

Course Name:	Hardware Description Language Lab (2UXL401)	Semester:	IV
Date of Performance:	12/01/2021	Batch No:	B2
Faculty Name:	Prof. Bhargavi Kaslikar	Roll No:	1912060
Faculty Sign & Date:		Grade/Marks:	

Experiment No: 1

Title: Study of basic VHDL code: Adder (Dataflow)

Aim and Objective of the Experiment:

Write a VHDL code

- To implement a half adder.
- A full adder using half adder
- A 4-bit adder using full adder.

Write a testbench to verify your results for half adder and four bit adder Implement the full adder on CPLD.

To study basic structure of VHDL code and to understand use of test bench for simulation.
To know the process for implementation on CPLD.

COs to be achieved:

CO 1: Use basic Concurrent and Sequential statements in VHDL and write codes for simple applications

CO 2: Test a VHDL code and verify the circuit model.

CO 3: Synthesize and Implement the designed circuits on CPLD/ FPGA.

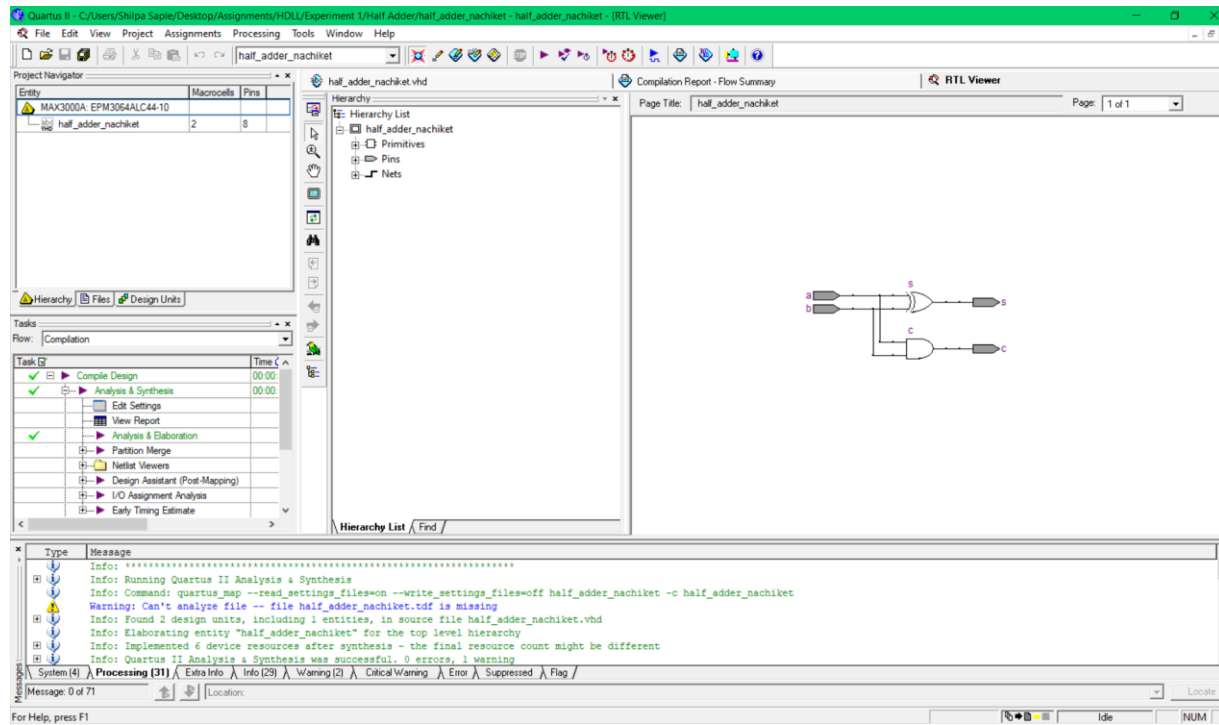
Work to be done

Upload VHDL codes for half adder, full adder (structural) and four-bit adder (structural) and test bench for half adder and 4-bit adder.

Also Upload Simulation waveforms for 4 bit adder.

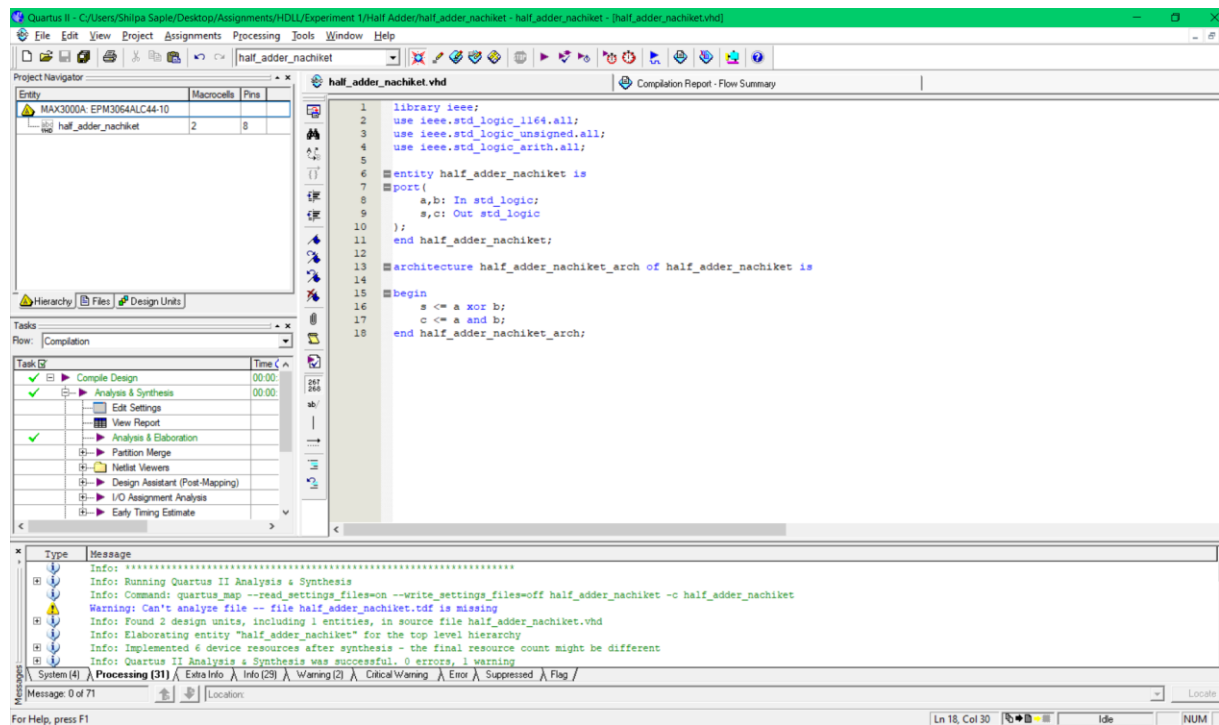
Upload scanned image for post lab questions

Half Adder:



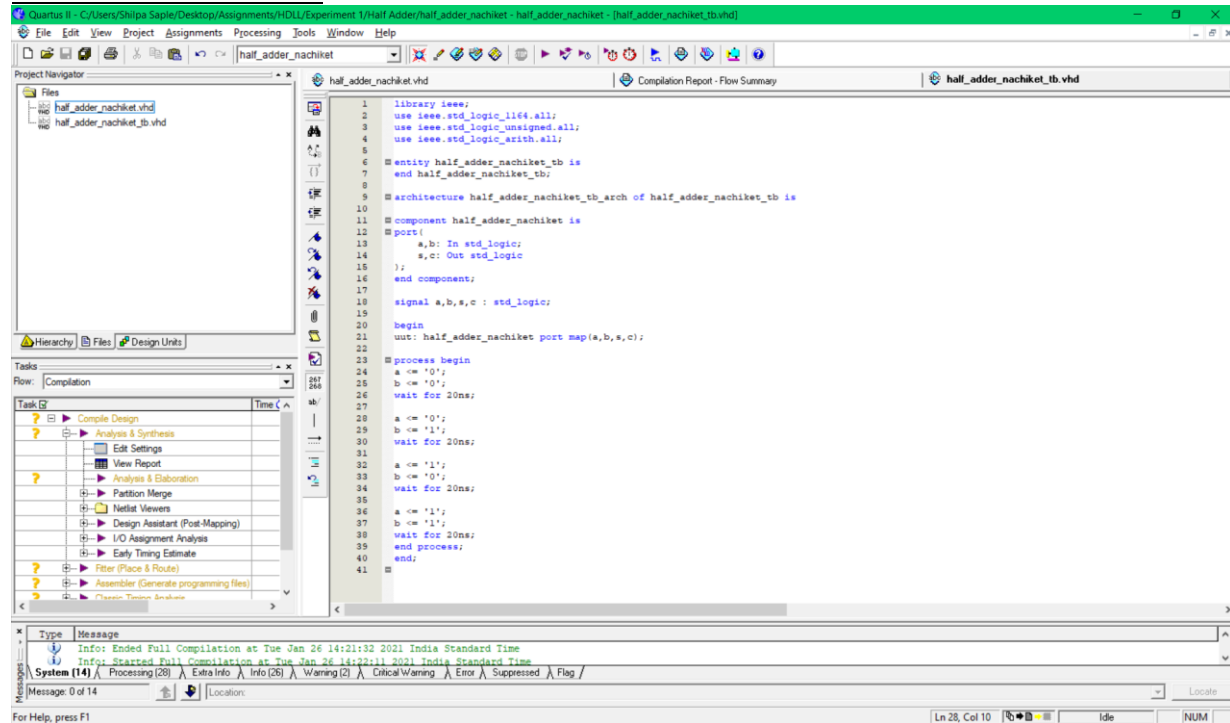
The screenshot shows the Quartus II RTL Viewer for the 'half_adder_nachiket' project. The top pane displays the RTL diagram of a half adder, which consists of two 3-input AND gates and one 3-input OR gate. The inputs are labeled 'a' and 'b', and the outputs are labeled 's' (sum) and 'c' (carry). The bottom pane shows the compilation messages, including the command 'quartus_map --read_settings_files=on --write_settings_files=off half_adder_nachiket -c half_adder_nachiket' and the warning 'Warning: Can't analyze file -- file half_adder_nachiket.tdf is missing'. The messages also indicate that the synthesis was successful with 0 errors and 1 warning.

Entity – half-adder:



The screenshot shows the Quartus II IDE with the VHDL code for the 'half_adder_nachiket' entity. The code defines the entity with two inputs 'a' and 'b' and two outputs 's' and 'c'. The architecture 'half_adder_nachiket_arch' implements the logic using XOR and AND gates. The compilation messages at the bottom show the command 'quartus_map --read_settings_files=on --write_settings_files=off half_adder_nachiket -c half_adder_nachiket' and the warning 'Warning: Can't analyze file -- file half_adder_nachiket.tdf is missing'. The messages also indicate that the synthesis was successful with 0 errors and 1 warning.

Testbench – Half-adder:



```

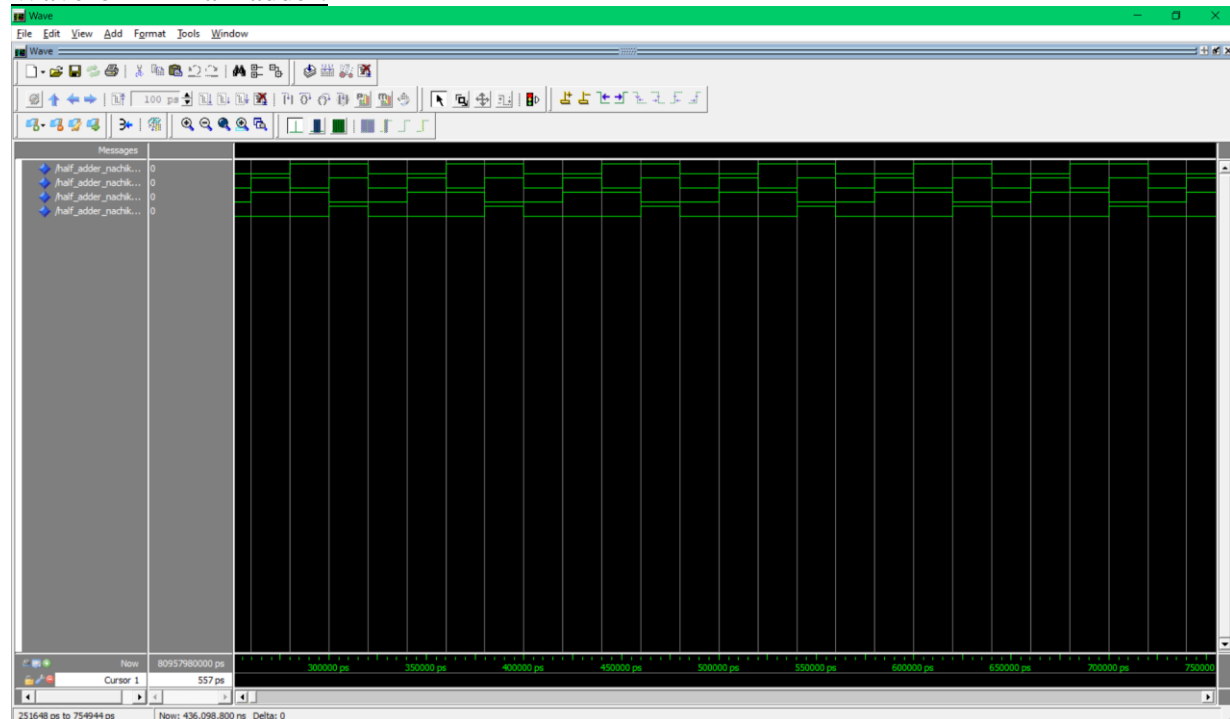
1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 use ieee.std_logic_arith.all;
5
6 entity half_adder_nachiket_tb is
7 end half_adder_nachiket_tb;
8
9 architecture half_adder_nachiket_tb_arch of half_adder_nachiket_tb is
10
11 component half_adder_nachiket is
12 port(
13     a,b: In std_logic;
14     s,c: Out std_logic;
15 );
16 end component;
17
18 signal a,b,s,c : std_logic;
19
20 begin
21     uut: half_adder_nachiket port map(a,b,s,c);
22
23 process begin
24     a <= '0';
25     b <= '0';
26     wait for 20ns;
27
28     a <= '0';
29     b <= '1';
30     wait for 20ns;
31
32     a <= '1';
33     b <= '0';
34     wait for 20ns;
35
36     a <= '1';
37     b <= '1';
38     wait for 20ns;
39 end process;
40 end;
41

```

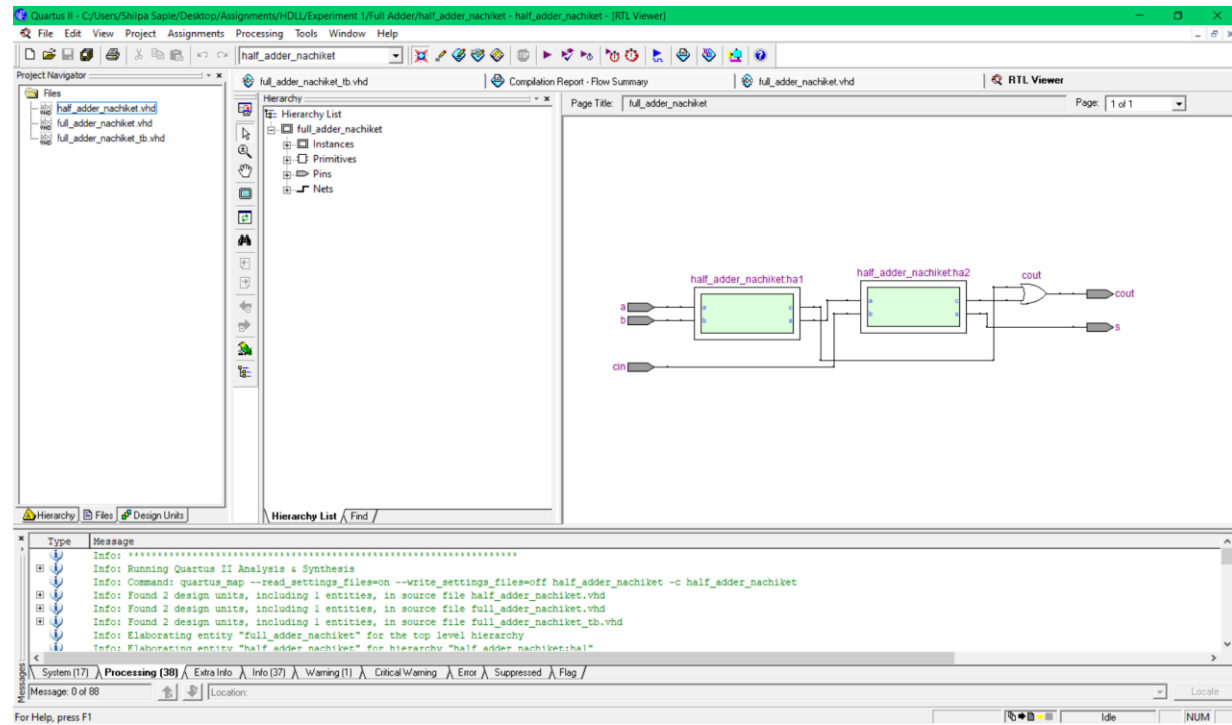
Messages:

- Info: Ended Full Compilation at Tue Jan 26 14:21:32 2021 India Standard Time
- Info: Started Full Compilation at Tue Jan 26 14:22:11 2021 India Standard Time
- System [14] Processing [28] Info [26] Warning [2] CriticalWarning [0] Error [0] Suppressed [0] Flag [0]

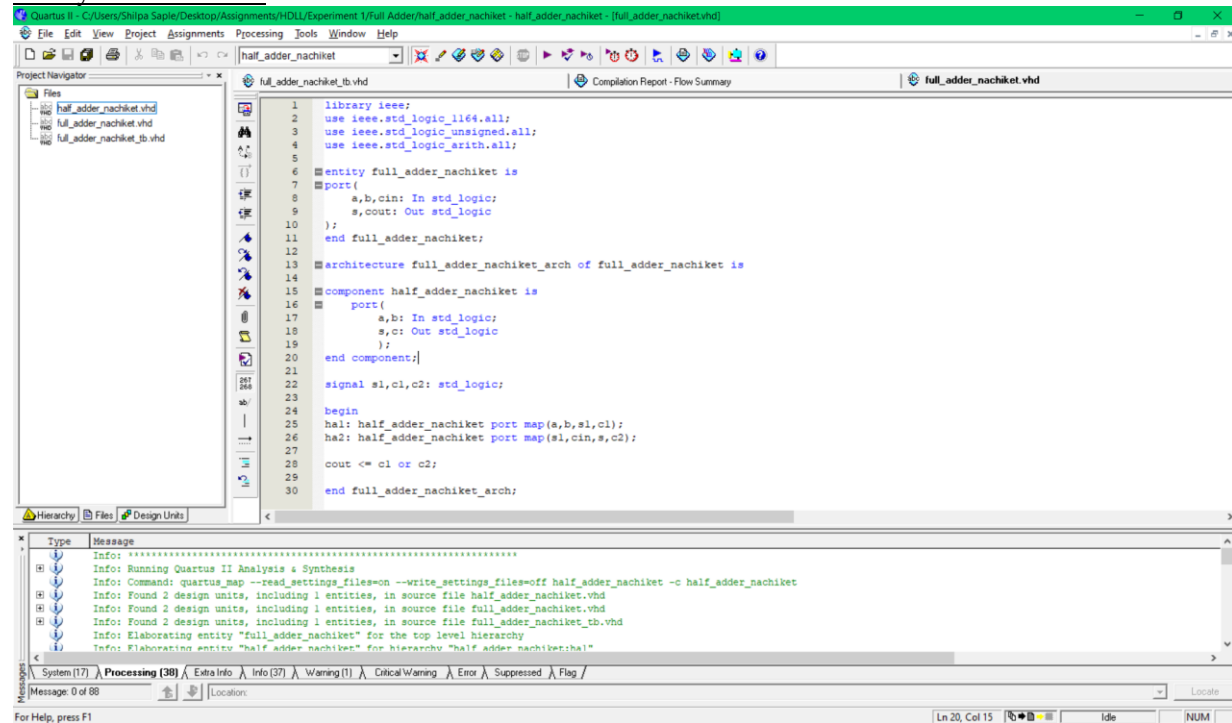
Waveform – Half-adder:



Full Adder:



Entity – Full-adder:



```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 use ieee.std_logic_arith.all;
5
6 entity full_adder_nachiket is
7 port(
8   a,b,cin: In std_logic;
9   s,cout: Out std_logic
10 );
11 end full_adder_nachiket;
12
13 architecture full_adder_nachiket_arch of full_adder_nachiket is
14
15   component half_adder_nachiket is
16     port(
17       a,b: In std_logic;
18       s,c: Out std_logic
19     );
20   end component;
21
22   signal s1,c1,c2: std_logic;
23
24 begin
25   ha1: half_adder_nachiket port map(a,b,s1,c1);
26   ha2: half_adder_nachiket port map(s1,cin,s,c2);
27
28   cout <= c1 or c2;
29
30 end full_adder_nachiket_arch;
  
```

The screenshot shows the Quartus II interface with the VHDL code editor open. The code defines the entity 'full_adder_nachiket' with inputs 'a', 'b', and 'cin', and outputs 's' and 'cout'. It uses two half-adder components to implement the full adder logic. The message window at the bottom shows the synthesis process results, indicating that the design was successfully synthesized.

Testbench – Full-adder:

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5
6  entity full_adder_nachiket_tb is
7  end full_adder_nachiket_tb;
8
9  architecture full_adder_nachiket_tb_arch of full_adder_nachiket_tb is
10
11  component full_adder_nachiket is
12      port(
13          a,b,cin: In std_logic;
14          s,out: Out std_logic
15      );
16  end component;
17
18  signal a,b,cin,s,out: std_logic;
19
20  begin
21      uut: full_adder_nachiket port map(a,b,cin,s,out);
22
23  process begin
24      a <= '0';
25      b <= '0';
26      cin <= '0';
27      wait for 20ns;
28      a <= '0';
29      b <= '0';
30      cin <= '1';
31      wait for 20ns;
32      a <= '0';
33      b <= '1';
34      cin <= '0';
35      wait for 20ns;
36      a <= '1';
37      b <= '1';
38      cin <= '0';
39      wait for 20ns;
40
41  end process;
42
43 end;

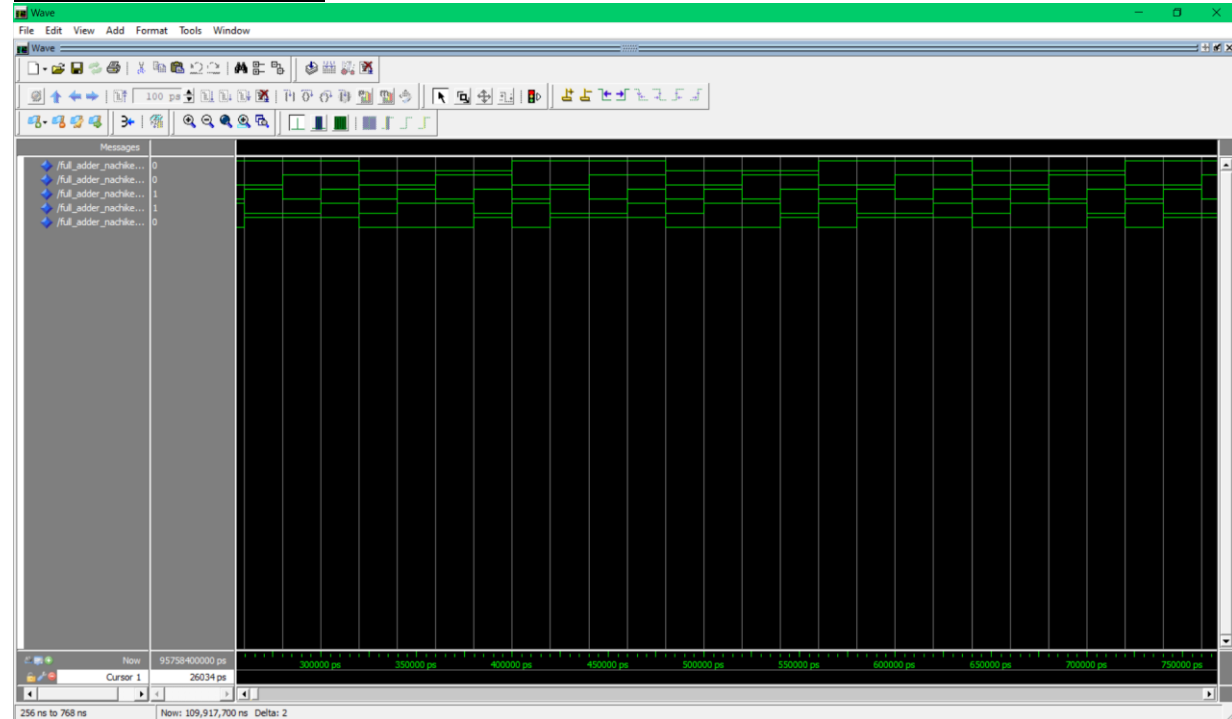
```

```

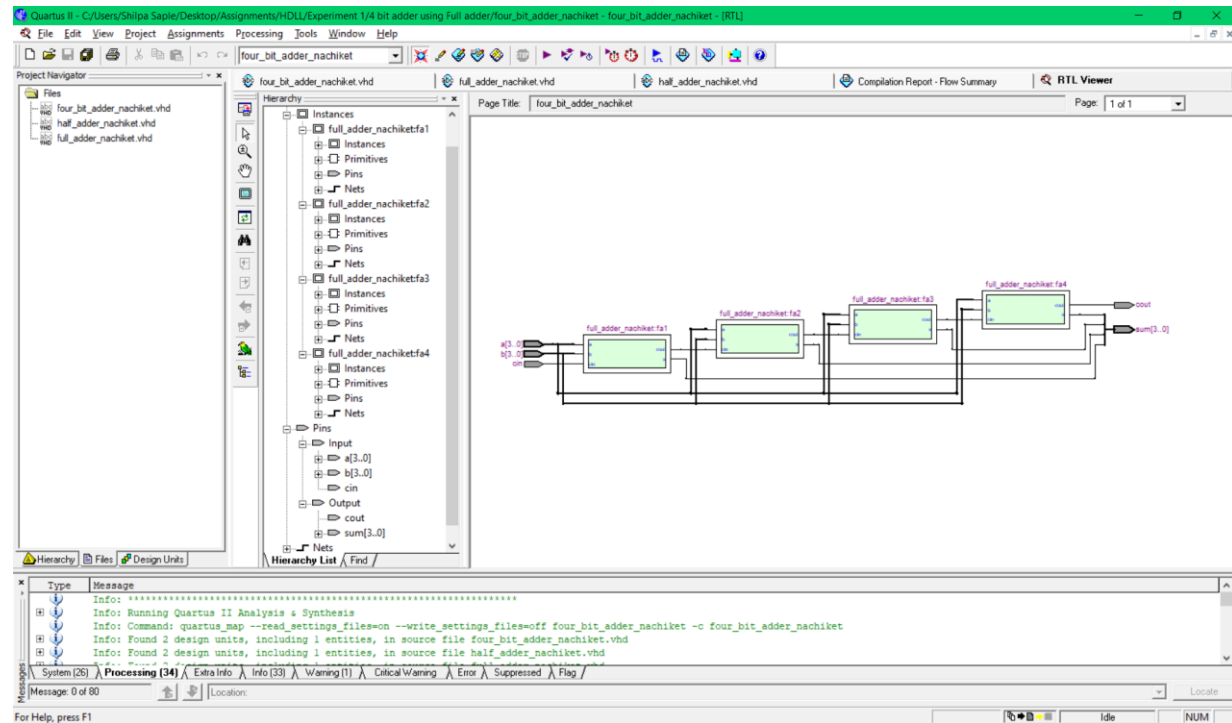
32  cin <= '1';
33  wait for 20ns;
34  a <= '0';
35  b <= '1';
36  cin <= '0';
37  wait for 20ns;
38  a <= '0';
39  b <= '1';
40  cin <= '1';
41  wait for 20ns;
42  a <= '1';
43  b <= '0';
44  cin <= '0';
45  wait for 20ns;
46  a <= '1';
47  b <= '1';
48  cin <= '1';
49  wait for 20ns;
50  a <= '1';
51  b <= '1';
52  cin <= '0';
53  wait for 20ns;
54  a <= '1';
55  b <= '1';
56  cin <= '1';
57  wait for 20ns;
58  a <= '1';
59  b <= '1';
60  cin <= '1';
61  wait for 20ns;
62
63 end process;
64
65 end;

```

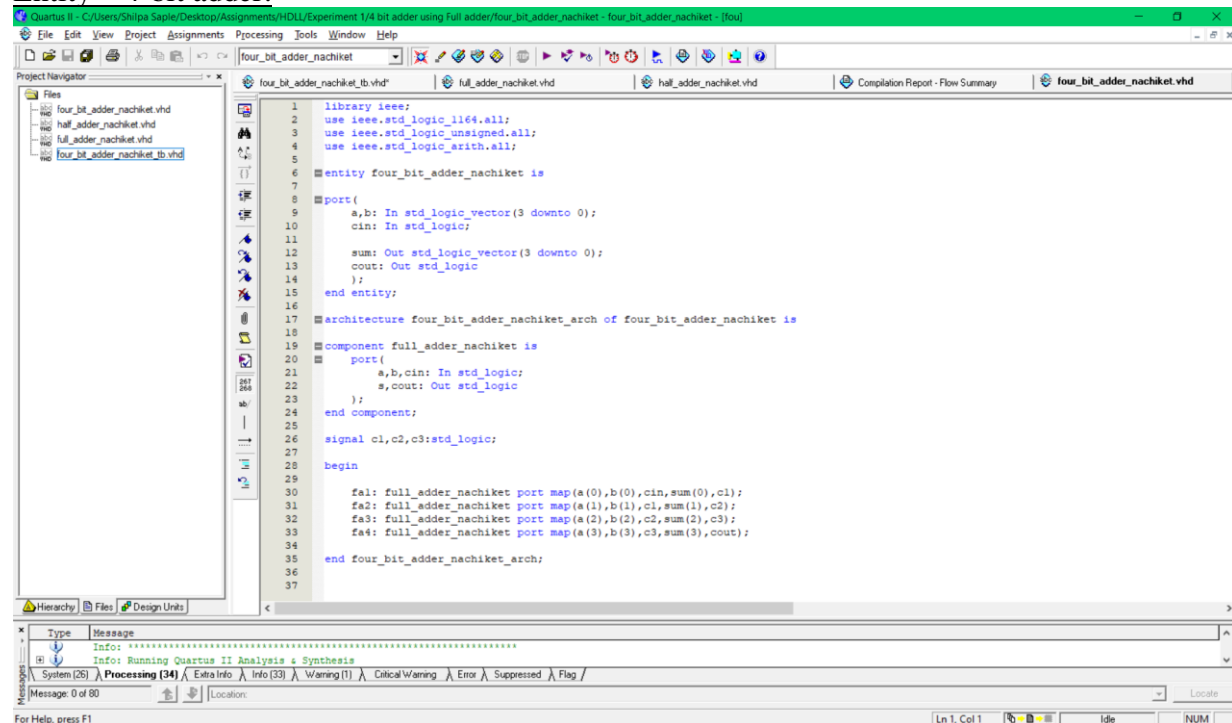
Waveform – Full-adder:



4-bit Adder using Full-adder:



Entity – 4-bit adder:



```

1 library ieee;
2 use ieee.std_logic_1164.all;
3 use ieee.std_logic_unsigned.all;
4 use ieee.std_logic_arith.all;
5
6 entity four_bit_adder_nachiket is
7
8 port(
9     a,b: In std_logic_vector(3 downto 0);
10    cin: In std_logic;
11
12    sum: Out std_logic_vector(3 downto 0);
13    cout: Out std_logic
14 );
15 end entity;
16
17 architecture four_bit_adder_nachiket_arch of four_bit_adder_nachiket is
18
19     component full_adder_nachiket is
20     port(
21         a,b,cin: In std_logic;
22         s,cout: Out std_logic
23     );
24     end component;
25
26     signal c1,c2,c3:std_logic;
27
28     begin
29
30         fa1: full_adder_nachiket port map (a(0),b(0),cin,sum(0),c1);
31         fa2: full_adder_nachiket port map (a(1),b(1),c1,sum(1),c2);
32         fa3: full_adder_nachiket port map (a(2),b(2),c2,sum(2),c3);
33         fa4: full_adder_nachiket port map (a(3),b(3),c3,sum(3),cout);
34
35     end four_bit_adder_nachiket_arch;
36
37

```

The screenshot shows the Quartus II IDE with the VHDL code for a 4-bit adder entity. The code defines the entity and its architecture, using four full-adder blocks to implement the 4-bit adder. The message window at the bottom shows the synthesis results, indicating that the design was successfully synthesized.

Testbench – 4-bit adder:

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3  use ieee.std_logic_unsigned.all;
4  use ieee.std_logic_arith.all;
5
6  entity four_bit_adder_nachiket_tb is
7  end four_bit_adder_nachiket_tb;
8
9  architecture four_bit_adder_nachiket_tb_arch of four_bit_adder_nachiket_tb is
10
11  component four_bit_adder_nachiket is
12  port(
13    a,b: In std_logic_vector(3 downto 0);
14    cin: In std_logic;
15
16    sum: Out std_logic_vector(3 downto 0);
17    cout: Out std_logic;
18  );
19  end component;
20
21  signal a,b,sum:std_logic_vector(3 downto 0);
22  signal cin,cout:std_logic;
23
24  begin
25
26  uut:four_bit_adder_nachiket port map(a,b,cin,sum,cout);
27
28  process begin
29
30    a <= "0010";
31    b <= "0100";
32    cin <= '1';
33    wait for 100ns;
34
35    a <= "1100";
36    b <= "0101";
37    cin <= '1';
38    wait for 100ns;
39
40    a <= "1111";

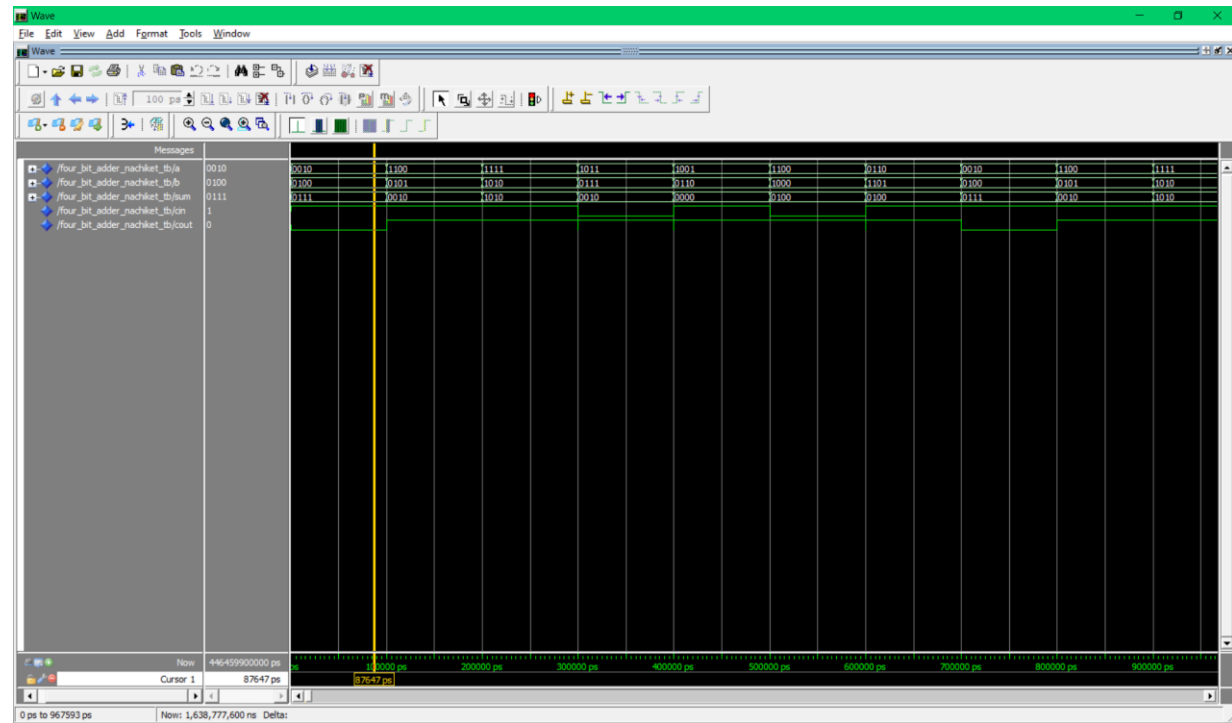
```

```

29
30    a <= "0010";
31    b <= "0100";
32    cin <= '1';
33    wait for 100ns;
34
35    a <= "1100";
36    b <= "0101";
37    cin <= '1';
38    wait for 100ns;
39
40    a <= "1111";
41    b <= "1010";
42    cin <= '1';
43    wait for 100ns;
44
45    a <= "1011";
46    b <= "0111";
47    cin <= '0';
48    wait for 100ns;
49
50    a <= "1001";
51    b <= "0110";
52    cin <= '1';
53    wait for 100ns;
54
55    a <= "1100";
56    b <= "1000";
57    cin <= '0';
58    wait for 100ns;
59
60    a <= "0110";
61    b <= "1101";
62    cin <= '1';
63    wait for 100ns;
64
65  end process;
66
67 end;

```


Waveform – 4-bit adder:



Post Lab Subjective/Objective type Questions:

Upload Answer of following question before coming to next laboratory.

Q1. Analyse the following code and write its output.

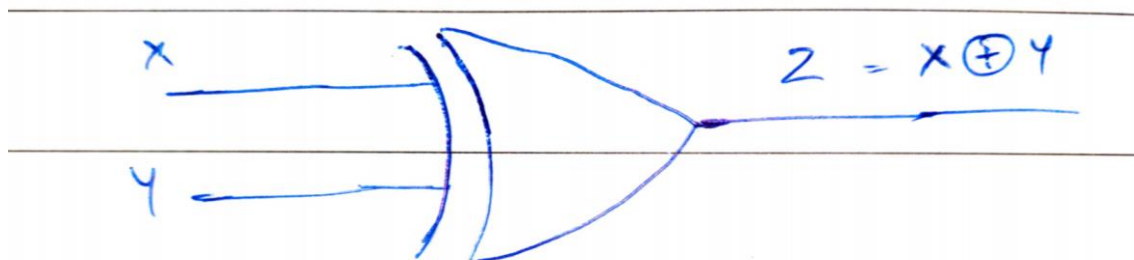
```
library ieee;
use ieee.std_logic_1164.all;
entity xyz is
port ( x,y: in std_logic;
      z : out std_logic);
end entity;
architecture arch_xyz of xyz is
begin
z <= '0' when (x='1' and y='1') else
      '0'when (x='0' and y='0')else
      '1';
end arch_xyz;
```

Ans:

The given code is similar to XOR gate as the output 'z' is low (0) if both the inputs 'x' and 'y' are same i.e. if both of them are high or if both of them are low at the same time. Output is high if 'x' and 'y' are different.

x	y	z
0	0	0
0	1	1
1	0	1
1	1	0

1912060



Q 2 Write a test bench for the above code

Ans:

1912060 - HDL PostLab Q2

```

library ieee;
use ieee.std_logic_1164.all;

entity xyz_tb is
end xyz_tb;

architecture xyz_tb_arch of xyz_tb is
    component xyz is
        port (
            x, y: in std_logic;
            z: out std_logic);
        end component;

    signal x, y, z: std_logic;

    begin
        uut: xyz port map (x, y, z);

        process begin
            x <= '0';
            y <= '1';
            wait for 20 ns;

            x <= '1';
            y <= '0';
            wait for 20 ns;

            x <= '1';
            y <= '1';
            wait for 20 ns;

        end process;
    end;

```

x <= '0';
y <= '1';
wait for 20 ns;

x <= '1';
y <= '0';
wait for 20 ns;

x <= '1';
y <= '1';
wait for 20 ns;

end process;

end;



Conclusion:

In this experiment we learned the basics of VHDL and how to use Quartus and simulate the wave using Modelsim.

The VHDL code for Half Adder was written first, which was then used to make the Full adder. Then, the full adder was used in the coding of 4-bit adder.

We also learnt to write testbench for the circuit, in which we give various inputs to the circuit so that we can see the output for different conditions.

Signature of faculty in-charge with Date: