Samuel Lichtenheld

9/10/15

Assignment\_1

Appendix

Test bench files can be found starting on page 8

Introduction

The purpose of this lab is to familiarize one’s self with the tools being used in the class, such as Quartus and Modelsim. Learning about propagation delay and determining the maximum clock rate was covered as well.

1.1

Quartus installed.

1.2

The quartus tutorial was not very helpful for me personally, because I am already familiar with quartus as I have used it in Digital Logic, Microprocessors, and Digital Design.

1.3

Quartus 15.0 and Modelsim 10.3 installed.

1.4

Similar to the quartus tutorial, the modelsim tutorial was not very helpful because I am already used to this software.

2.1

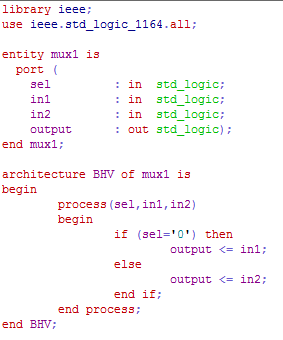


Figure : Timing simulation of Mux1

2.1.1

~7 ns

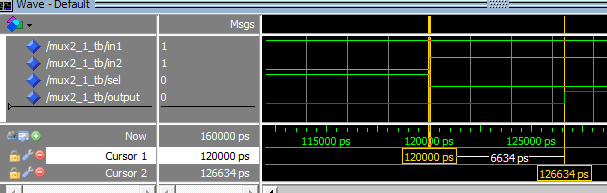


Figure : Worst case propagation time for Mux1

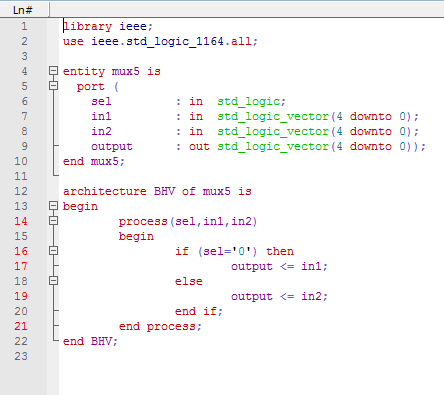


Figure : Mux5

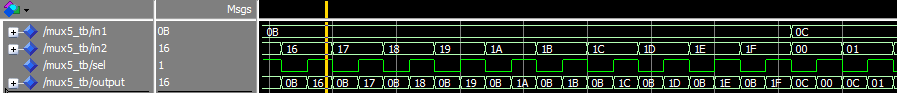


Figure : Mux5 tested

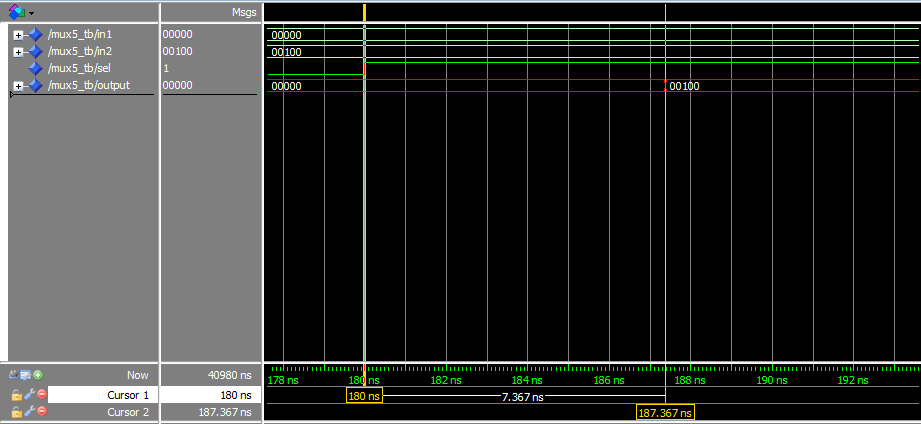


Figure : Mux5 timing calculated

2.1.2

Rs, rt, and rd are 5 bit numbers because there are 32 registers in MIPS. 2^5=32

2.1.3

Worst case time is ~7ns.

2.1.4

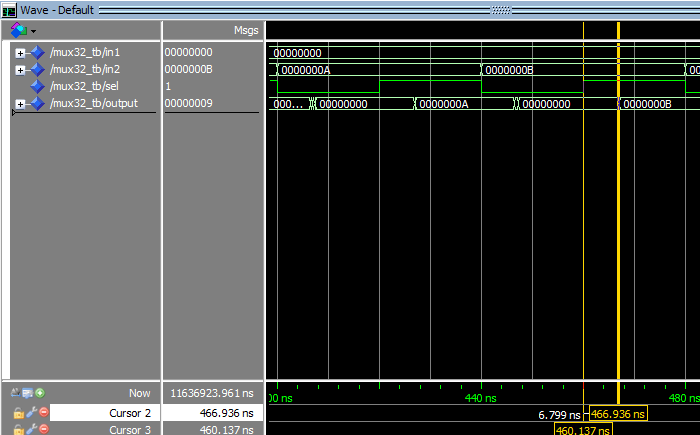


Figure : Mux32 Timing Simulation

Worst case is ~7ns

2.1.5

There are no differences in the propagation times for the three multiplexer designs. My guess is that they all can fit on one LUT.

2.2

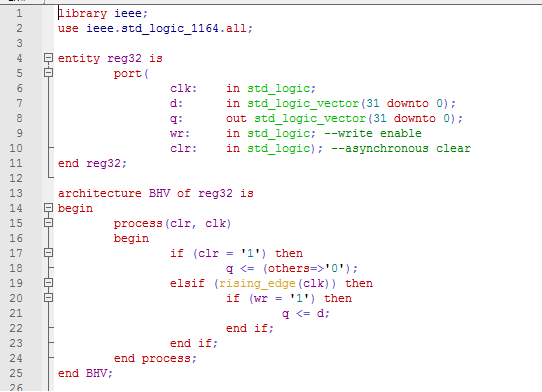
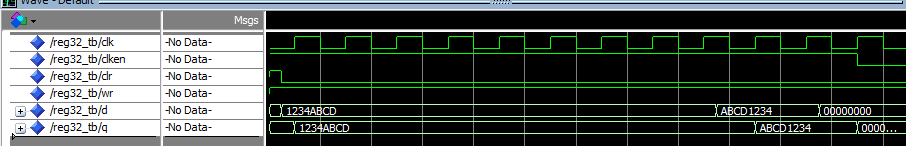
 

Figure : Reg32 Test

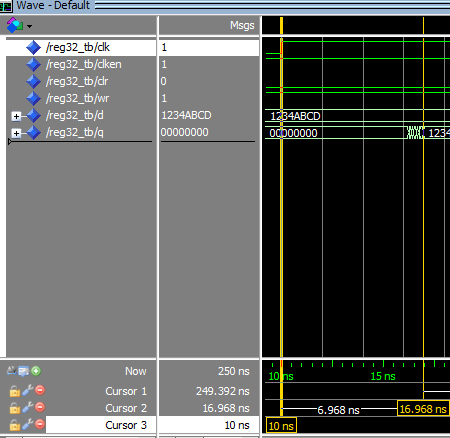


Figure : Reg32 Timing calculated

2.2.1

MCLK = 1/TMIN = 1/6.968ns = 143.5 Mhz

2.3

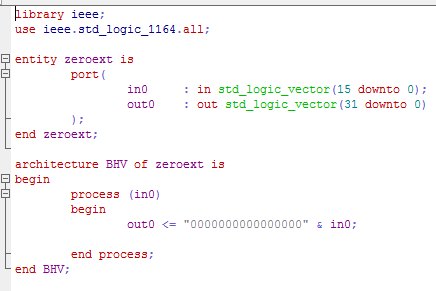


Figure : Zeroext VHDL

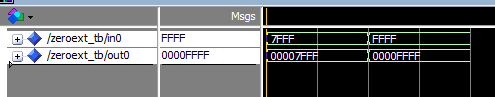


Figure : Zeroext Simulation

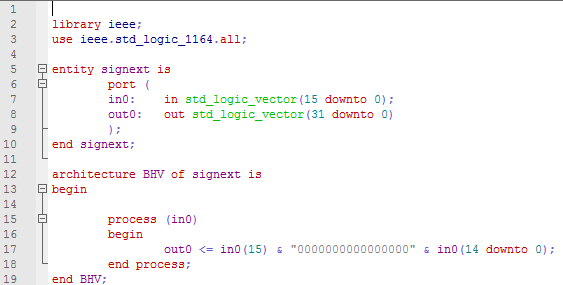


Figure : Signext VHDL



Figure : Signext Simulation

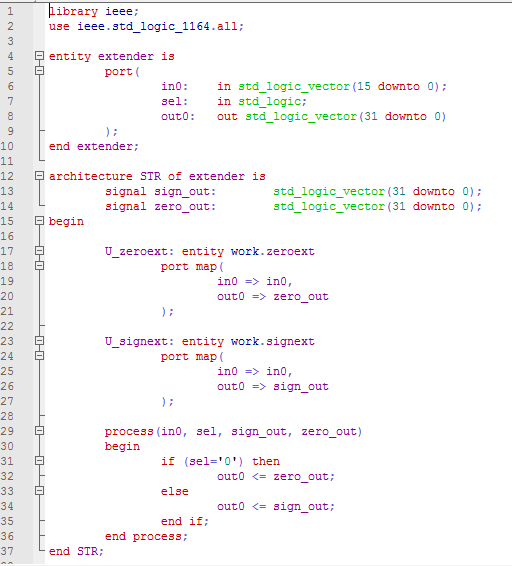


Figure : Extender VHDL

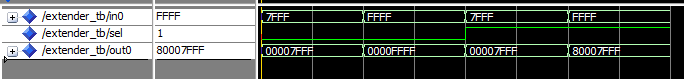


Figure : Extender Simulation

Testbenches

**REG32**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.math\_real.all;

use ieee.numeric\_std.all;

entity mux32\_tb is

end mux32\_tb;

architecture TB of mux32\_tb is

signal in1 : std\_logic\_vector(31 downto 0);

signal in2 : std\_logic\_vector(31 downto 0);

signal sel : std\_logic;

signal output : std\_logic\_vector(31 downto 0);

begin -- TB

UUT : entity work.mux32

port map (

sel => sel,

in1 => in1,

in2 => in2,

output => output);

process

variable temp : std\_logic\_vector(64 downto 0);

function mux\_test (

signal in1: std\_logic\_vector(31 downto 0);

signal in2: std\_logic\_vector(31 downto 0);

signal sel: std\_logic)

return std\_logic\_vector is

begin -- mux\_test

if (sel = '0') then

return in1;

else

return in2;

end if;

end mux\_test;

begin

--test all input combinations

for i in 0 to 2\*\*30 loop

temp := std\_logic\_vector(to\_unsigned(i,65));

in1 <= temp(64 downto 33);

in2 <= temp(32 downto 1);

sel <= temp(0);

wait for 20 ns;

assert(output = mux\_test(in1,in2,sel))

report "Error : output\_with\_select incorrect" severity warning;

end loop;

wait;

report "SIMULATION FINISHED!!!";

wait;

end process;

end TB;

**REG 32**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity reg32\_tb is

end reg32\_tb;

architecture TB of reg32\_tb is

signal clk : std\_logic := '0';

signal clken : std\_logic := '1';

signal clr : std\_logic;

signal wr : std\_logic;

signal d : std\_logic\_vector(31 downto 0);

signal q : std\_logic\_vector(31 downto 0);

begin

UUT : entity work.reg32

port map(

clk => clk,

clr => clr,

wr => wr,

d => d,

q => q

);

clk <= not clk and clken after 10 ns;

process

begin

clr <= '1';

wr <= '1';

d <= (others => '0');

wait for 5 ns;

clr <= '0';

d <= x"1234ABCD";

for i in 0 to 8 loop

wait until (rising\_edge(clk));

end loop;

wait for 5 ns;

d<= x"ABCD1234";

for i in 0 to 1 loop

wait until (rising\_edge(clk));

end loop;

wait for 5 ns;

d<= (others=> '0');

wait until (rising\_edge(clk));

report "SIMULATION FINISHED!!!";

clken <= '0';

wait;

end process;

end TB;

**EXTENDER**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

entity extender\_tb is

end extender\_tb;

architecture TB of extender\_TB is

signal in0: std\_logic\_vector(15 downto 0);

signal sel: std\_logic;

signal out0: std\_logic\_vector(31 downto 0);

begin

UUT: entity work.extender

port map(

in0 => in0,

sel => sel,

out0 => out0

);

process

function extend\_test (

signal in0: std\_logic\_vector(15 downto 0);

signal sel: std\_logic;

signal out0: std\_logic\_vector(31 downto 0))

return std\_logic is

begin -- zeroext test

if (sel='0') then

if (out0= in0(15) & "0000000000000000" & in0(14 downto 0)) then

return '1';

else

return '0';

end if;

else

if (out0= in0(15) & "0000000000000000" & in0(14 downto 0)) then

return '1';

else

return '0';

end if;

end if;

end extend\_test;

begin -- begin process

sel <= '0';

in0 <= "0111111111111111";

wait for 10 ns;

assert(extend\_test(in0,sel,out0) = '1')

report "Error : output incorrect for in0 = 0x7FFF and sel=" & std\_logic'image(sel) severity warning;

in0 <= "1111111111111111";

wait for 10 ns;

assert(extend\_test(in0,sel,out0) = '1')

report "Error : output incorrect for in0 = 0xFFFF and sel=" & std\_logic'image(sel) severity warning;

sel <= '1';

in0 <= "0111111111111111";

wait for 10 ns;

assert(extend\_test(in0,sel,out0) = '1')

report "Error : output incorrect for in0 = 0x7FFF and sel=" & std\_logic'image(sel) severity warning;

in0 <= "1111111111111111";

wait for 10 ns;

assert(extend\_test(in0,sel,out0) = '1')

report "Error : output incorrect for in0 = 0xFFFF and sel=" & std\_logic'image(sel) severity warning;

report "Simulation Finished";

wait;

end process;

end TB;