ENEL 453 Laboratory 1

B02, B06

Brendon Kopp & Benjamin Hilborn

January 20, 2015

We declare that this laboratory report is entirely our own work and includes no material which has been copied from any other source excepting that material which is clearly identified as the work of others.

Table of Contents

5.1 Complete the testbench 2

5.1.1 Screenshot of waveform 2

5.1.2 Testbench code 2

5.2 Implement original design 4

5.2.1 Screenshot of RTL schematic 4

5.2.2 Discussion on differences from prelab 4

5.2.3 Code from the constraints file 4

5.3 Modify the design 5

5.3.1 Screenshot of waveforms 5

5.3.2 Modified switch logic code 5

5.4 Generating modified design 6

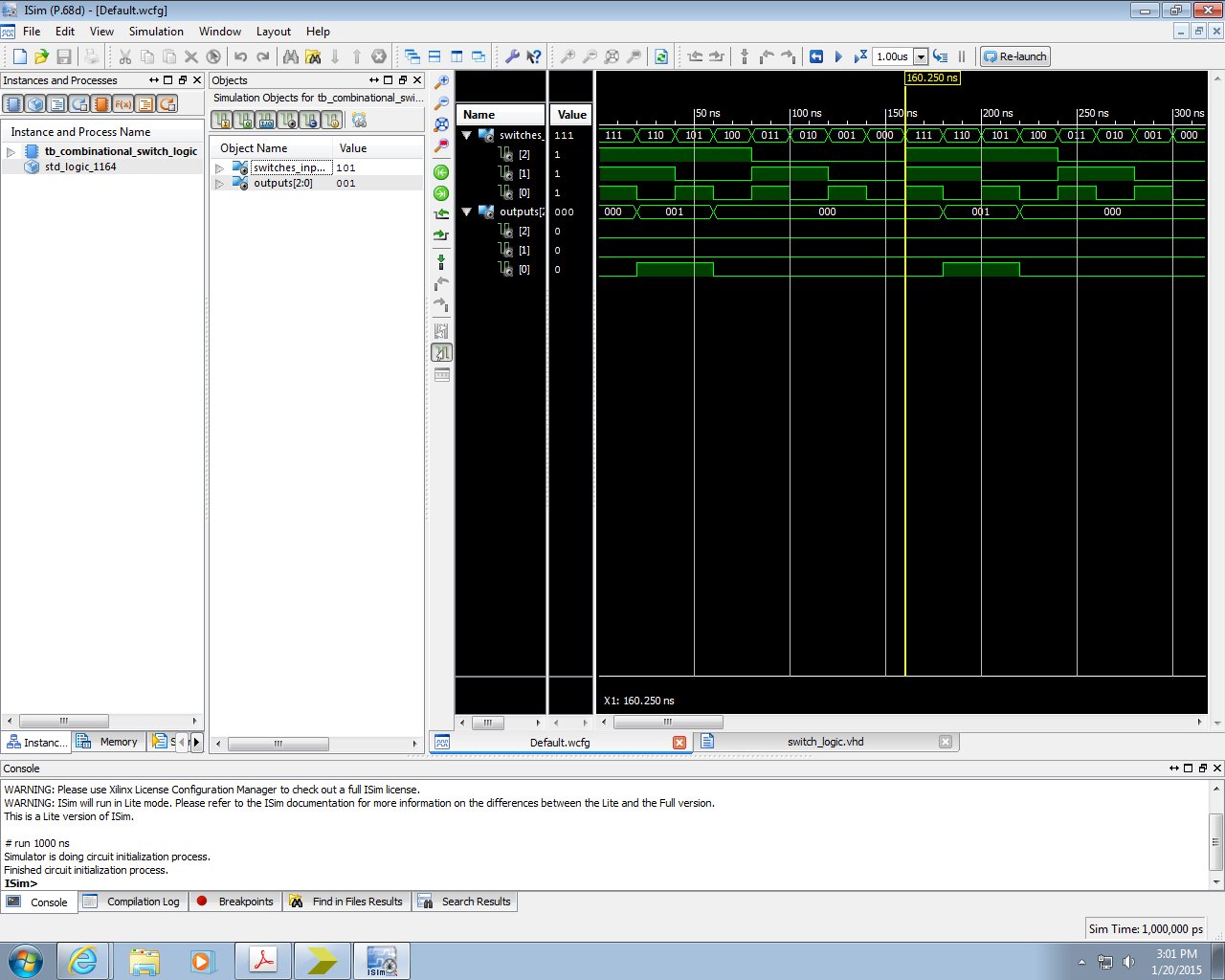
5.4.1 Screenshot of RTL schematic 6

5.4.2 Discussion on changes 6

6 Additional Questions 7

# 5.1 Complete the testbench

## 5.1.1 Screenshot of waveform



## 5.1.2 Testbench code

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;

ENTITY tb\_combinational\_switch\_logic IS

END tb\_combinational\_switch\_logic;

ARCHITECTURE behavior OF tb\_combinational\_switch\_logic IS

-- Component Declaration for the Unit Under Test (UUT)

COMPONENT switch\_logic

PORT(

switches\_inputs : IN std\_logic\_vector(2 downto 0);

outputs : OUT std\_logic\_vector(2 downto 0)

);

END COMPONENT;

--Inputs

signal switches\_inputs : std\_logic\_vector(2 downto 0) := (others => '0');

--Outputs

signal outputs : std\_logic\_vector(2 downto 0);

-- No clocks detected in port list. Replace <clock> below with

-- appropriate port name

--constant <clock>\_period : time := 10 ns;

BEGIN

-- Instantiate the Unit Under Test (UUT)

uut: switch\_logic PORT MAP (

switches\_inputs => switches\_inputs,

outputs => outputs

);

-- Clock process definitions

--<clock>\_process :process

--begin

-- <clock> <= '0';

-- wait for <clock>\_period/2;

-- <clock> <= '1';

-- wait for <clock>\_period/2;

--end process;

A\_process: process

begin

switches\_inputs(0) <= '1';

--YOU DO THIS

wait for 20ns; --(pick some value, for example 20 ns)

switches\_inputs(0) <= '0';

wait for 20ns;

end process;

B\_process: process

begin

switches\_inputs(1) <= '1';

--YOU DO THIS

wait for 40ns;

switches\_inputs(1) <= '0';

wait for 40ns;

end process;

C\_process: process

begin

switches\_inputs(2) <= '1';

--YOU DO THIS

wait for 80ns;

switches\_inputs(2) <= '0';

wait for 80ns;

end process;

-- Stimulus process

stim\_proc: process

begin

-- -- hold reset state for 100 ns.

-- wait for 100 ns;

-- -- Set all inputs to 0

-- switches\_inputs(0) <= '0'; --A

-- switches\_inputs(1) <= '0'; --B

-- switches\_inputs(2) <= '0'; --C

-- wait for 50 ns;

-- -- Test an input combination

-- switches\_inputs(0) <= '1'; --A

-- switches\_inputs(1) <= '0'; --B

-- switches\_inputs(2) <= '0'; --C

-- wait for 100 ns;

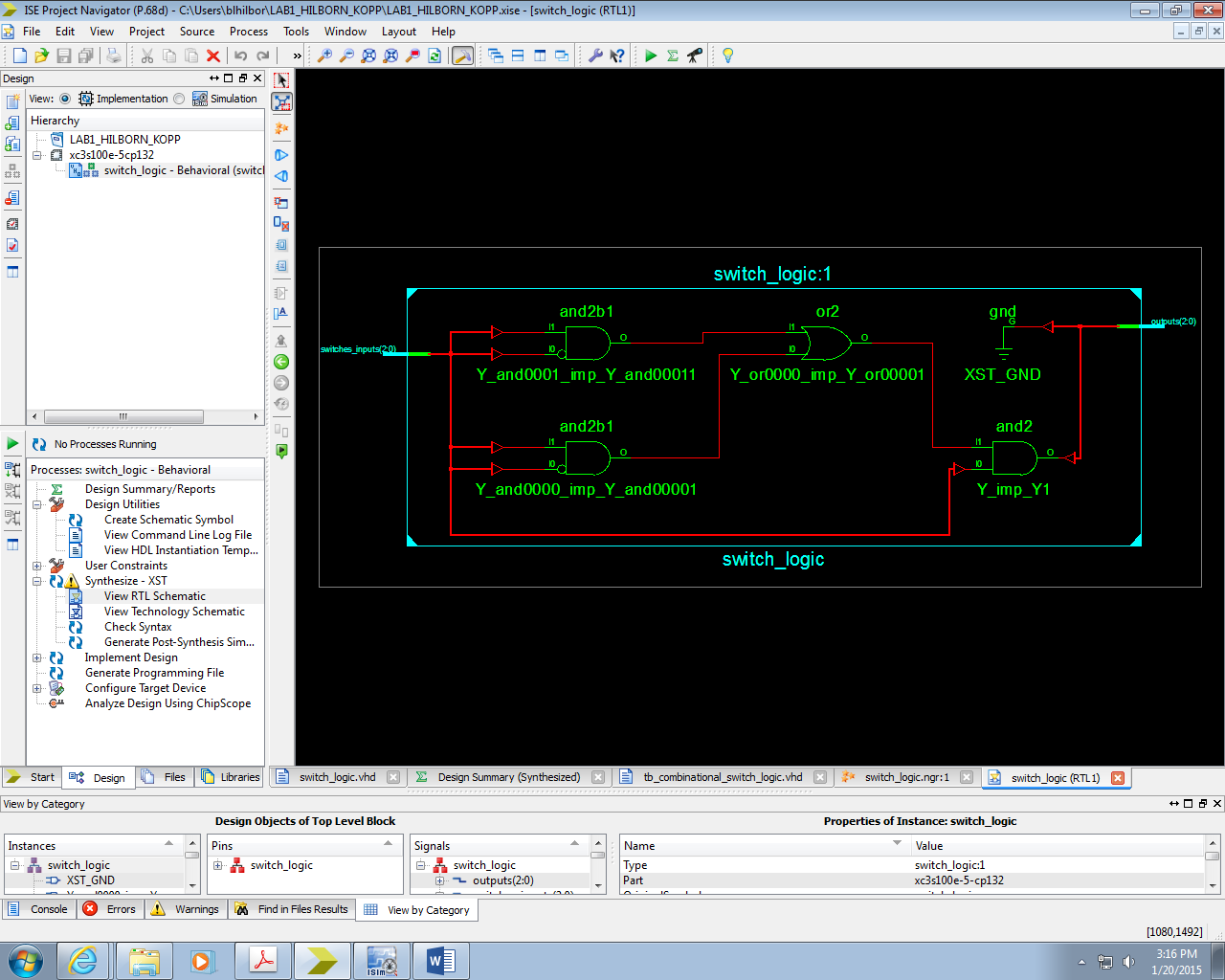
wait; -- Keeps it from restarting

end process;

END;

# 5.2 Implement original design

## 5.2.1 Screenshot of RTL schematic



## 5.2.2 Discussion on differences from prelab

This very closely matches the diagrams we designed for the prelab. The only difference between them is that we used NOR gates, while the generated schematic used inverting inputs to the gates.

## 5.2.3 Code from the constraints file

NET "outputs(0)" LOC="M5"; #LED0

NET "outputs(1)" LOC="M11"; #LED1

NET "outputs(2)" LOC="P7"; #LED2

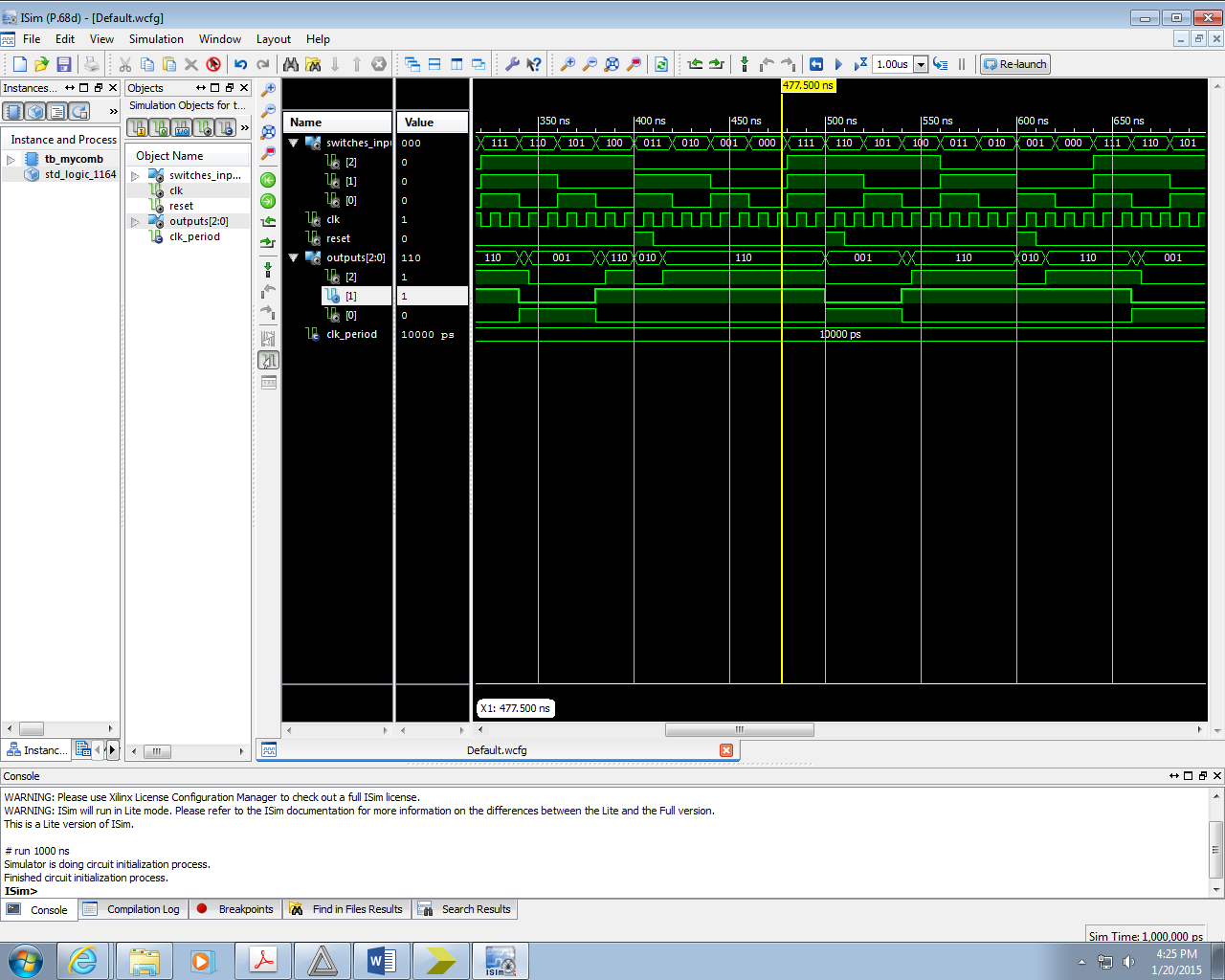
NET "switches\_inputs(0)" LOC="N3"; #switch0

NET "switches\_inputs(1)" LOC="E2"; #switch1

NET "switches\_inputs(2)" LOC="F3"; #switch2

# 5.3 Modify the design

## 5.3.1 Screenshot of waveforms



## 5.3.2 Modified switch logic code

architecture Behavioral of switch\_logic is

-- Internal signals:

signal Y: std\_logic;

signal X\_th, X\_mine: std\_logic;

signal A, B, C: std\_logic;

begin

logic\_of\_switches: process(clk, reset) begin

if (reset = '1') then

-- Reset X\_mine to a known state

X\_mine <= '0';

elsif (rising\_edge(clk)) then

-- Assign logic to X\_mine here

X\_mine <= (not C) or (B and A) or ((not A) and (not B));

end if;

end process;

-- Combinational logic!

Y <= C and (((not B) and A) or ((not A) and B));

X\_th <= not (C and (((not B) and A) or ((not A) and B)));

-- Assign the outputs. We only have one signal for now

outputs(0) <= Y;

outputs(1) <= X\_th; -- We will connect these later

outputs(2) <= X\_mine;

-- Grab the inputs from the slide switches on the FPGA board

A <= switches\_inputs(0);

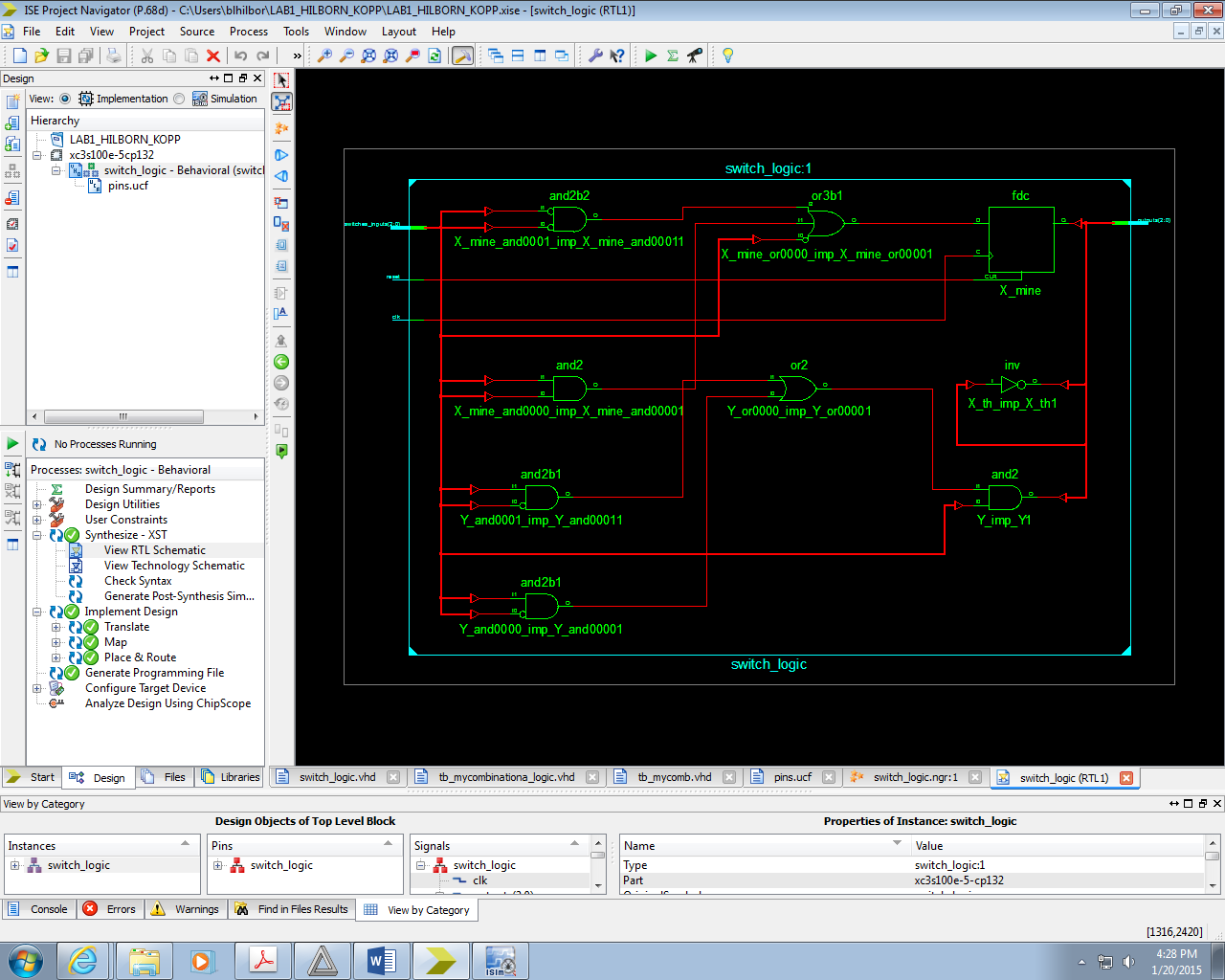
B <= switches\_inputs(1);

C <= switches\_inputs(2);

end architecture Behavioral;

# 5.4 Generating modified design

## 5.4.1 Screenshot of RTL schematic



## 5.4.2 Discussion on changes

This schematic appears to accurately represent the changes we made to our code. However, we are unsure as to the function of the inverter labeled X\_th\_imp\_X\_th1.

# 6 Additional Questions

1. From the synthesis report for the sequential circuit, extract the maximum clock frequency that the synthesized circuit can operate at. How is this maximum frequency calculated?
   1. The maximum clock frequency is 124.4 MHz which was calculated from the pad to pad delay (8.041ns). This maximum frequency is calculated by taking the highest max delay (of CLK-to-OUT, and PAD-to-PAD) shown in the PAR clock report and converting to frequency (1/T).
2. What is the meaning of “concurrent statements”? How is this different from a sequential high-level language like C?
   1. A concurrent statement executes continuously. It does not wait for previous lines of code before it executes. A process is an example of a concurrent statement. In a sequential language like C, a line of code is executed only after the previous line of code has been executed.