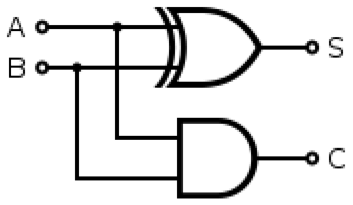
## Adder

A half adder can add two bits. It has two inputs, generally labeled ***A*** and ***B***, and two outputs, the sum ***S*** and carry ***C***. ***S*** is the two-bit **XOR** of ***A*** and ***B***, and ***C*** is the ***AND*** of ***A*** and ***B***. Essentially the output of a half adder is the sum of two one-bit numbers, with ***C*** being the most significant of these two outputs.

A **full adder** is capable of adding three bits: two bits and one carry bit. It has three inputs - ***A***, ***B***, and carry ***C***, such that multiple full adders can be used to add larger numbers. To remove ambiguity between the input and output carry lines, the carry in is labelled $C_i$ or $C_{in}$ while the carry out is labelled $C_o$ or $C_{out}$.

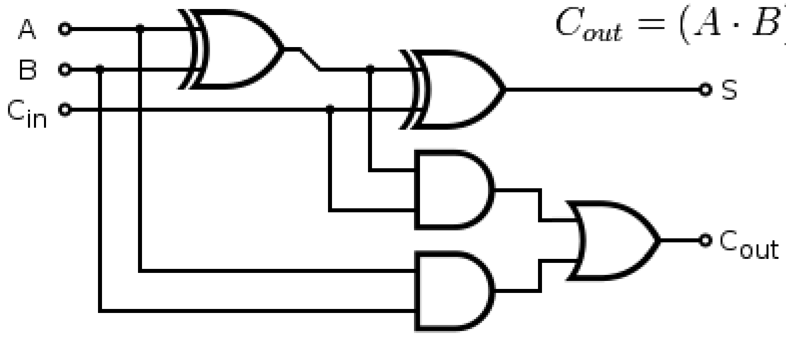


Half Adder

$$S = A \oplus B$$
$$C = A \cdot B$$

| $A$ | $B$ | $C$ | $S$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

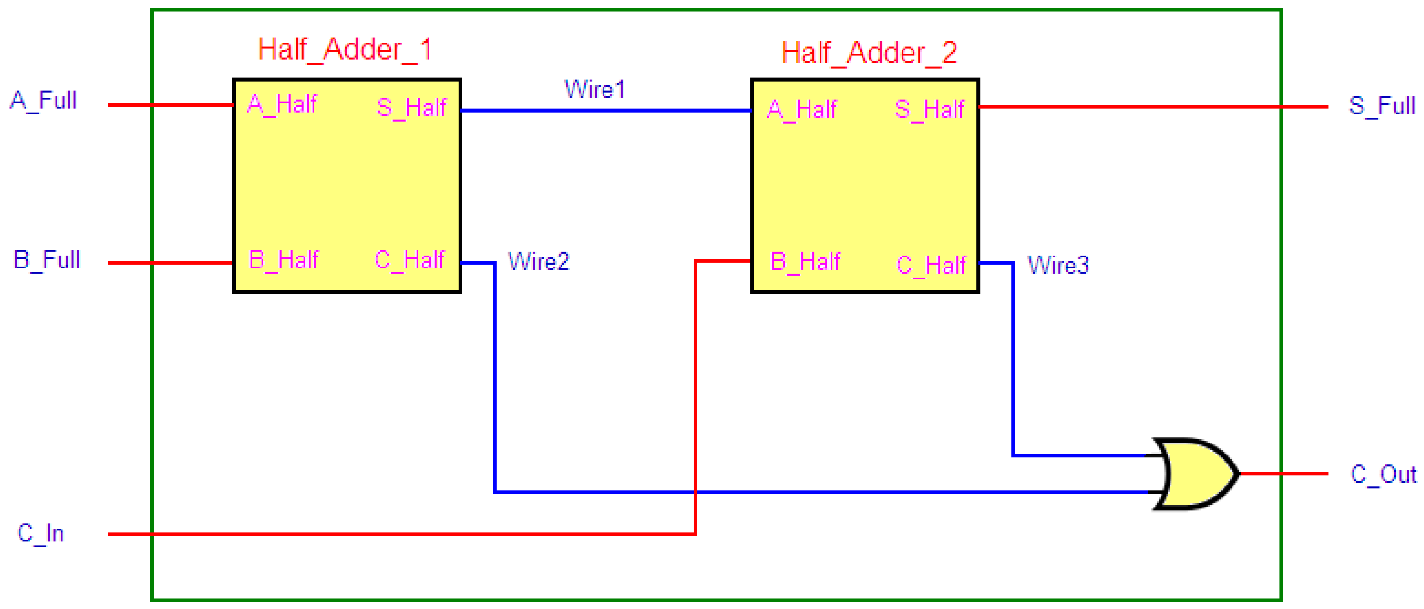
$$S = (A \oplus B) \oplus C_{in}$$
$$C_{out} = (A \cdot B) + (C_{in} \cdot (A \oplus B))$$



Full Adder

| Input | | | Output | |
|---|---|---|---|---|
| $A$ | $B$ | $C_i$ | $C_o$ | $S$ |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

## Full Adder



VHDL CODE

```
==============================================================================
H_Adder.vhdl
==============================================================================
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--  Hendra Kesuma

entity Half_Adder_ent is
    Port ( A_Half : in std_logic;
           B_Half : in std_logic;
           S_Half : out std_logic;
           C_Half : out std_logic);
end Half_Adder_ent;

architecture Half_Adder_arch of Half_Adder_ent is
begin
  process(A_Half, B_Half)
  begin
    S_Half <= (A_Half xor B_Half);
    C_Half <= (A_Half and B_Half);
  end process;
end Half_Adder_arch;
```

VHDL CODE

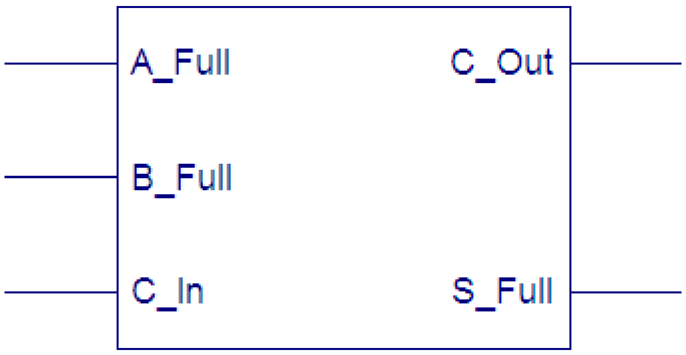
```
==========================================================================
Full_Adder.vhdl
==========================================================================
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

--  Hendra Kesuma

entity Full_Adder_ent is
   Port ( A_Full : in std_logic;
          B_Full : in std_logic;
          C_In : in std_logic;
          S_Full : out std_logic;
          C_Out : out std_logic);
end Full_Adder_ent;

architecture Full_Adder_arch of Full_Adder_ent is

component Half_Adder_ent
port(
     A_Half : in std_logic;
     B_Half : in std_logic;
     S_Half : out std_logic;
     C_Half : out std_logic);
end component;

  signal Wire1 : std_logic;
  signal Wire2 : std_logic;
  signal Wire3 : std_logic;


begin
  Half_Adder_1: Half_Adder_ent
  port map(
     A_Half => A_Full,
     B_Half => B_Full,
     S_Half => Wire1,
     C_Half => Wire2
     );

  Half_Adder_2: Half_Adder_ent
  port map(
     A_Half => Wire1,
     B_Half => C_In,
     S_Half => S_Full,
     C_Half => Wire3
     );
  C_Out <= (Wire3 or Wire2);

end Full_Adder_arch;
```

TESTBENCH CODE

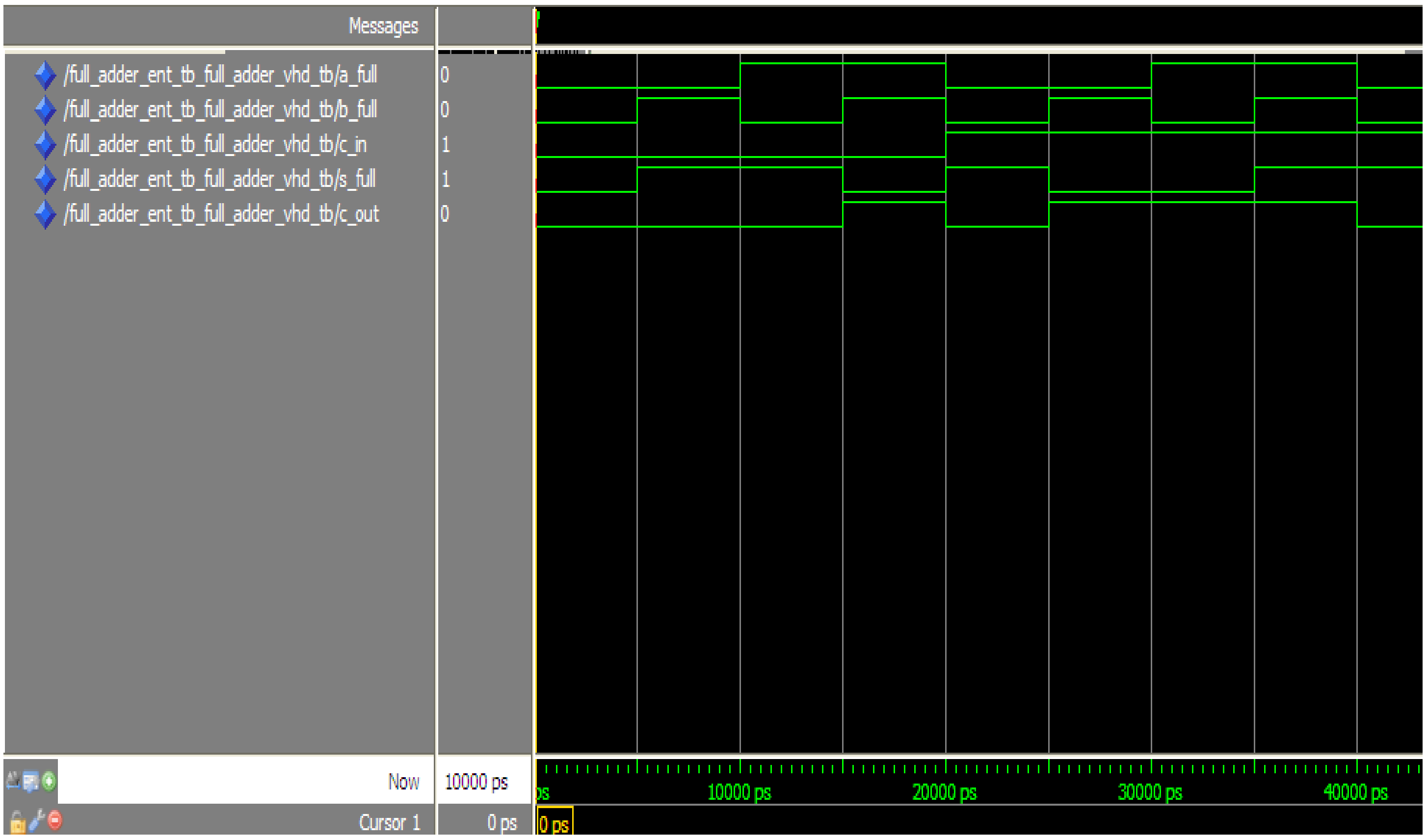
```
================================================================================
tb_Full_Adder.vhd
================================================================================
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
USE ieee.numeric_std.ALL;

--  Hendra Kesuma

ENTITY full_adder_ent_tb_Full_Adder_vhd_tb IS
END full_adder_ent_tb_Full_Adder_vhd_tb;

ARCHITECTURE behavior OF full_adder_ent_tb_Full_Adder_vhd_tb IS

	COMPONENT full_adder_ent
	PORT(
		A_Full : IN std_logic;
		B_Full : IN std_logic;
		C_In : IN std_logic;
		S_Full : OUT std_logic;
		C_Out : OUT std_logic
		);
	END COMPONENT;

	SIGNAL A_Full :  std_logic;
	SIGNAL B_Full :  std_logic;
	SIGNAL C_In :  std_logic;
	SIGNAL S_Full :  std_logic;
	SIGNAL C_Out :  std_logic;

BEGIN

	uut: full_adder_ent PORT MAP(
		A_Full => A_Full,
		B_Full => B_Full,
		C_In => C_In,
		S_Full => S_Full,
		C_Out => C_Out
	);


-- *** Test Bench - User Defined Section ***
```

```
  tb : PROCESS
  BEGIN
  A_Full <= '0' after 0 ns,
          '1' after 10 ns,
                 '0' after 20 ns,
                 '1' after 30 ns,
                 '0' after 40 ns ;

  B_Full <= '0' after 0 ns,
          '1' after 5 ns,
                 '0' after 10 ns,
                 '1' after 15 ns,
                 '0' after 20 ns,
              '1' after 25 ns,
                 '0' after 30 ns,
                 '1' after 35 ns,
                 '0' after 40 ns;

  C_In   <= '0' after 0 ns,
          '1' after 20 ns;

  wait; -- will wait forever
  END PROCESS;
-- *** End Test Bench - User Defined Section ***

END;
```

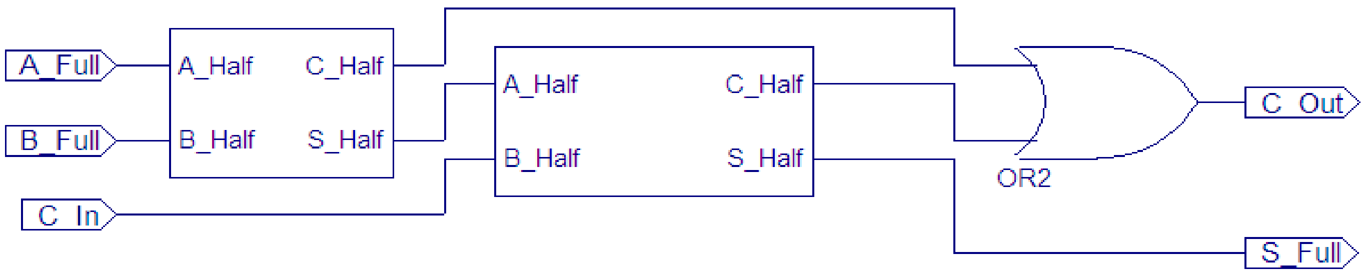
## RTL Schematic



## SIMULATION


Messages
/full_adder_ent_tb_full_adder_vhd_tb/a_full
/full_adder_ent_tb_full_adder_vhd_tb/b_full
/full_adder_ent_tb_full_adder_vhd_tb/c_in
/full_adder_ent_tb_full_adder_vhd_tb/s_full
/full_adder_ent_tb_full_adder_vhd_tb/c_out
0
0
1
1
0
Now
10000 ps
Cursor 1
0 ps
ps
10000 ps
20000 ps
30000 ps
40000 ps
0 ps

## RTL Schematic


A_Full
B_Full
C_In
A_Half
B_Half
C_Half
S_Half
A_Half
B_Half
C_Half
S_Half
OR2
C_Out
S_Full