half subtractor circuit***

***full subtractor circuit***

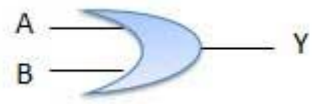
In this project we are designing a full subtractor by only using IC's. So first we need to know about these IC's (logic gates) and the process of subtraction.

In this project we are using ***XOR, AND, NOT, OR*** LOGIC GATES.

***OR GATE: -*** A circuit which performs an OR operation is shown in figure. It has n input (n >= 2) and one output.

$$Y = A \text{ OR } B \text{ OR } C \ldots\ldots N$$
$$Y = A + B + C \ldots\ldots N$$

Logic diagram



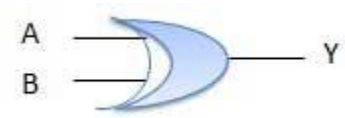

Truth Table

| Inputs | | Output |
|---|---|---|
| A | B | A + B |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

- ***XOR GATE: -*** XOR or Ex-OR gate is a special type of gate. It can be used in the half adder, full adder and subtractor. The exclusive-OR gate is abbreviated as EX-OR gate or sometime as X-OR gate.
  It has n input (n >= 2) and one output.

$$Y = A \text{ XOR } B \text{ XOR } C \ldots\ldots N$$
$$Y = A \oplus B \oplus C \ldots\ldots N$$
$$Y = \overline{A}B + A\overline{B}$$

Logic diagram




Truth Table

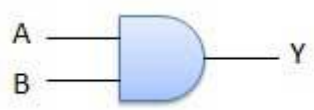
| Inputs | | Output |
|---|---|---|
| A | B | A ⊕ B |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

-

- ***AND GATE: -*** A circuit which performs an AND operation is shown in figure. It has n input (n >= 2) and one output.

Y = A AND B AND C ..... N
Y = A.B.C ..... N
Y = ABC ..... N

Logic diagram



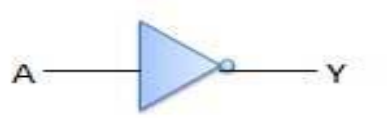

Truth Table

| Inputs | | Output |
|---|---|---|
| A | B | AB |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

- ***NOT GATE: -*** NOT gate is also known as Inverter. It has one input A and one output Y.

Y = NOT A
Y = $\overline{A}$

Logic diagram




Truth Table

| Inputs | Output |
|---|---|
| A | B |
| 0 | 1 |
| 1 | 0 |

- ***BINARY SUBTRACTION: -*** The binary subtraction operation works similarly to the base 10 decimal system, except that it is a base 2 system. The binary system consists of only two digits, 1 and 0. The binary code uses the digits 1's and 0's to make certain processes turn off or on.

The four rules of binary subtraction are:

- **0 - 0 = 0**
- **1 - 0 = 1**
- **1 - 1 = 0**
- **0 - 1 = 1**

(with a borrow of 1)

Let’s take an example, we have to subtract (11101) =29 and (10001) =17. So, result must be 12

1 1 1 0 1

(-) 1 0 0 0 1

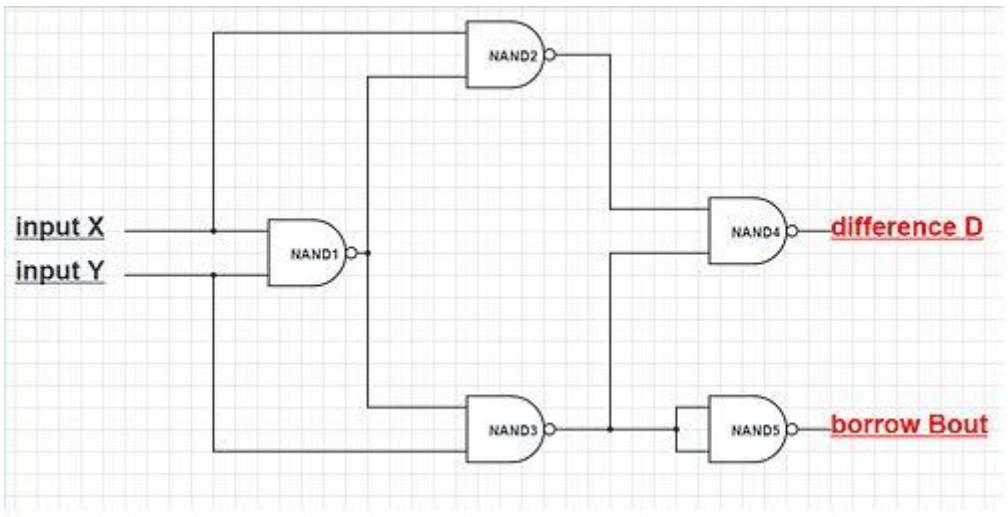
———————

(1 1 0 0) = 12

———————

## HALF SUBTRACTOR

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




### TRUTH TABLE

| Inputs | | Outputs | |
|---|---|---|---|
| ***X*** | ***Y*** | ***D*** | $B_{out}$ |
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 |

From the truth table of the half subtractor we can see that the DIFFERENCE (D) output is the result of the Exclusive-OR gate and the Borrow-out (Bout) is the result of the NOT-AND combination. Then the Boolean expression for a half subtractor is as follows.

For the **DIFFERENCE** bit:

$$D = X \text{ XOR } Y$$

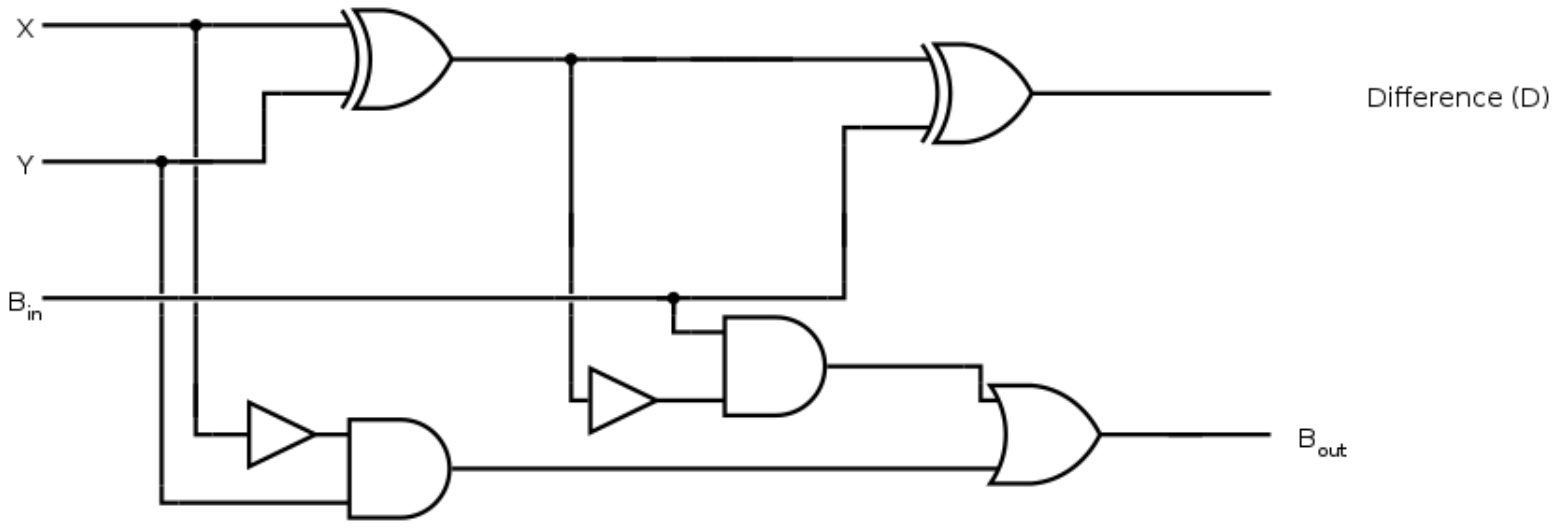
For the **BORROW** bit

$$B = \text{not-}X \text{ AND } Y$$

***FULL SUBTRACTOR:*** **-** The main difference between the Full Subtractor and the previous Half Subtractor circuit is that a full subtractor has three inputs. The two single bit data inputs **X** (**minuend**) and **Y (subtrahend**) the same as before plus an additional ***Borrow-in*** **(B-in**) input to receive the borrow generated by the subtraction process from a previous stage.

The combinational circuit of a “full subtractor” performs the operation of subtraction on three binary bits producing outputs for the difference D and borrow B-out. Just like the binary **adder circuit**, the full subtractor

can also be thought of as two half subtractors connected together, with the first half subtractor passing its borrow to the second half subtractor (same as in adder circuit).




TRUTH TABLE

| Inputs | | | Outputs | |
|---|---|---|---|---|
| ***X*** | ***Y*** | $B_{in}$ | ***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 |

Then the Boolean expression for a full subtractor is as follows.

For the **DIFFERENCE** (D) bit:

$$\mathbf{D} = (X.Y.B_{IN}) + (X.Y.B_{IN}) + (X.Y.B_{IN}) + (X.Y.B_{IN})$$

which can be simplified too:

⇨ $\mathbf{D} = (X \text{ XOR } Y) \text{ XOR } B_{IN} = (X \oplus Y) \oplus B_{IN}$

For the **BORROW OUT** ($B_{OUT}$) bit:

$$\mathbf{B_{OUT}} = (X.Y.B_{IN}) + (X.Y.B_{IN}) + (X.Y.B_{IN}) + (X.Y.B_{IN})$$

which can be simplified too:

⇨ $\mathbf{B_{OUT}} = X \text{ AND } Y \text{ OR } (X \text{ XOR } Y)B_{IN} = X.Y + (X \oplus Y)B_{IN}$

**NOTE: --** The Boolean expressions of the half subtractor with a half adder, we can see that the two expressions for the SUM (adder) and DIFFERENCE (subtractor) are exactly the same and so they should be because of the Exclusive-OR gate function. The two Boolean expressions for the binary subtractor BORROW is also very similar to that for the adders CARRY. Then all that is needed to convert a half adder to a half subtractor is the inversion of the minuend input X.

## PRACTICAL IMPLEMENTATION USING PROTEUS SOFTWARE

**COMPONENTS USED: -** XOR *GATE (IC 4070),* AND GATE (IC 4081 or 7408), OR GATE (IC 4071), NOT GATE LOGICPROBE (binary output) and LOGICSTATE (binary input).

In this project I took 2 numbers A= (1101) = 13 and B= (1100) = 12. So, by subtracting these two numbers we must get 1 (in binary form) as a result. So, in the binaryprobe we must get (00001) as output.

$$\begin{array}{r} 1\ 1\ 0\ 1 \\ (-)1\ 1\ 0\ 0 \\ \hline (00001) = 1 \end{array}$$