Institute of Engineering and Technology DAVV

SESSION 2020-21

BRANCH: E&I (7TH SEM)

SUBJECT: CIRCUIT DESIGN USING HDL

SUBJECT CODE: EIR7E1

LAB ASSIGNMENT

Submitted to:	**Submitted by:**
Dr. Vaibhav Neema	**Shefali Saxena**
	17E7044
	DE17235

Data Flow Modeling in VHDL:

For dataflow modeling in VHDL, we specify the functionality of an entity by defining the flow of information through each gate. We primarily use concurrent signal assignment statements and block statements in dataflow modeling.

In this modeling style, the flow of data through the entity is expressed using concurrent (parallel) signal. The concurrent statements in VHDL are WHEN and GENERATE.

Besides them, assignments using only operators (AND, NOT, +, *, sll, etc.) can also be used to construct code.

Concurrent statements are those statements that are executed in parallel. These are the primary choice for modeling the behavior of an entity in the dataflow style.

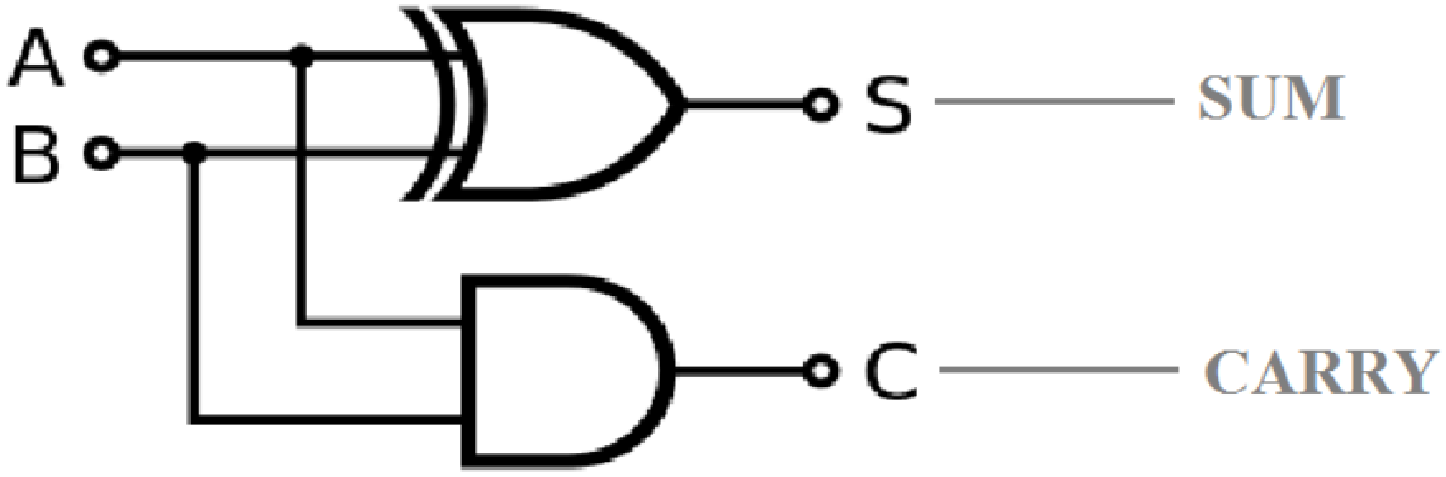
Finally, a special kind of assignment, called BLOCK, can also be employed in this kind of code.

In concurrent code, the following can be used –

- Operators
- The WHEN statement (WHEN/ELSE or WITH/SELECT/WHEN);
- The GENERATE statement;
- The BLOCK statement

Design of 2 bit adder using Data Flow Modelling

Logic diagram of Basic Half adder:

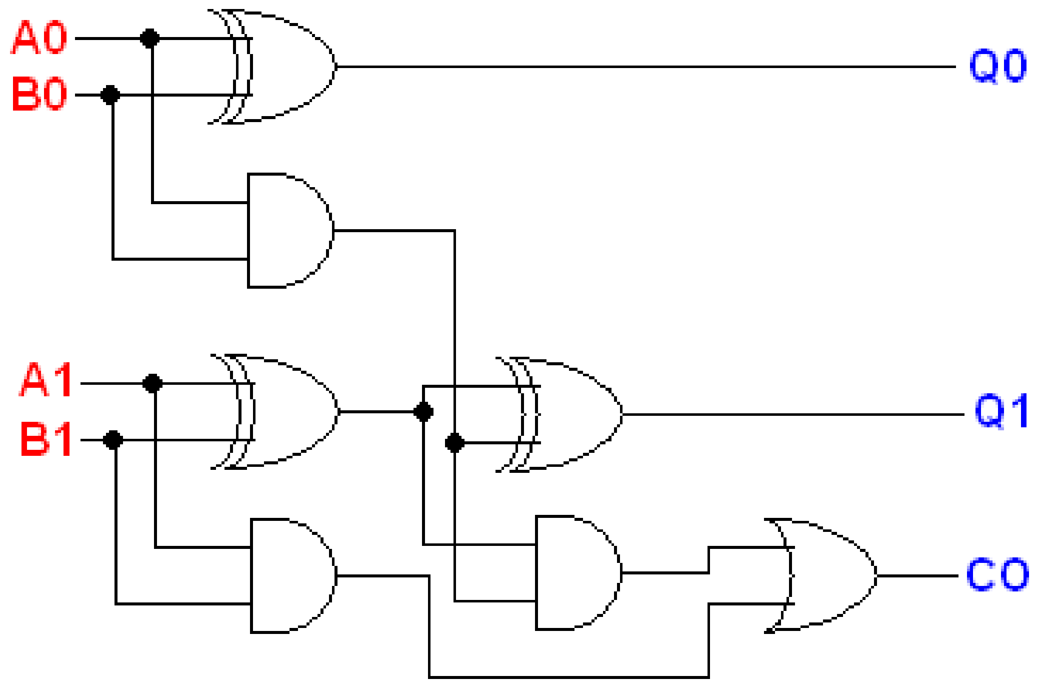


Logic Expression:

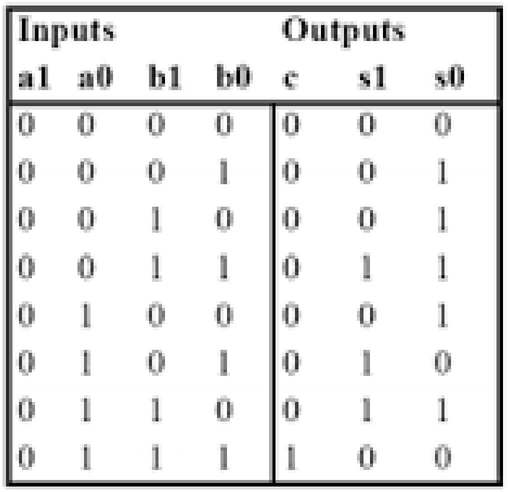
$$S = \overline{A}B + A\overline{B} = A \oplus B$$

$$C = AB$$

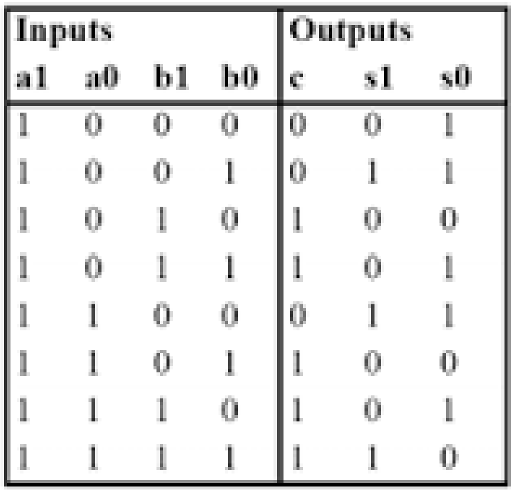
Logic Diagram of 2 bit adder:



Truth Table:



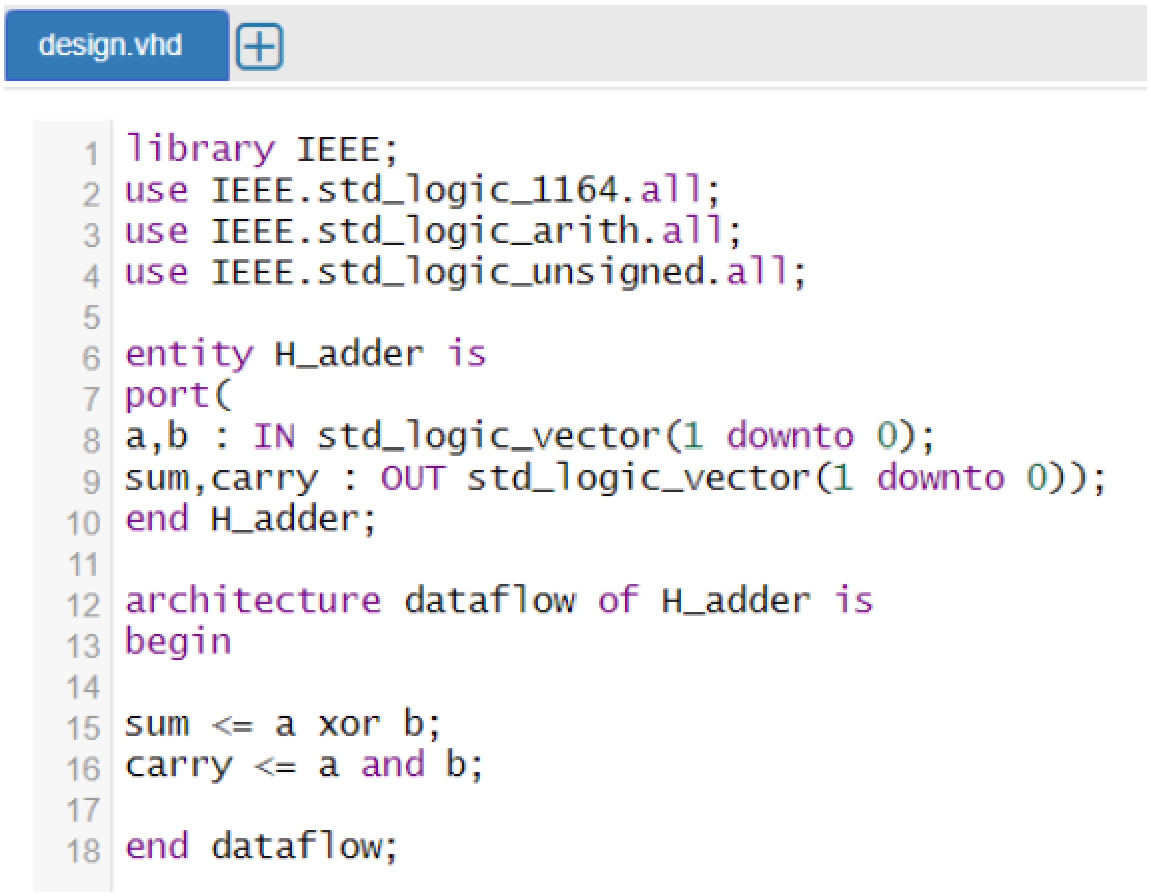
Inputs				Outputs		
a1	a0	b1	b0	c	s1	s0
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	1
0	1	0	0	0	0	1
0	1	0	1	0	1	0
0	1	1	0	0	1	1
0	1	1	1	1	0	0



Inputs				Outputs		
a1	a0	b1	b0	c	s1	s0
1	0	0	0	0	0	1
1	0	0	1	0	1	1
1	0	1	0	1	0	0
1	0	1	1	1	0	1
1	1	0	0	0	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	1
1	1	1	1	1	1	0

Design code:

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

entity H_adder is
port(
a,b : IN std_logic_vector(1 downto 0);
sum,carry : OUT std_logic_vector(1 downto 0));
end H_adder;

architecture dataflow of H_adder is
begin

sum <= a xor b;
carry <= a and b;

end dataflow;
```

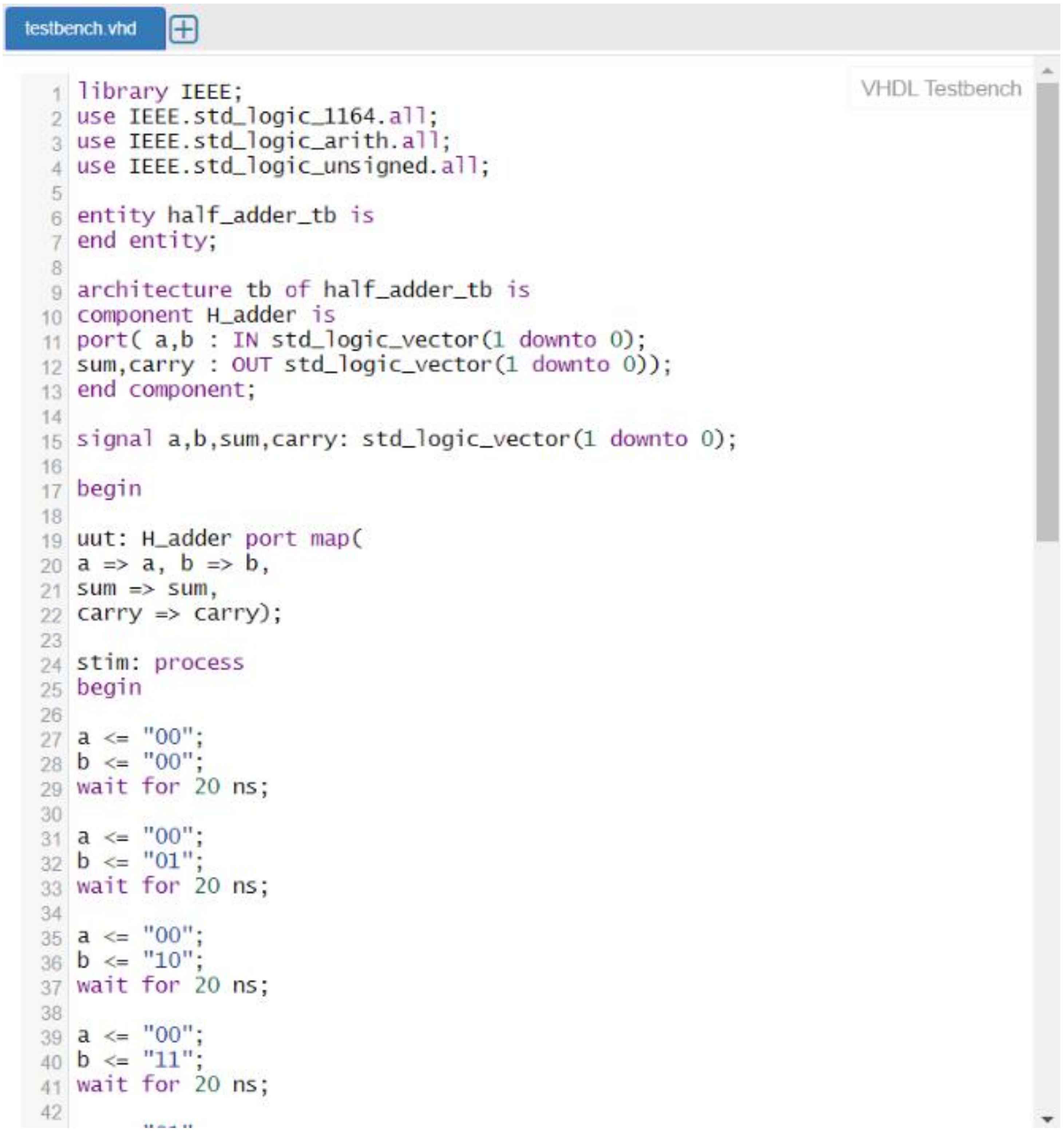

design.vhd

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

entity H_adder is
port(
a,b : IN std_logic_vector(1 downto 0);
sum,carry : OUT std_logic_vector(1 downto 0));
end H_adder;

architecture dataflow of H_adder is
begin

sum <= a xor b;
carry <= a and b;

end dataflow;
```

Testbench Code:

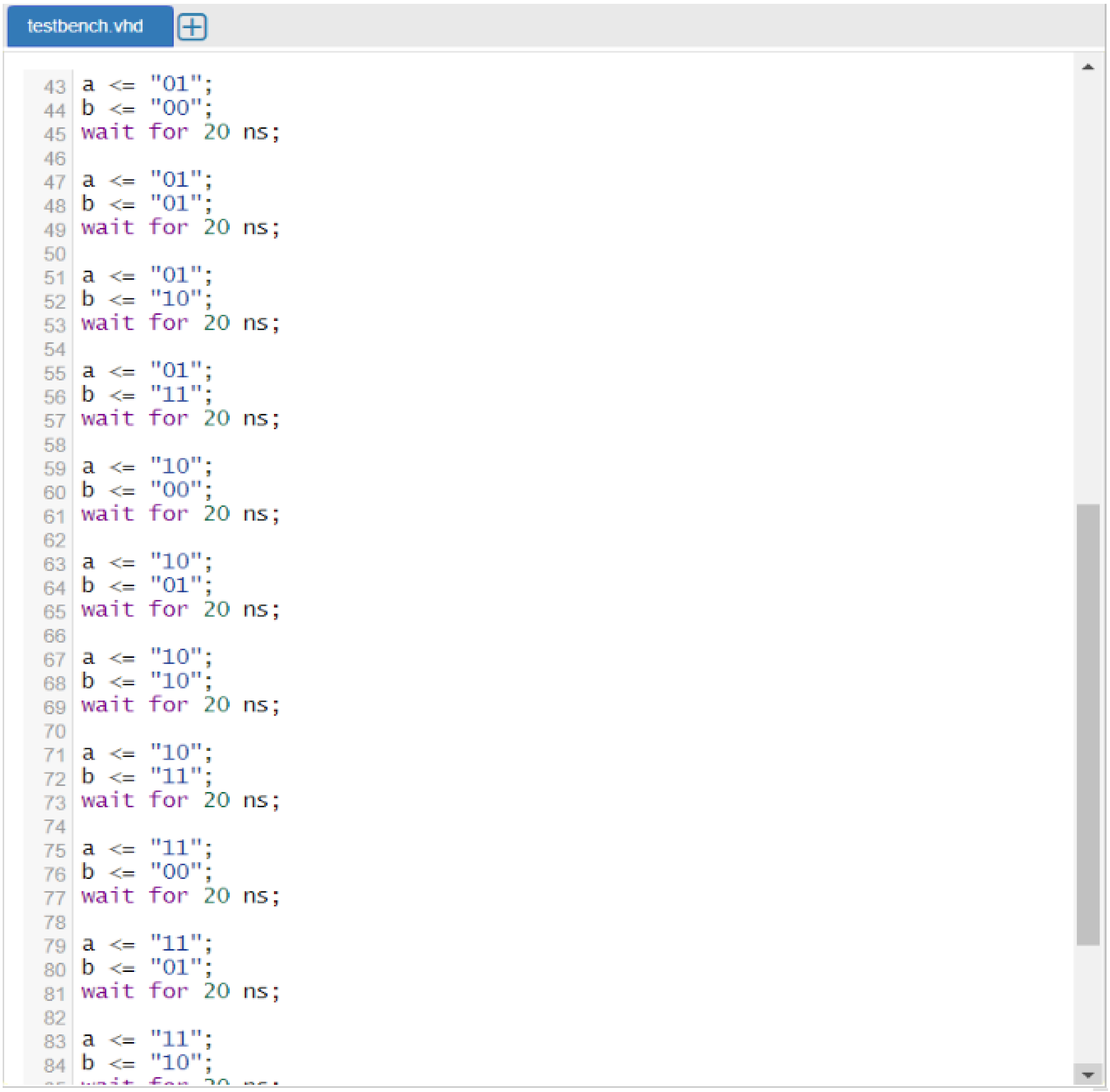
```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;

entity half_adder_tb is
end entity;

architecture tb of half_adder_tb is
component H_adder is
port( a,b : IN std_logic_vector(1 downto 0);
sum,carry : OUT std_logic_vector(1 downto 0));
end component;

signal a,b,sum,carry: std_logic_vector(1 downto 0);

begin

uut: H_adder port map(
a => a, b => b,
sum => sum,
carry => carry);

stim: process
begin

a <= "00";
b <= "00";
wait for 20 ns;

a <= "00";
b <= "01";
wait for 20 ns;

a <= "00";
b <= "10";
wait for 20 ns;

a <= "00";
b <= "11";
wait for 20 ns;

a <= "01";
b <= "00";
wait for 20 ns;

a <= "01";
b <= "01";
wait for 20 ns;

a <= "01";
b <= "10";
wait for 20 ns;

a <= "01";
b <= "11";
wait for 20 ns;

a <= "10";
b <= "00";
wait for 20 ns;

a <= "10";
```

```
b <= "01";
wait for 20 ns;

a <= "10";
b <= "10";
wait for 20 ns;

a <= "10";
b <= "11";
wait for 20 ns;

a <= "11";
b <= "00";
wait for 20 ns;

a <= "11";
b <= "01";
wait for 20 ns;

a <= "11";
b <= "10";
wait for 20 ns;

a <= "11";
b <= "11";
wait for 20 ns;
wait;

end process;

end tb;
```

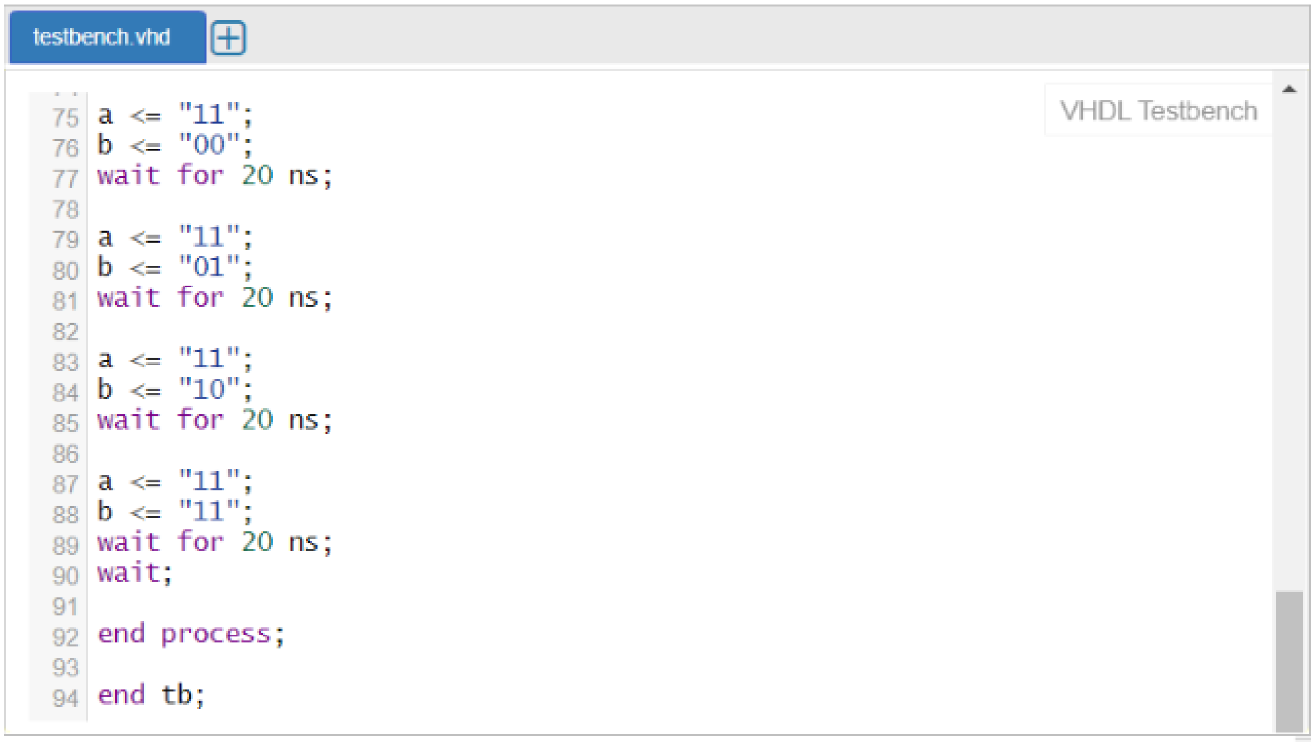

testbench.vhd

VHDL Testbench

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

entity half_adder_tb is
end entity;

architecture tb of half_adder_tb is
component H_adder is
port( a,b : IN std_logic_vector(1 downto 0);
sum,carry : OUT std_logic_vector(1 downto 0));
end component;

signal a,b,sum,carry: std_logic_vector(1 downto 0);

begin

uut: H_adder port map(
a => a, b => b,
sum => sum,
carry => carry);

stim: process
begin

a <= "00";
b <= "00";
wait for 20 ns;

a <= "00";
b <= "01";
wait for 20 ns;

a <= "00";
b <= "10";
wait for 20 ns;

a <= "00";
b <= "11";
wait for 20 ns;

```

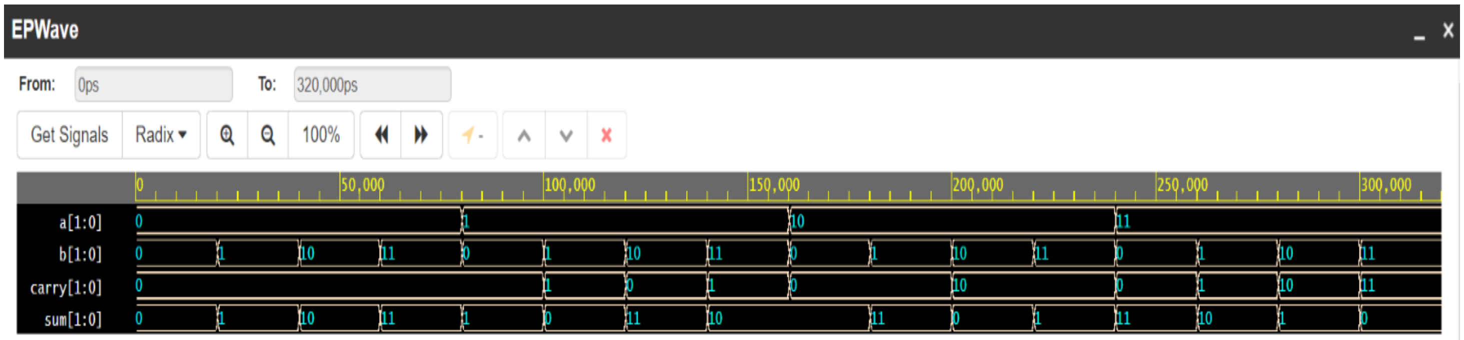


```
a <= "01";
b <= "00";
wait for 20 ns;

a <= "01";
b <= "01";
wait for 20 ns;

a <= "01";
b <= "10";
wait for 20 ns;

a <= "01";
b <= "11";
wait for 20 ns;

a <= "10";
b <= "00";
wait for 20 ns;

a <= "10";
b <= "01";
wait for 20 ns;

a <= "10";
b <= "10";
wait for 20 ns;

a <= "10";
b <= "11";
wait for 20 ns;

a <= "11";
b <= "00";
wait for 20 ns;

a <= "11";
b <= "01";
wait for 20 ns;

a <= "11";
b <= "10";
```



```vhdl
a <= "11";
b <= "00";
wait for 20 ns;

a <= "11";
b <= "01";
wait for 20 ns;

a <= "11";
b <= "10";
wait for 20 ns;

a <= "11";
b <= "11";
wait for 20 ns;
wait;

end process;

end tb;
```

Waveform:



Note: To revert to EPWave opening in a new browser window, set that option on your user page.

https://www.edaplayground.com/x/jHmQ