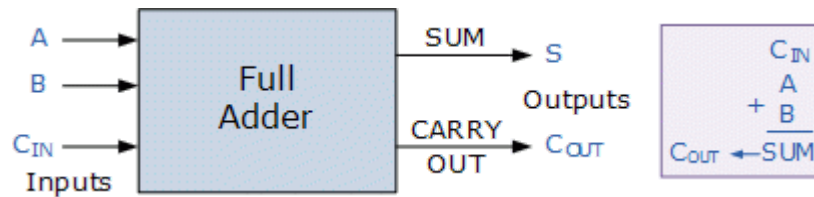


[Home](#) / [Combinational Logic](#) / Binary Adder

Credits: [www.electronics-tutorial.ws](http://www.electronics-tutorial.ws)



## Binary Adder (HALF ADDER & FULL ADDER)

Binary Adders are arithmetic circuits in the form of half-adders and full-adders used to add together two binary digits

Another common and very useful combinational logic circuit which can be constructed using just a few basic logic gates allowing it to add together two or more binary numbers is the **Binary Adder**.

A basic Binary Adder circuit can be made from standard AND and Ex-OR gates allowing us to “add” together two single bit binary numbers, A and B.

The addition of these two digits produces an output called the SUM of the addition and a second output called the CARRY or Carry-out, ( $C_{OUT}$ ) bit according to the rules for binary addition. One of the main uses for the *Binary Adder* is in arithmetic and counting circuits. Consider the simple addition of the two denary (base 10) numbers below.

123	A	(Augend)
+ 789	<u>B</u>	(Addend)
912	SUM	

From our maths lessons at school, we learnt that each number column is added together starting from the right hand side and that each digit has a weighted value depending upon its position within the columns.

When each column is added together a carry is generated if the result is greater or equal to 10, the base number. This carry is then added to the result of the addition of the next column to the left and so on, simple school math’s addition, add the numbers and carry.

The adding of binary numbers is exactly the same idea as that for adding together decimal numbers but this time a carry is only generated when the result in any column is greater or equal to “2”, the base number of binary. In other words  $1 + 1$  creates a carry.

## Binary Addition

**Binary Addition** follows these same basic rules as for the denary addition above except in binary there are only two digits with the largest digit being "1". So when adding binary numbers, a carry out is generated when the "SUM" equals or is greater than two (1+1) and this becomes a "CARRY" bit for any subsequent addition being passed over to the next column for addition and so on. Consider the single bit addition below.

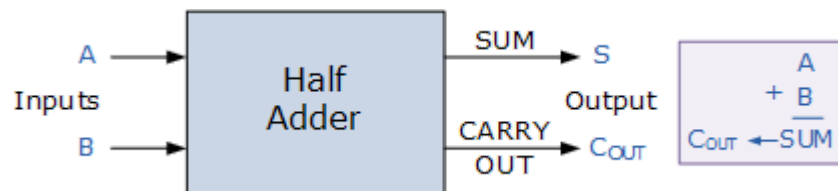
### Binary Addition of Two Bits

0	0	1	1
<u>+ 0</u>	<u>+ 1</u>	<u>+ 0</u>	<u>+ 1</u>
0	1	1 (carry)	1 ← 0

When the two single bits, A and B are added together, the addition of "0 + 0", "0 + 1" and "1 + 0" results in either a "0" or a "1" until you get to the final column of "1 + 1" then the sum is equal to "2". But the number two does not exist in binary however, 2 in binary is equal to 10, in other words a zero for the sum plus an extra carry bit.

Then the operation of a simple adder requires two data inputs producing two outputs, the Sum (S) of the equation and a Carry (C) bit as shown.

### Binary Adder Block Diagram



For the simple 1-bit addition problem above, the resulting carry bit could be ignored but you may have noticed something else with regards to the addition of these two bits, the sum of their binary addition resembles that of an **Exclusive-OR Gate**. If we label the two bits as A and B then the resulting truth table is the sum of the two bits but without the final carry.

### 2-input Exclusive-OR Gate

Symbol	Truth Table		
<p>2-input Ex-OR Gate</p>	B	A	S
	0	0	0
	0	1	1

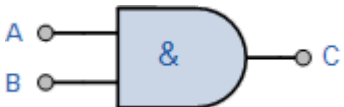
	1	0	1
	1	1	0

We can see from the truth table above, that an Exclusive-OR gate only produces an output “1” when either input is at logic “1”, but not both the same as for the binary addition of the previous two bits. However in order to perform the addition of two numbers, microprocessors and electronic calculators require the extra carry bit to correctly calculate the equations so we need to rewrite the previous summation to include two-bits of output data as shown below.

00	00	01	01
<u>+ 00</u>	<u>+ 01</u>	<u>+ 00</u>	<u>+ 01</u>
00	01	01	10

From the above equations we now know that an **Exclusive-OR** gate will only produce an output “1” when “EITHER” input is at logic “1”, so we need an additional output to produce the carry bit when “BOTH” inputs A and B are at logic “1”. One digital gate that fits the bill perfectly producing an output “1” when both of its inputs A and B are “1” (HIGH) is the standard **AND Gate**.

## 2-input AND Gate

Symbol	Truth Table		
 <p>2-input AND Gate</p>	B	A	C
	0	0	0
	0	1	0
	1	0	0
	1	1	1

By combining the Exclusive-OR gate with the AND gate results in a simple digital binary adder circuit known commonly as the “**Half Adder**” circuit.

## A Half Adder Circuit

A half adder is a logical circuit that performs an addition operation on two binary digits. The half adder produces a sum and a carry value which are both binary digits.

## Half Adder Truth Table with Carry-Out

Symbol	Truth Table			
	B	A	SUM	CARRY
	0	0	0	0
	0	1	1	0
	1	0	1	0
	1	1	0	1

From the truth table of the half adder we can see that the SUM (S) output is the result of the Exclusive-OR gate and the Carry-out (Cout) is the result of the AND gate. Then the Boolean expression for a half adder is as follows.

For the **SUM** bit:

$$\text{SUM} = A \text{ XOR } B = A \oplus B$$

For the **CARRY** bit:

$$\text{CARRY} = A \text{ AND } B = A \cdot B$$

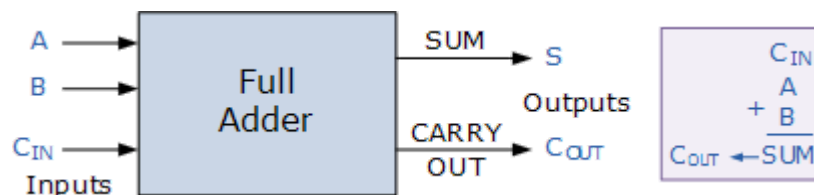
One major disadvantage of the *Half Adder* circuit when used as a binary adder, is that there is no provision for a “Carry-in” from the previous circuit when adding together multiple data bits.

For example, suppose we want to add together two 8-bit bytes of data, any resulting carry bit would need to be able to “ripple” or move across the bit patterns starting from the least significant bit (LSB). The most complicated operation the half adder can do is “1 + 1” but as the half adder has no carry input the resultant added value would be incorrect. One simple way to overcome this problem is to use a **Full Adder** type binary adder circuit.

## A Full Adder Circuit

The main difference between the **Full Adder** and the previous **Half Adder** is that a full adder has three inputs. The same two single bit data inputs A and B as before plus an additional *Carry-in* (C-in) input to receive the carry from a previous stage as shown below.

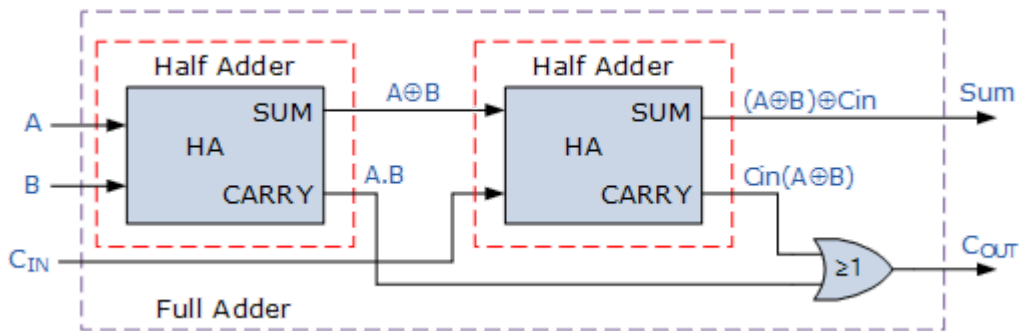
## Full Adder Block Diagram



Then the **full adder** is a logical circuit that performs an addition operation on three binary digits and just like the half adder, it also generates a carry out to the next addition column. Then a *Carry-in* is a possible carry from a less significant digit, while a *Carry-out* represents a carry to a more significant digit.

In many ways, the full adder can be thought of as two half adders connected together, with the first half adder passing its carry to the second half adder as shown.

## Full Adder Logic Diagram



As the full adder circuit above is basically two half adders connected together, the truth table for the full adder includes an additional column to take into account the *Carry-in*,  $C_{IN}$  input as well as the summed output,  $S$  and the Carry-out,  $C_{OUT}$  bit.

## Full Adder Truth Table with Carry

Symbol	Truth Table				
	C-in	B	A	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

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

For the **SUM** (S) bit:

$$\text{SUM} = (A \text{ XOR } B) \text{ XOR } C_{in} = (A \oplus B) \oplus C_{in}$$

For the **CARRY-OUT** (Cout) bit:

$$\text{CARRY-OUT} = A \text{ AND } B \text{ OR } C_{in}(A \text{ XOR } B) = A \cdot B + C_{in}(A \oplus B)$$