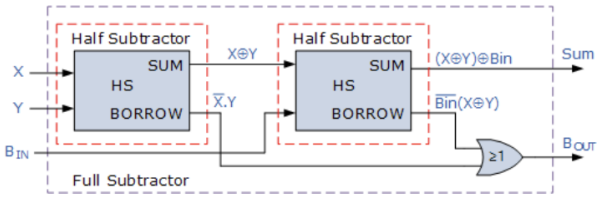


Binary Subtractor

The Binary Subtractor is another type of combinational arithmetic circuit that produces an output which is the subtraction of two binary numbers

As their name implies, 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. Then obviously, the operation of subtraction is the opposite to that of addition.

We learnt from our maths lessons at school that the minus sign, "–" is used for a subtraction calculation, and when one number is subtracted from another, a borrow is required if the subtrahend is greater than the minuend. Consider the simple subtraction of the two denary (base 10) numbers below.

123	X	
– 78	Y	(Subtrahend)
45	DIFFERENCE	

We can not directly subtract 8 from 3 in the first column as 8 is greater than 3, so we have to borrow a 10, the base number, from the next column and add it to the minuend to produce 13 minus 8. This "borrowed" 10 is then return back to the subtrahend of the next column once the difference is found. Simple school math's, borrow a 10 if needed, find the difference and return the borrow.

The subtraction of one binary number from another is exactly the same idea as that for subtracting two decimal numbers but as the *binary number system* is a Base-2 numbering system which uses "0" and "1" as its two independent digits, large binary numbers which are to be subtracted from each other are therefore represented in terms of "0's" and "1's".

Binary Subtraction

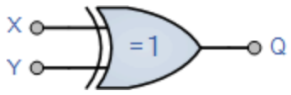
Binary Subtraction can take many forms but the rules for subtraction are the same whichever process you use. As binary notation only has two digits, subtracting a "0" from a "0" or a "1" leaves the result unchanged as 0-0 = 0 and 1-0 = 1. Subtracting a "1" from a "1" results in a "0", but subtracting a "1" from a "0" requires a borrow. In other words 0 – 1 requires a borrow.

Binary Subtraction of Two Bits

0	1	1	(borrow) 1 → 0
− 0	− 0	− 1	− 1
0	1	0	1

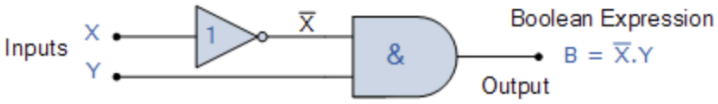
For the simple 1-bit subtraction problem above, if the borrow bit is ignored the result of their binary subtraction resembles that of an Exclusive-OR Gate. To prevent any confusion in this tutorial between a binary subtractor input labelled, B and the resulting borrow bit output from the binary subtractor also being labelled, B, we will label the two input bits as X for the minuend and Y for the subtrahend. Then the resulting truth table is the difference between the two input bits of a single binary subtractor is given as:

2-input Exclusive-OR Gate

Symbol


As with the Binary Adder, the difference between the two digits is only a "1" when these two inputs are not equal as given by the Ex-OR expression. However, we need an additional output to produce the borrow bit when input X = 0 and Y = 1. Unfortunately there are no standard logic gates that will produce an output for this particular combination of X and Y inputs.

But we know that an AND Gate produces an output "1" when both of its inputs X and Y are "1" (HIGH) so if we use an inverter or NOT Gate to complement the input X before it is fed to the AND gate, we can produce the required borrow output when X = 0 and Y = 1 as shown below.

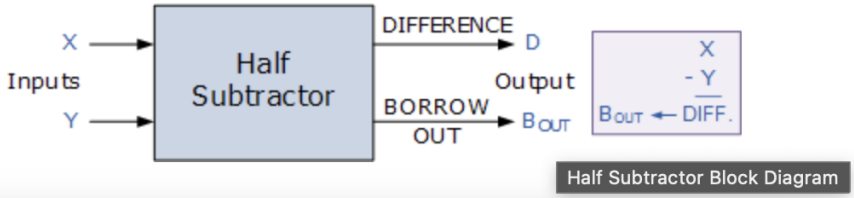


Then by combining the Exclusive-OR gate with the NOT-AND combination results in a simple digital binary subtractor circuit known commonly as the **Half Subtractor** as shown.

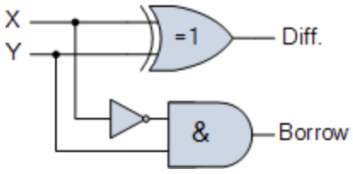
A Half Binary Subtractor Circuit

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.

Half Subtractor with Borrow-out



Half Subtractor Block Diagram

Symbol	Truth Table			
	Y	X	DIFFERENCE	BORROW
	0	0	0	0
	0	1	1	0
	1	0	1	1
	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.

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:

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

For the **BORROW** bit

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

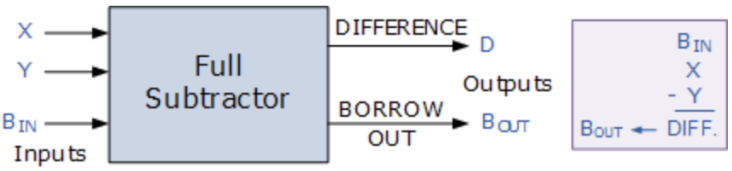
If we compare the Boolean expressions of the half subtractor with a half adder, we can see that the two expressions for the SUM (adder) and DIFFERENCE (subtractor) are exactly the same and so they should be because of the Exclusive-OR gate function. The two Boolean expressions for the binary subtractor 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.

One major disadvantage of the *Half Subtractor* circuit when used as a binary subtractor, is that there is no provision for a "Borrow-in" from the previous circuit when subtracting multiple data bits from each other. Then we need to produce what is called a "full binary subtractor" circuit to take into account this borrow-in input from a previous circuit.

A Full Binary Subtractor Circuit

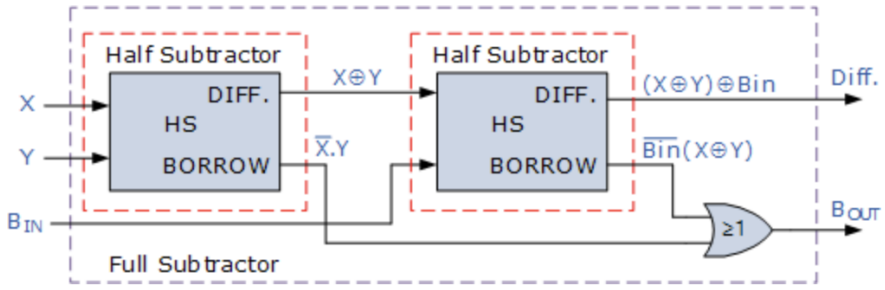
The main difference between the **Full Subtractor** and the previous **Half Subtractor** circuit is that a full subtractor has three inputs. The two single bit data inputs X (minuend) and Y (subtrahend) the same as before plus an additional *Borrow-in* (B-in) input to receive the borrow generated by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

Full Binary Subtractor Block Diagram



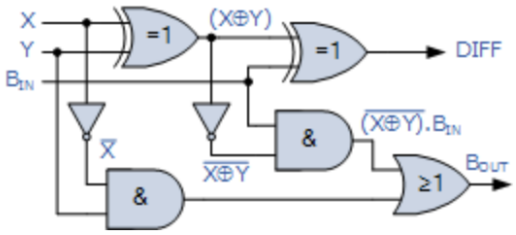
Then the combinational circuit of a "full subtractor" performs the operation of subtraction on three binary bits producing outputs for the difference D and borrow B-out. Just like the binary adder circuit, the full subtractor can also be thought of as two half subtractors connected together, with the first half subtractor passing its borrow to the second half subtractor as follows.

Full Binary Subtractor Logic Diagram



As the full subtractor circuit above represents two half subtractors cascaded together, the truth table for the full subtractor will have eight different input combinations as there are three input variables, the data bits and the *Borrow-in*, B_{IN} input. Also includes the difference output, D and the Borrow-out, B_{OUT} bit.

Full Subtractor Truth Table

Symbol	Truth Table				
	B-in	Y	X	Diff.	B-out
	0	0	0	0	0
	0	0	1	1	0
	0	1	0	1	1
	0	1	1	0	0
	1	0	0	1	1
	1	0	1	0	0
	1	1	0	0	1
	1	1	1	1	1

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

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

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

which can be simplified too:

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

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

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

which will also simplify too:

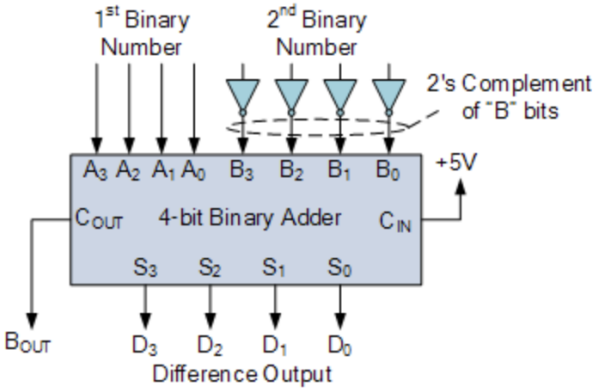
$$B_{OUT} = \overline{X} \text{ AND } Y \text{ OR } \overline{(X \text{ XOR } Y)}B_{IN} = \overline{X}.Y + \overline{(X \oplus Y)}B_{IN}$$

An n-bit Binary Subtractor

As with the binary adder, we can also have n number of 1-bit full binary subtractor connected or "cascaded" together to subtract two parallel n-bit numbers from each other. For example two 4-bit binary numbers. We said before that the only difference between a full adder and a full subtractor was the inversion of one of the inputs.

So by using an n-bit adder and n number of inverters (NOT Gates), the process of subtraction becomes an addition as we can use two's complement notation on all the bits in the subtrahend and setting the carry input of the least significant bit to a logic "1" (HIGH).

Binary Subtractor using 2's Complement



Then we can use a 4-bit full-adder ICs such as the 74LS283 and CD4008 to perform subtraction simply by using two's complement on the subtrahend, B inputs as X – Y is the same as saying, X + (-Y) which equals X plus the two's complement of Y.

If we wanted to use the 4-bit adder for addition once again, all we would need to do is set the carry-in (C_{IN}) input LOW at logic "0". Because we can use the 4-bit adder IC such as the 74LS83 or 74LS283 as a full-adder or a full-subtractor they are available as a single adder/subtractor circuit with a single control input for selecting between the two operations.