# 1-bit Full Adder

Introduction

A 1-bit adder takes two 1-bit inputs and adds them together. To make it a full adder, it also needs to consider a carry in and carry out flag. Thus a 1-bit full adder takes three 1-bit inputs and contains two 1-bit outputs. The first two inputs are the two bits that are to be added together, respectively A and B. The third input is a carry in flag. This flag specifies whether or not a previous addition has occurred which contained a carry out. The first output is the 1-bit result of the addition. The second output is the carry out flag which specifies if the result of the addition was larger than the 1-bit result.

Purpose of Including Carry In and Carry Out Flags

When adding numbers using circuits it is necessary to look at each bit of a number at a time. The least significant bits are added first, with a carry in flag set to 0. The next two least significant bits are added together, using a carry in flag set to the carry out flag of the previous operationg. This approach allows the chaining 1-bit full adders together to make n-bit full adders.

As the binary number system only contains 1's and 0's, it is not necessary to account for a carry out of any value other than 0 or 1, making the circuit design much simpler than it would be if we were trying to work with decimal number system!

Circuit Diagram
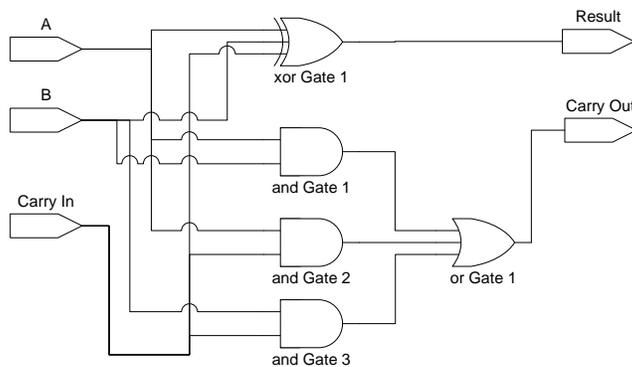
Logic Information

Inputs:
A – 1-bit
B – 1-bit
Carry In Flag – 1-bit

Output:
Result – 1-Bit
Carry Out – 1-Bit

Gates:

xor Gate 1: $A \oplus B \oplus C$

and Gate 1: $A \cdot B$

and Gate 2: $A \cdot CarryIn$

and Gate 3: $B \cdot CarryIn$

or Gate 1: $andGate1 \cdot andGate2 \cdot andGate3$

VHDL Code

```
library IEEE;
use IEEE.std_logic_1164.all;

entity onebitfulladder is
port (num1, num2, carryin : in std_logic;
        result, carryout : out std_logic);

end onebitfulladder;

architecture behavior of onebitfulladder is
signal S1, S2, S3 : std_logic;
begin
        result <= num1 xor num2 xor carryin;
        S1 <= num1 and num2;
        S2 <= num1 and carryin;
        S3 <= carryin and num2;
        Carryout <= S1 or S2 or S3;
end behavior;
```