Lab report For Lab 1 of EECS 31L 2019 Fall

Student Name: Yuchen Yan

Student ID:28249722

Lab Session Hours: LAB A2 16442 M 6:00 – 8:50 P.M.

T.A.: CHOOKHACHIZADE.M.

The code is working.

Objective:

This lab is about designing a 2-bits full adder, in the lab\_1 document, we’ve learned about how to write 1 bit adder with VHDL, so I decided to use that 1 bit adder and form a 2 bit adder with what I learned from eecs31’s contents.

LAB1 CODE:

(Full Adder Part)

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Full\_adder is

Port (

a : in STD\_LOGIC;

b : in STD\_LOGIC;

c : in STD\_LOGIC;

sum : out STD\_LOGIC;

carry : out STD\_LOGIC );

end Full\_adder;

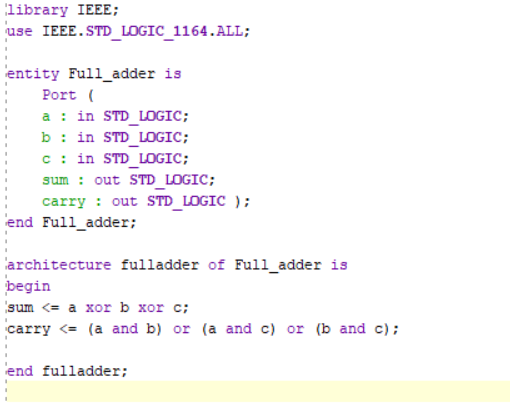
architecture fulladder of Full\_adder is

begin

sum <= a xor b xor c;

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

end fulladder;



Two bit full-adder by using one bit full adder:

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 FA is

-- Port ( );

PORT(

A2A1 : IN STD\_LOGIC\_VECTOR (1 downto 0) ;

B2B1 : IN STD\_LOGIC\_VECTOR (1 downto 0) ;

Cin : IN STD\_LOGIC ;

SUM1SUM2 : OUT STD\_LOGIC\_VECTOR (1 downto 0) ;

Cout : OUT STD\_LOGIC );

end FA;

architecture FA1 of FA is

component Full\_adder

PORT(

a,b,c: IN STD\_LOGIC ;

sum, carry : OUT STD\_LOGIC );

end component Full\_adder;

signal t1 : std\_logic;

begin

FA1: Full\_adder PORT MAP( a=>A2A1(0), b=>B2B1(0), c=>Cin, sum=>SUM1SUM2(0), carry=>t1);

FA2: Full\_adder PORT MAP( a=>A2A1(1), b=>B2B1(1), c=>t1, sum=>SUM1SUM2(1), carry=>Cout);

END FA1;

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Full\_adder is

Port (

a : in STD\_LOGIC;

b : in STD\_LOGIC;

c : in STD\_LOGIC;

sum : out STD\_LOGIC;

carry : out STD\_LOGIC );

end Full\_adder;

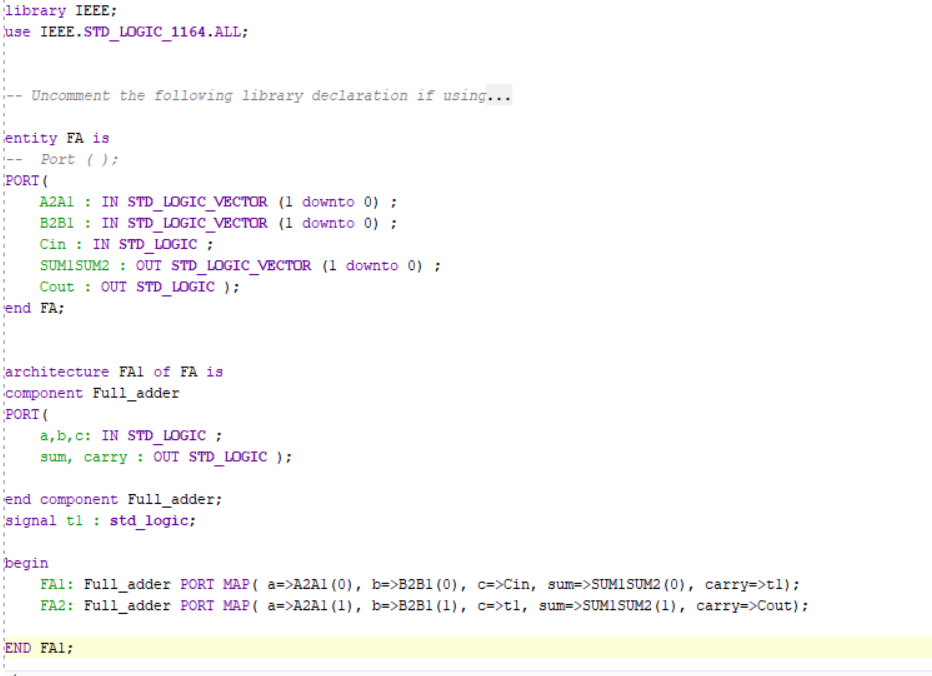
architecture fulladder of Full\_adder is

begin

sum <= a xor b xor c;

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

end fulladder;



Output screenshots of two bits full adder:

