Half and Full Adder written in VHDL

Mahdi Haghverdi

October 30, 2023

Contents

1 Half Adder Source Code

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity half_adder is
5  port (
6      i, ii : in std_logic;
7      sum, carry : out std_logic
8  );
9  end half_adder;
10
11 architecture arch of half_adder is
12 begin
13     sum <= i xor ii;
14     carry <= i and ii;
```

```
15 end arch;
```

2 Full Adder Source Code

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity full_adder is
5      port (
6          a, b, ci : in std_logic;
7          s, c : out std_logic
8      );
9  end full_adder;
10
11
12 architecture arch of full_adder is
13 component half_adder
14     port (
15         i, ii : in std_logic;
16         sum, carry : out std_logic
17     );
18 end component;
19
20 for half_adder_0: half_adder use entity work.half_adder;
21 for half_adder_1: half_adder use entity work.half_adder;
22 signal asb, aab, asbco: std_logic;
23
24 begin
25 half_adder_0: half_adder port map(
26     -- entity-signal-name => local-signal-name
27     i => a,
28     ii => b,
29     sum => asb,
30     carry => aab
31 );
32
33 half_adder_1: half_adder port map(
34     i => asb,
35     ii => ci,
36     sum => s,
37     carry => asbco
38 );
39
40 c <= aab or asbco;
41 end arch;
```

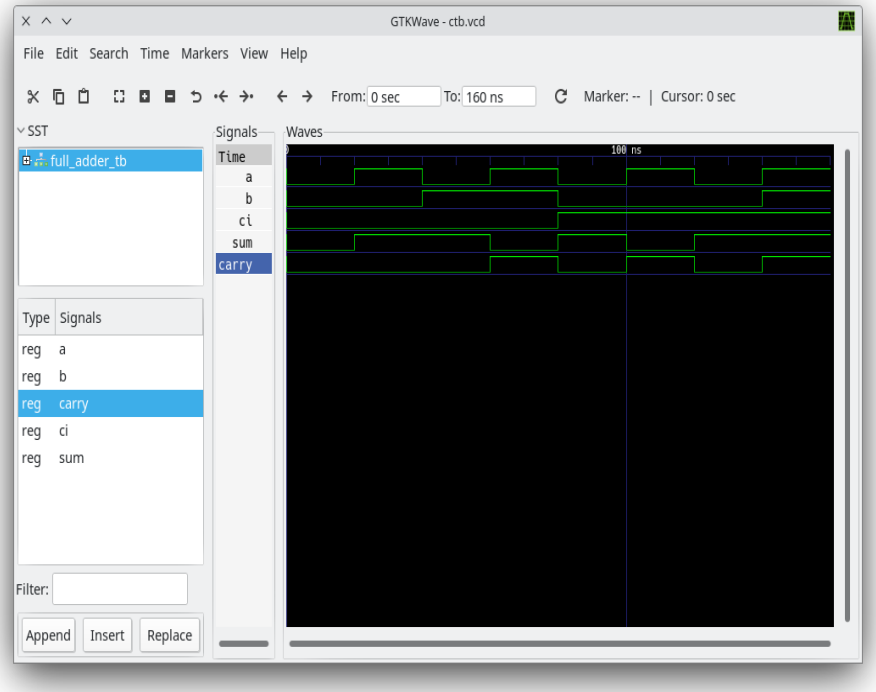
3 Concurrent Test Bench Source Code

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity full_adder_tb is
5  end full_adder_tb;
6
7  architecture tb of full_adder_tb is
8      signal a, b, ci : std_logic; -- inputs
9      signal sum, carry : std_logic; -- outputs
10 begin
11 -- connecting testbench signals with full_adder.vhdl
12 UUT : entity work.full_adder port map
13       (a => a, b => b, ci => ci, s => sum, c => carry);
14
15     -- inputs
16     -- ci ba
17     --  0 00 at  0 ns
18     --  0 01 at 20 ns
19     --  0 10 at 40 ns
20     --  0 11 at 60 ns
21     --  1 00 at 80 ns
22     --  1 01 at 100 ns
23     --  1 10 at 120 ns
24     --  1 11 at 140 ns
25     --  1 11 at 160 ns
26
27     a <= '0',
28          '1' after 20 ns,
29          '0' after 40 ns,
30          '1' after 60 ns,
31          '0' after 80 ns,
32          '1' after 100 ns,
33          '0' after 120 ns,
34          '1' after 140 ns,
35          '1' after 160 ns;
36     b <= '0',
37          '1' after 40 ns,
38          '0' after 80 ns,
39          '0' after 120 ns,
40          '1' after 140 ns,
41          '1' after 160 ns;
```

```
42      ci <= '0',
43            '1' after 80 ns,
44            '1' after 120 ns,
45            '1' after 160 ns;
46  end tb ;
```

3.1 Output Signals

3.1.1 GTKWave Output



3.2 Explanation

In the `UUT` entity of the test bench, we first use the full adder that we wrote, and then concurrently, we pass the commented truth table to the inputs of the full adder, the desired output should exists on the GTKWave output, in which it is.

4 Process Based Test Bench Source Code

```
1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  entity full_adder_process_tb is
5  end full_adder_process_tb;
6
7  architecture tb of full_adder_process_tb is
8      signal a, b, ci : std_logic; -- inputs
9      signal sum, carry : std_logic; -- outputs
10 begin
11 -- connecting testbench signals with full_adder.vhdl
12     UUT : entity work.full_adder port map (a => a, b => b, ci
           => ci, s => sum, c => carry);
13     tb1 : process
14     constant period: time := 20 ns;
15         begin
16             -- a b s c
17             -- 0 0 0 0
18             -- 0 1 1 0
19             -- 1 0 1 0
20             -- 1 1 0 1
21
22             a <= '0';
23             b <= '0';
24             ci <= '0';
25             wait for period;
26             assert ((sum = '0') and (carry = '0')) -- expected
                   output
27             -- error will be reported if sum or carry is not 0
28             report "test failed for input combination" severity
                   error;
29
30             a <= '0';
31             b <= '1';
32             ci <= '0';
33             wait for period;
34             assert ((sum = '1') and (carry = '0'))
35             report "test failed for input combination 001"
                   severity error;
36
37             a <= '1';
```

```
38          b <= '0';
39          ci <= '0';
40          wait for period;
41          assert ((sum = '1') and (carry = '0'))
42          report "test failed for input combination 010"
                severity error;
43
44          a <= '1';
45          b <= '1';
46          ci <= '0';
47          wait for period;
48          assert ((sum = '0') and (carry = '1'))
49          report "test failed for input combination 011"
                severity error;
50
51
52          a <= '0';
53          b <= '0';
54          ci <= '1';
55          wait for period;
56          assert ((sum = '1') and (carry = '0')) -- expected
                output
57          -- error will be reported if sum or carry is not 0
58          report "test failed for input combination 100"
                severity error;
59
60          a <= '0';
61          b <= '1';
62          ci <= '1';
63          wait for period;
64          assert ((sum = '0') and (carry = '1'))
65          report "test failed for input combination 101"
                severity error;
66
67          a <= '1';
68          b <= '0';
69          ci <= '1';
70          wait for period;
71          assert ((sum = '0') and (carry = '1'))
72          report "test failed for input combination 110"
                severity error;
73
74          a <= '1';
75          b <= '1';
```

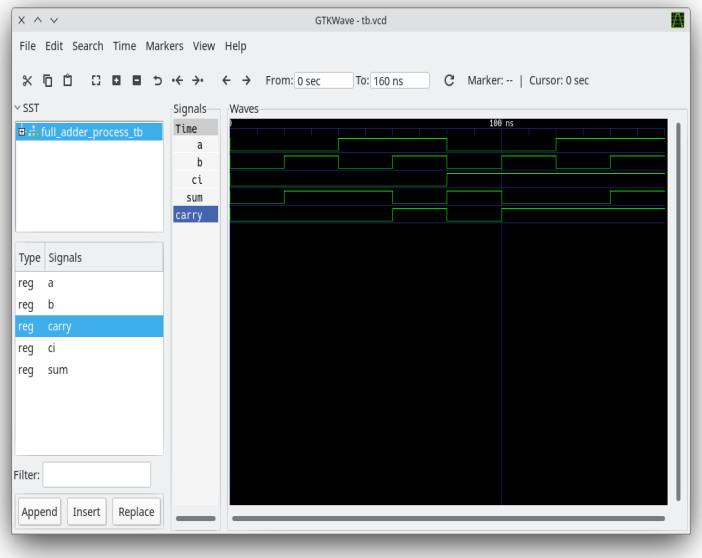
```
76              ci <= '1';
77              wait for period;
78              assert ((sum = '1') and (carry = '1'))
79              report "test failed for input combination 111"
                    severity error;
80
81
82              assert false report "all tests passed." severity
                    note;
83              wait; -- indefinitely suspend process
84      end process;
85  end tb ;
```

4.1 Output Signals

4.1.1 Compilation Output

```
❯ ghdl -r full_adder_process_tb --vcd=tb.vcd
test_bench.vhdl:83:13:@160ns:(assertion note): all tests passed.
```

4.1.2 GTKWave Output



4.2 Explanation

Process-based testing is another approach of writing VHDL test bench.

After instantiating the full adder that we wrote, we create a process named `tb1`. With the period of `20 ns` which is defined as `constant`, we pass eight diffrent inputs to the full adder.

Then, with the `assert` statement, we test the output to be desired and corrent output. if it is not, we report a detailed message for the *test case*. If all the test cases pass, we report `all tests passed.`.

This kind of testing is a little bit harder to write but has the advantage of reporting error message for the written test cases.