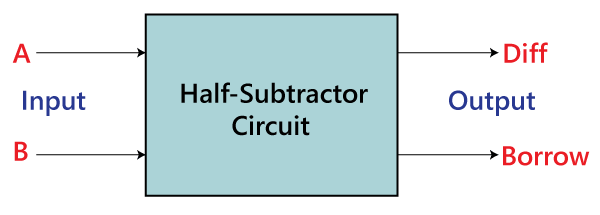
# Half Subtractor

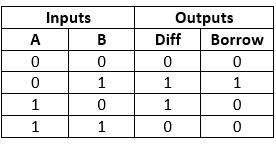
## Theory: -

The half subtractor is also a building block for subtracting two binary numbers. It has two inputs and two outputs. This circuit is used to subtract two single bit binary numbers A and B. The **'diff**' and **'borrow'** are two output states of the half subtractor.

### Block diagram



### Truth Table



The SOP form of the **Diff** and **Borrow** is as follows:

Diff= A'B+AB'  
Borrow = A'B

In the above table,

OOPs Concepts in Java

* 'A' and 'B' are the input variables whose values are going to be subtracted.
* The 'Diff' and 'Borrow' are the variables whose values define the subtraction result, i.e., difference and borrow.
* The first two rows and the last row, the difference is 1, but the 'Borrow' variable is 0.
* The third row is different from the remaining one. When we subtract the bit 1 from the bit 0, the borrow bit is produced.

## 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 hs is

Port ( a : in STD\_LOGIC;

b : in STD\_LOGIC;

diff : out STD\_LOGIC;

borrow : out STD\_LOGIC);

end hs;

architecture Behavioral of hs is

## Testbench Code

library IEEE;

use IEEE.Std\_logic\_1164.all;

use IEEE.Numeric\_Std.all;

entity hs\_tb is

end;

architecture bench of hs\_tb is

component hs

Port ( a : in STD\_LOGIC;

b : in STD\_LOGIC;

diff : out STD\_LOGIC;

borrow : out STD\_LOGIC);

end component;

signal a: STD\_LOGIC;

signal b: STD\_LOGIC;

signal diff: STD\_LOGIC;

signal borrow: STD\_LOGIC;

begin

uut: hs port map ( a => a,

b => b,

diff => diff,

borrow => borrow );

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;

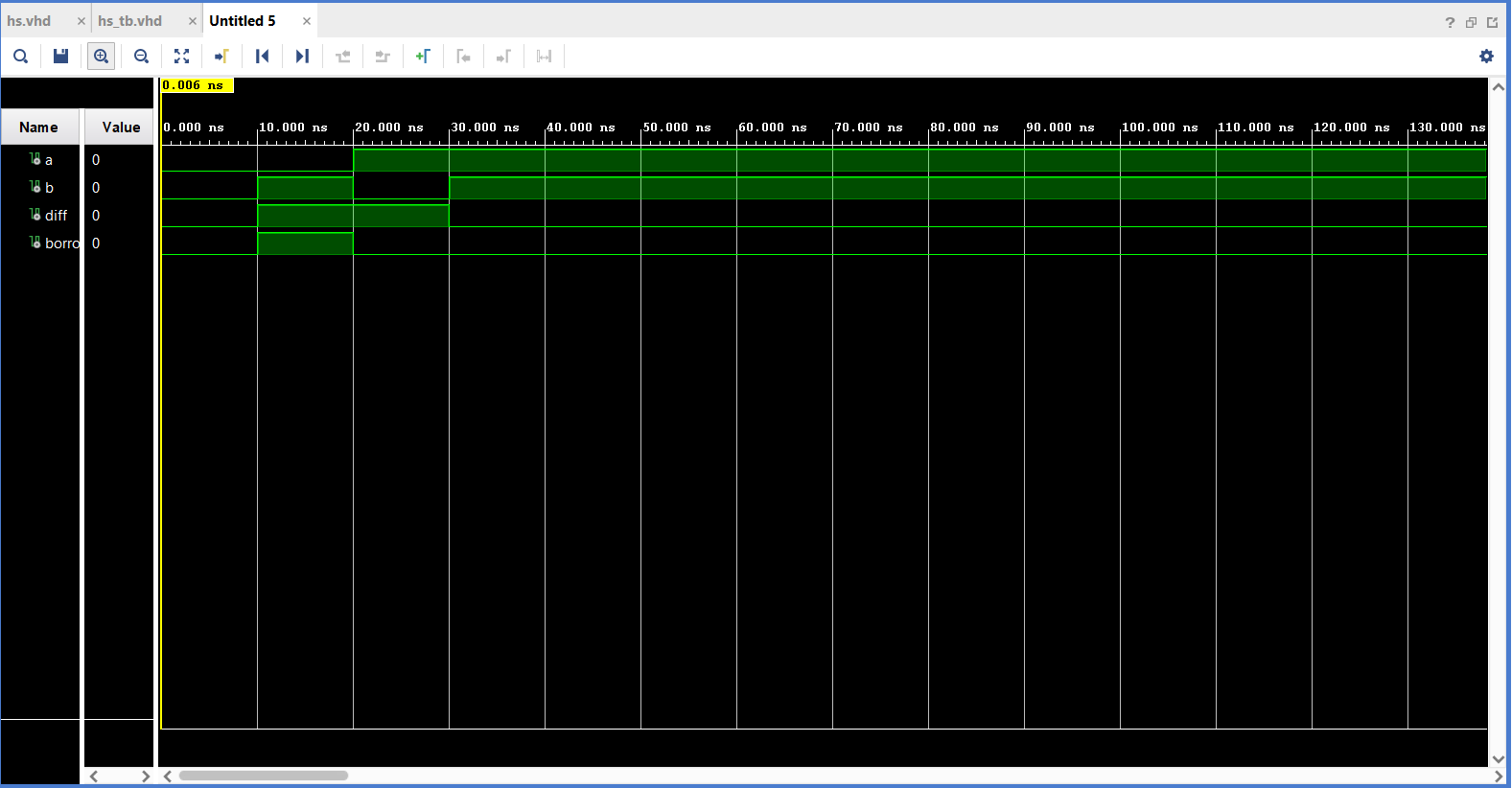
-- Put test bench stimulus code here

wait;

end process;

end;

## Observation



## Output

