Experiment / Assignment / Tutorial No. 7

Grade: AA / AB / BB / BC / CC / CD /DD

Signature of the Staff In-charge with date

Batch: A3	Roll No.: 1601012104	Experiment / assignment / tutorial No.: 7

Title: VHDL programming for gates.

Objective:

Expected Outcome of Experiment:

CO4: Implement digital networks using VHDL.

Books/ Journals/ Websites referred:

- ModelSim Software Link:
  https://www.mentor.com/company/higher_ed/modelsim-student-edition

- J. Bhasker, "VHDL Primer", Pearson Education
- Douglas L. Perry, "VHDL Programming by Example", Tata McGraw Hill
- http://esd.cs.ucr.edu/labs/tutorial/

Pre Lab/ Prior Concepts:

VHDL is an acronym for VHSlC Hardware Description Language (VHSIC is an acronym for Very High Speed Integrated Circuits). It is a hardware description language that can be used to model a digital system at many levels of abstraction ranging from the algorithmic level to the gate level. The complexity of the digital system being modeled could vary from that of a simple gate to a complete digital electronic system, or anything in between. The digital system can also be described hierarchically. Timing can also be explicitly modeled in the same description.

VHDL Programming Structure

Entity and Architecture are the two main basic programming structures in VHDL.

Entity: Entity can be seen as the black box view of the system. We define the inputs and outputs of the system which we need to interface. It is used to declare the I/O ports of the circuit.
Eg:

```
Entity ANDGATE is
        Port (A: in std_logic;
              B: in std_logic;
              Y: out std_logic);
End entity ANDGATE;
```

Entity name ANDGATE is given by the programmer, each entity must have a name.

Architecture: Architecture defines what is in our black box that we described using ENTITY. The description code resides within architecture portion. Either behavioral or structural models can be used to describe our system in the architecture. In Architecture we will have interconnections, processes, components, etc.
Eg:

```
Architecture AND1 of ANDGATE is
        --declarations
Begin
        --statements
        Y <= A AND B;
End architecture AND1;
```

Entity name or architecture name is user defined. Identifiers can have uppercase alphabets, lowercase alphabets, and numbers and underscore (_). First letter of identifier must be an alphabet and identifier cannot end with an underscore. In VHDL, keywords and user identifiers are case insensitive.

VHDL is strongly typed language i.e. every object must be declared. Standardized design libraries are typically used and are included prior to the entity declaration. This is accomplished by including the code "library ieee;" and "use ieee.std_logic_1164.all;"

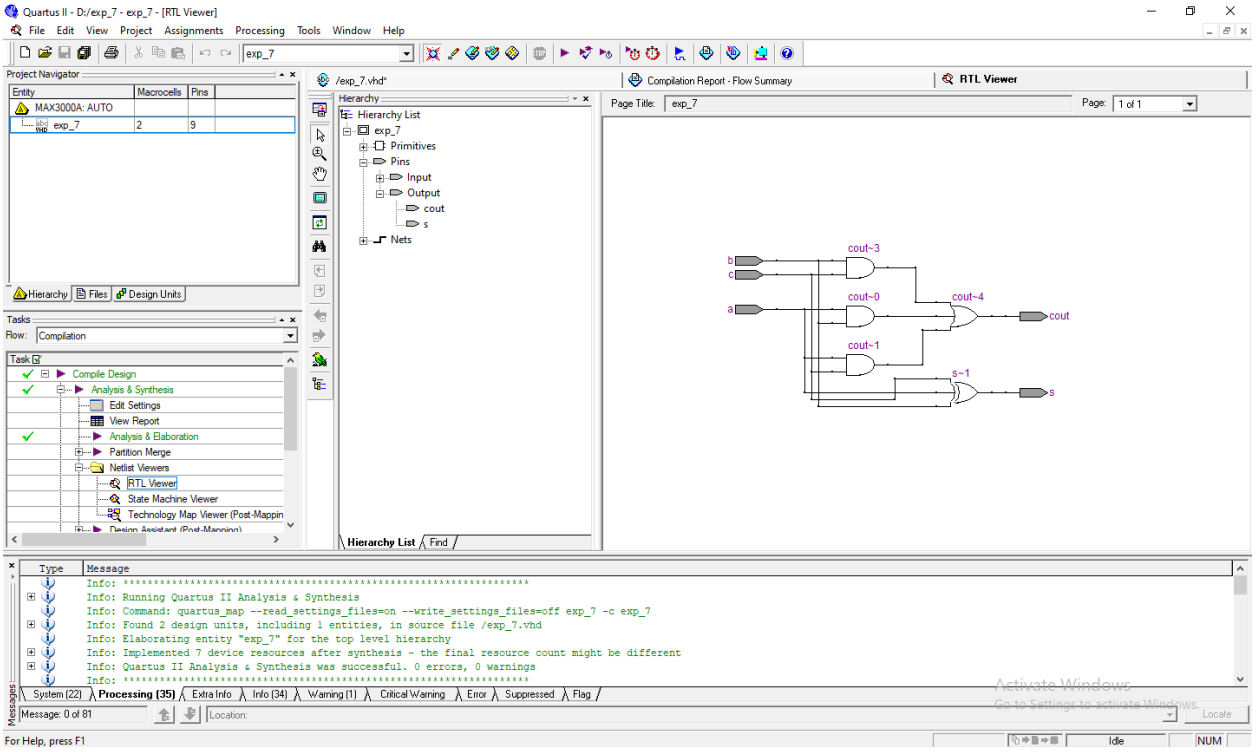
Implementation Details:

VHDL program code

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

entity exp_7 is
port(
a,b,c : in std_logic;
s, carry : out std_logic
);
end entity;

architecture exp7_arch of exp_7 is
begin
s <= a xor b xor c;
```

```
carry <= (a and b) or (c and a) or (c and b);
end;
```

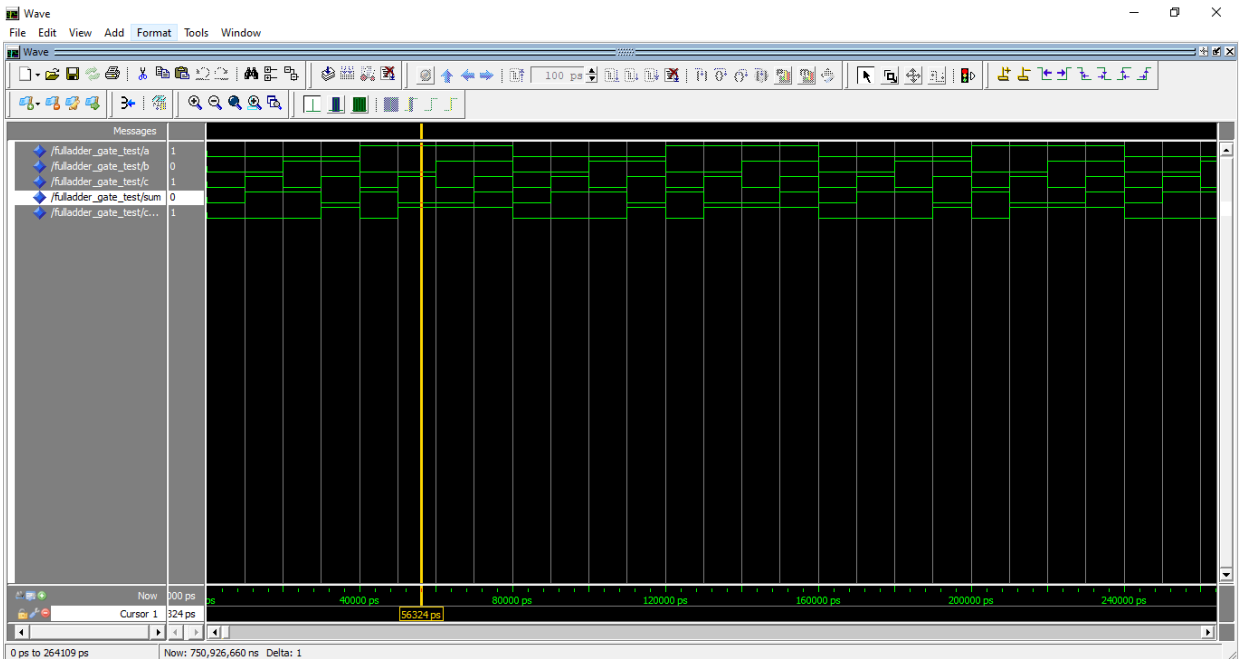
RTL Viewer Output:

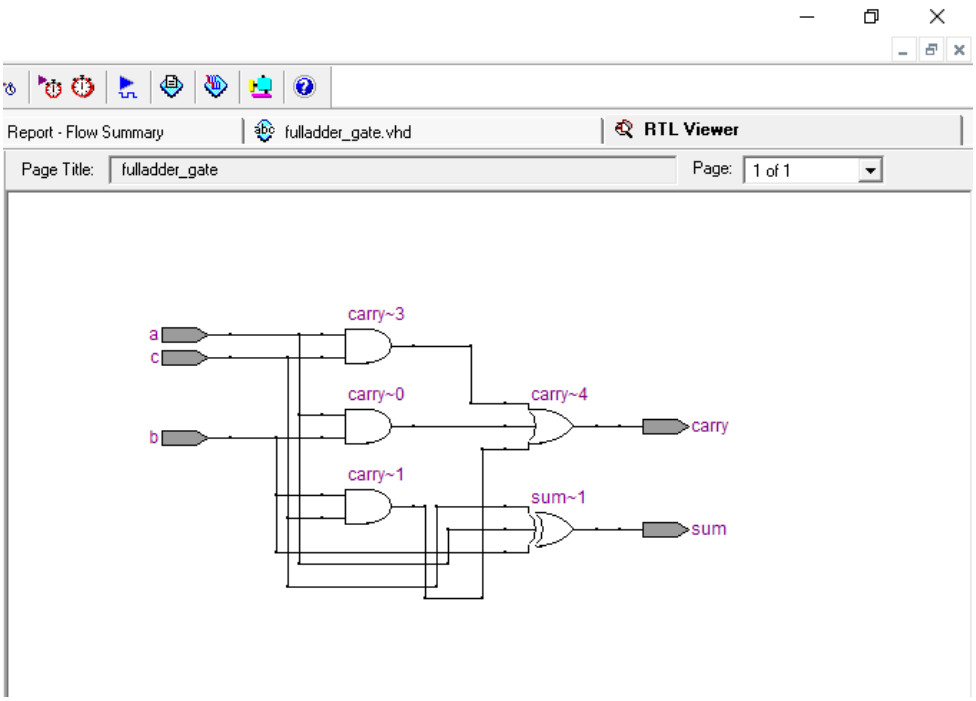


Full Adder Test Branch

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_arith.all;
use IEEE.STD_LOGIC_unsigned.all;

entity fulladder_gate_test is -- entity of tb should be blank
end fulladder_gate_test;
--}} End of automatically maintained section
architecture fulladder_gate_test_arch of fulladder_gate_test is
 component fulladder_gate is -- same as entity of your main code
port(
a,b,c : in std_logic;
sum,carry: out std_logic
);
```

```
 end component;

 signal a,b,c,sum,carry : STD_LOGIC;
begin
uut: fulladder_gate port map (a,b,c,sum,carry) ; --positional mapping
process
 begin -- write down the test cases
 a <='0';
 b<='0';
 c<='0';
 wait for 10 ns;
 a <='0';
 b<='0';
 c<='1';
 wait for 10 ns;
  a <='0';
 b<='1';
 c<='0';
  wait for 10 ns;
  a <='0';
 b<='1';
 c<='1';
  wait for 10 ns;
  a <='1';
 b<='0';
 c<='0';
 wait for 10 ns;
 a <='1';
 b<='0';
 c<='1';
 wait for 10 ns;
 a <='1';
 b<='1';
 c<='0';
 wait for 10 ns;
 a <='1';
 b<='1';
 c<='1';
 wait for 10 ns;
 end process;

end ;
```


Wave
File Edit View Add Format Tools Window
Messages
/fulladder_gate_test/a
/fulladder_gate_test/b
/fulladder_gate_test/c
/fulladder_gate_test/sum
/fulladder_gate_test/c...
Now
Cursor 1
40000 ps
80000 ps
120000 ps
160000 ps
200000 ps
240000 ps
56324 ps
0 ps to 264109 ps
Now: 750,926,660 ns Delta: 1


Report - Flow Summary
fulladder_gate.vhd
RTL Viewer
Page Title: fulladder_gate
Page: 1 of 1
a
c
b
carry~3
carry~0
carry~1
carry~4
sum~1
carry
sum

Conclusion:
Successfully implemented the given task.

Post Lab Descriptive Questions

1. What are two types of HDL?

HDLs are standard text-based expressions of the structure of electronic systems and their behaviors over time. Like concurrent programming languages, HDL syntax and semantics include explicit notations for expressing concurrency. However, in contrast to most software programming languages, HDLs also include an explicit notion of time, which is a primary attribute of hardware.

The two most widely used and well-supported HDL varieties used in industry are Verilog and VHDL:

The VHSIC Hardware Description Language (VHDL) is a hardware description language (HDL) that can model the behavior and structure of digital systems at multiple levels of abstraction, ranging from the system level down to that of logic gates, for design entry, documentation, and verification purposes. Verilog, standardized as IEEE 1364, is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level of abstraction. It is also used in the verification of analog circuits and mixed-signal circuits, as well as in the design of genetic circuits.