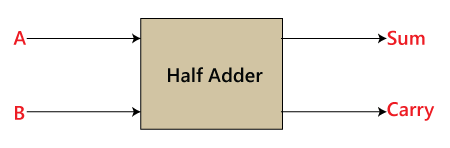
# Half Adder

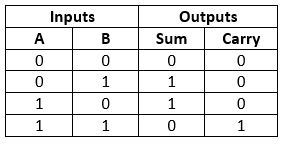
## Theory: -

The Half-Adder is a basic building block of adding two numbers as two inputs and produce out two outputs. The adder is used to perform OR operation of two single bit binary numbers. The **augent** and **addent** bits are two input states, and **'carry**' and **'sum** 'are two output states of the half adder.

## Block diagram



## Truth Table



In the above table,

'A' and' B' are the input states, and 'sum' and 'carry' are the output states.

1. The carry output is 0 in case where both the inputs are not 1.
2. The least significant bit of the sum is defined by the 'sum' bit.

The SOP form of the sum and carry are as follows:

Sum = x'y+xy'  
Carry = xy

## Source Code

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Uncomment the following library declaration if using

-- arithmetic functions with Signed or Unsigned values

--use IEEE.NUMERIC\_STD.ALL;

-- Uncomment the following library declaration if instantiating

-- any Xilinx leaf cells in this code.

--library UNISIM;

--use UNISIM.VComponents.all;

entity ha is

Port ( a : in STD\_LOGIC;

b : in STD\_LOGIC;

carry : out STD\_LOGIC;

sum : out STD\_LOGIC);

end ha;

architecture Behavioral of ha is

begin

sum <= a xor b;

carry <= a and b;

end Behavioral;

## Testbench Code

library IEEE;

use IEEE.Std\_logic\_1164.all;

use IEEE.Numeric\_Std.all;

entity ha\_tb is

end;

architecture bench of ha\_tb is

component ha

Port ( a : in STD\_LOGIC;

b : in STD\_LOGIC;

carry : out STD\_LOGIC;

sum : out STD\_LOGIC);

end component;

signal a: STD\_LOGIC;

signal b: STD\_LOGIC;

signal carry: STD\_LOGIC;

signal sum: STD\_LOGIC;

begin

uut: ha port map ( a => a,

b => b,

carry => carry,

sum => sum );

stimulus: process

begin

-- Put initialisation code here

a <= '0';

b <= '0';

wait for 10ns;

a <= '0';

b <= '1';

wait for 10ns;

a <= '1';

b <= '0';

wait for 10ns;

a <= '1';

b <= '1';

wait for 10ns;

-- Put test bench stimulus code here

wait;

end process;

## Observation

### 

## Output

