COEN 313
Digital Systems Design II-Winter 2017
LAB 2: Structural and Concurrent VHDL

The purpose of this lab is to become acquainted with structural and concurrent VHDL A secondary purpose is to become acquainted with different VHDL coding styles and to gain insight on the differences between writing VHDL code for simulation versus synthesis.

Introduction

An 8-bit two's complementer circuit will be designed in this lab using both structural and concurrent VHDL. Recall that the two's complement of a binary number is equal to the one's complement of the number added with "1". For example, the two's complement of `0110 1101` is:

```
   1001 0010   ( flip the bits )
+         1    ( and then add 1)
   _________
   1001 0011   ( = the two's complement of 0110 1101)
```

The "flip the bits" may be easily implemented with a sufficient number of inverters. Adding "1" to the flipped bits can be performed with a parallel adder with one of the adder inputs hardwired to the constant value of "1". The block diagram of the two's complementer circuit is shown in Figure 1.

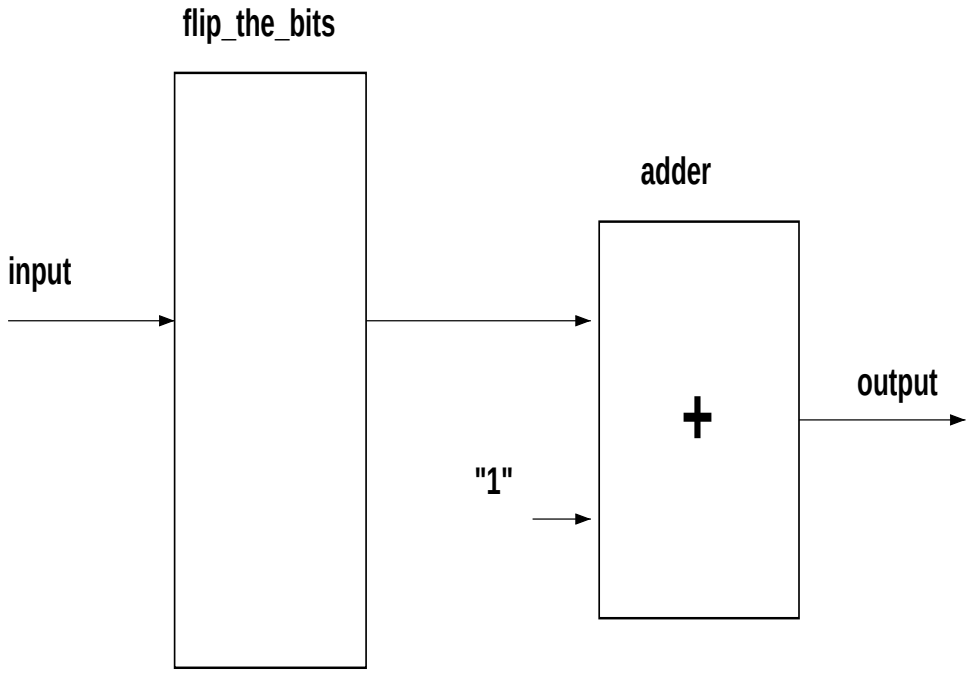

Figure 1: Block diagram of circuit.

Procedure

Employ **structural** VHDL in your design by implementing the circuit shown in Figure 1 using a **port map** statement to implement the "flip the bits" component. and a **concurrent signal assignment statement** to implement the parallel adder. The "flip the bits" component itself should also be implemented using **8 port map statements** instantiating 8 inverters. Write entity-architecture descriptions for the inverter gate using concurrent VHDL (i.e. the `not` operator for the inverter). You may use a concurrent signal assignment statement in your top-level code to negate any outputs which drive the LED outputs. Simulate your design with the Modelsim simulator to verify correct functioning for several typical values of the input. Synthesize your VHDL code with Precision RTL and obtain the RTL schematic diagram as produced by the synthesis tool. Program the FPGA board with the Xilinx ISE tool. Demonstrate the operation of the design by downloading your synthesized code to the FPGA demonstration board. Use the following VHDL entity specification:

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;  -- for the "+" operator

entity twos_comp is
port( input      : in std_logic_vector(7 downto 0);
      output     : out std_logic_vector(7 downto 0));
end twos_comp;
```

Questions

1. Rewrite the VHDL code for only the "flip the bits" component making use of only CSA statements (no port maps). Re-synthesize your design with Precision RTL and compare the resulting RTL schematic diagram with that of the original design. Comment on any differences/similarities between the two schematics. You do not have to download this version of the circuit to the FPGA board.

2a. Explain (by giving the VHDL code) how you would re-write the original code if the input and output to the circuit where changed from `std_logic_vector(7 downto 0)` to `std_logic_vector(15 downto 0)`.

2b. Would it be practical to extend the method as described in Question 2a to vectors of size 256 bits or 512 bits or 1024 bits?

3. Explain (by giving the VHDL code) how you would use **only** concurrent signal assignment statements to design a two's complementer with an input and output of size 16 bits with the following entity specification:

```
library IEEE;
```

```
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity twos_comp is
port( input      : in std_logic_vector(15 downto 0);
      output     : out std_logic_vector(15 downto 0));
end twos_comp;
```

4. After having answered Question 3, can you now appreciate the advantages of concurrent signal assignment statements and vector data types over the laborious `port map` statement?

5. What changes are necessary in your top-level VHDL code (for the `twos_comp` entity) in order to be able to **simulate** the `twos_comp` entity which uses the version of the ““flip the bits” component which was designed in Question 1? Were any changes necessary in the top-level twos_comp entity required in order to **synthesize** the design as in Question 1? In other words , explain the differences (between a simulation point of view and a logic synthesis point of view) in a compoment specification statement such as:

```
for bit_flipper : flip_the_bits use entity WORK.flip_the_bits(port_maps) ;
```

T. Obuchowicz
February 2, 2017