**Mux code**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.numeric\_std.all;

entity mux4ta1 is --this is the code that used for the 4 to 1 mux

port( a,b,c,d: in std\_logic\_vector(3 downto 0);

sel: in std\_logic\_vector(1 downto 0);

z: out std\_logic\_vector (3 downto 0));

end mux4ta1;

architecture mux of mux4ta1 is

begin

process( a,b,c,d,sel) is

begin

--this part of the code is going to dicatate what happens if select bits are picked

if (sel(0) ='0' and sel(1) = '0') then

z <= a;

elsif (sel(0) ='1' and sel(1) = '0') then

z <= c;

elsif (sel(0) ='0' and sel(1) = '1') then

z <= b;

else

z <= d;

end if;

end process;

end mux;

**mux top level code**

--This part of the code is used for the top level of the mux

entity muxcomp is

port(a,b: in bit\_vector (3 downto 0);

sel : in bit\_vector (1 downto 0);

z:out bit\_vector (3 downto 0));

end;

--This part of the code makes the mux

architecture muxcomp2 of muxcomp is

component mux

port( a,b,c,d: in bit\_vector(3 downto 0);

sel: in bit\_vector(1 downto 0);

z: out bit\_vector (3 downto 0));

end component;

--This part of the code makes the mux a component

signal op0:bit\_vector (3 downto 0);

signal op1:bit\_vector (3 downto 0);

signal op2:bit\_vector (3 downto 0);

signal op3:bit\_vector (3 downto 0);

--This part of the code controls all the operations needed for the top level

begin

op0<=((a(3) and b(3))& (a(2) and b(2))& (a(1) and b(1))&(b(0) and a(0)));

op1<=

((a(3) or b(3))& (a(2) or b(2))& (a(1) or b(1))&(b(0) or a(0)));

op2<=

((a(3) nand b(3))& (a(2) nand b(2))& (a(1) nand b(1))&(b(0) nand a(0)));

op3<=

((a(3) nor b(3))& (a(2) nor b(2))& (a(1) nor b(1))&(b(0) nor a(0)));

--This part of the code says all the operations that need tob e used

muxco2:mux port map(op0,op1,op2,op3,sel,z);

--This part of the code puts all of them into a port map so they can be used for later

end;

**Test bench for top level**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.numeric\_std.all;

entity test\_muxcomp is

end test\_muxcomp;

architecture test of test\_muxcomp is

signal a,b: std\_logic\_vector(3 downto 0);

signal sel: std\_logic\_vector(1 downto 0);

signal z: std\_logic\_vector(3 downto 0);

component mux is

port( a,b,c,d : in std\_logic\_vector(3 downto 0);

sel: in std\_logic\_vector (1 downto 0);

y : out std\_logic\_vector(3 downto 0) );

end component;

--below are all my test symbols that will be used

signal ta,tb,tc,td: std\_logic\_vector(3 downto 0);

signal selT: std\_logic\_vector(1 downto 0);

signal y: std\_logic\_vector(3 downto 0);

begin

DUT1:mux port map(ta,tb,tc,td,selT,y);

process

begin

--this is to toggle the select bits so that I can get different outputs and that is done for steps below it

selT <= "00";

wait for 10 ns;

selT <= "01";

wait for 10 ns;

selT <="10";

wait for 10 ns;

selT <="11";

wait for 10 ns;

end process;

process

begin

a <="1001";

b<= "1100";

wait for 100 ns;

--report "input = " & std\_logic'image(input(2)) & std\_logic'image(input(1)) & std\_logic'image(input(0)) & " isprime = " & std\_logic'image(isprime);

end process;end test;

**Test bench for mux**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.std\_logic\_unsigned.all;

use ieee.numeric\_std.all;

entity test\_mux is

end test\_mux;

architecture test of test\_mux is

signal a,b: std\_logic\_vector(3 downto 0);

signal sel: std\_logic\_vector(1 downto 0);

signal z: std\_logic\_vector(3 downto 0);

component mux is

port( a,b,c,d : in std\_logic\_vector(3 downto 0);

sel: in std\_logic\_vector (1 downto 0);

y : out std\_logic\_vector(3 downto 0) );

end component;

signal ta,tb,tc,td: std\_logic\_vector(3 downto 0);

signal selT: std\_logic\_vector(1 downto 0);

signal y: std\_logic\_vector(3 downto 0);

begin

DUT1:mux port map(ta,tb,tc,td,selT,y);

process

begin

--this is to toggle the select bits so that I can get different outputs and that is done for steps below it

selT <= "00";

wait for 10 ns;

selT <= "01";

wait for 10 ns;

selT <="10";

wait for 10 ns;

selT <="11";

wait for 10 ns;

end process;

process

begin

for i in 0 to 15 loop

ta <= std\_logic\_vector(to\_unsigned(i,4));

tb <= std\_logic\_vector(to\_unsigned(i,4));

tc <= std\_logic\_vector(to\_unsigned(i,4));

td <= std\_logic\_vector(to\_unsigned(i,4));

wait for 100 ns;

--report "input = " & std\_logic'image(input(2)) & std\_logic'image(input(1)) & std\_logic'image(input(0)) & " isprime = " & std\_logic'image(isprime);

assert y /= y;

end loop;

std.env.stop(0);

end process;end test;



