**Tim Menninger**

**Problem Set 1**

**Problem 1: Introductory Code**

a) …

b) Write relevant comments regarding solution.

Lines 2-3: The library declaration was given to us. We had to fill in the data use clause. The use of the *ieee* library was for the STD\_LOGIC data type, and therefore, the use clause for the *ieee* library was *std\_logic\_1164.all*.

Lines 5-10: Define the entities of the program.

Lines 12-29: We define the third part of our code (architecture) whose name matches the entity (mux).

Line 7: Inside the entity section, we must declare our inputs and outputs. This line declares two of our inputs, a and b, and defines their data type: std\_logic\_vector with 8 bits.

Line 8: We define the other input, which is of different type than a and b. The sel input is std\_logic\_vector, also, but is only two bits (1 downto 0).

Line 9: We define the output, x, as a std\_logic\_vector with 8 bits (7 downto 0).

Line 10: Formally ends the entity section.

Lines 15-27: Define a process with three arguments: a, b and sel. We look at four cases for “00” through “11” and handle each output accordingly.

Lines 17-19: If the sel input is “00”, we output “00000000”.

Lines 20-21: If the sel input is “01”, we output the value in a.

Lines 22-23: If the sel input is “10”, we output the value in b.

Lines 24-25: If the sel input is “11”, we output “ZZZZZZZZ”.

Line 26: Formally ends the if structure.

Line 27: Formally ends the process.

Line 29: Formally ends the architecture section.

c) …

**Problem 2-4:** …

**Problem 5:**

library ieee;

use ieee.std\_logic\_1164.all;

use ieee.numeric\_std.all;

----------------------------------------------------------------------------

entity adder is

generic (

n : natural := 8);

port (

a, b : in std\_logic\_vector((n – 1) downto 0);

cin : in std\_logic;

sum : out std\_logic\_vector((n – 1) downto 0);

cout : out std\_logic);

end entity

----------------------------------------------------------------------------

architecture computation of adder is

begin

process (a, b, cin)

signal uns\_a, uns\_b : unsigned;

signal pre\_sum : std\_logic\_vector(n downto 0);

begin

uns\_a <= unsigned(a);

uns\_b <= unsigned(b);

pre\_sum <= std\_logic\_vector(a + b);

sum <= pre\_sum((n-1) downto 0);

cout <= pre\_sum(n);

end process;

end architecture;