EELE 367 - Introduction to Logic Circuits

Lab 12(b) – Signed Binary Adders

Objective

This lab will give you more experience modeling addition circuits in VHDL by creating a 4-bit signed adder. The inputs for your adder will come from the switches on the DE0-CV FPGA board. The inputs and sum will each be displayed using two HEX displays. One HEX display will show the sign of the number and the other will show the magnitude of the number. The carry and two's complement overflow flags will also be calculated and displayed on the red LEDs.

Outcomes

After completing this lab you should be able to:

- Design a 4-bit, signed adder using the "+" operator from the number_std package.
- Type cast back and forth between types *signed* and *std_logic_vector*.
- Compute whether two's complement overflow occurred for a signed addition.

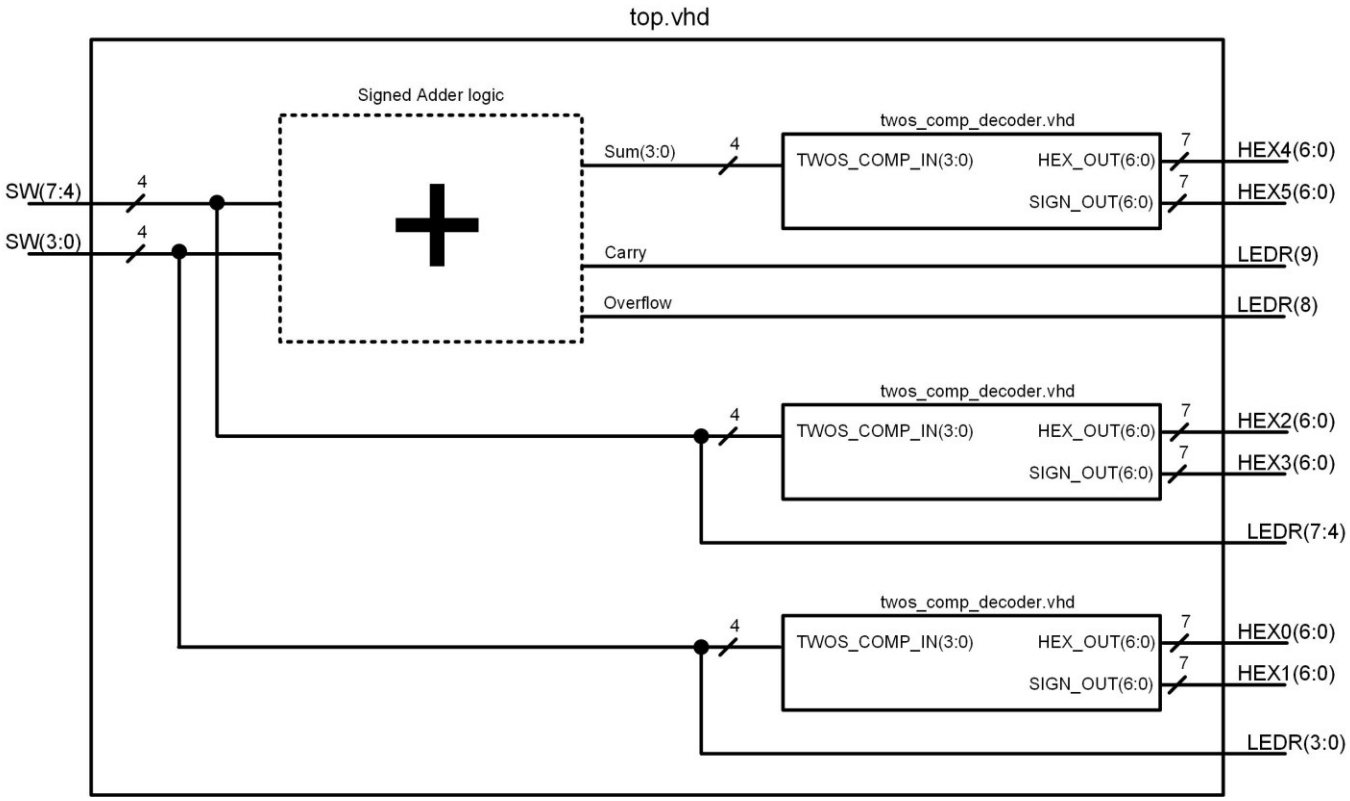
Deliverables

The deliverable(s) for this lab are as follows:

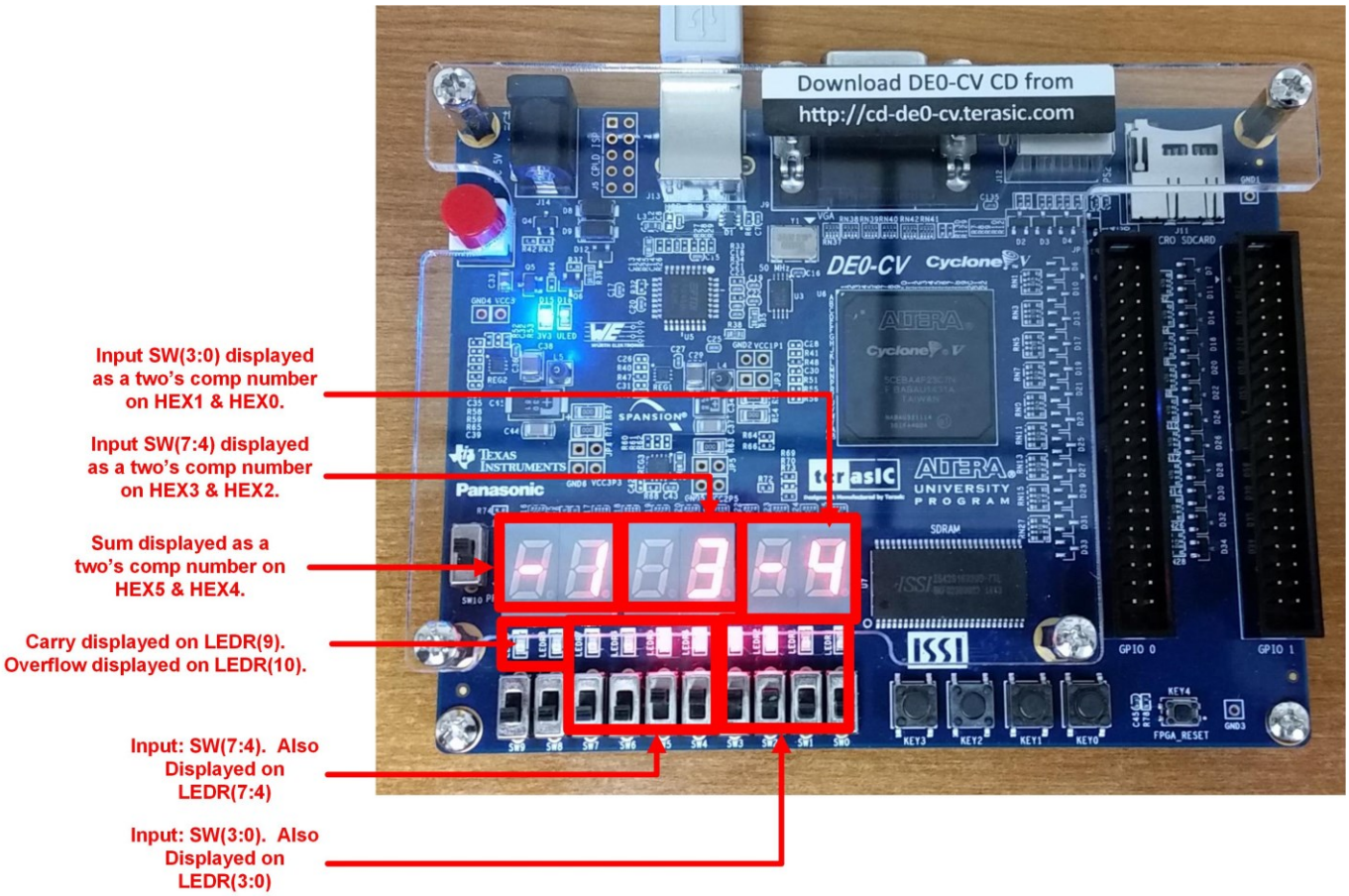
- Demonstration of your signed adder implemented on the DE0-CV board (90%).
- Upload your top.vhd file to the lab DropBox (10%).

Lab Work & Demonstration

You are going to create a 4-bit, signed adder. The following figure shows the block diagram for your system.



The following figure shows the DE0-CV I/O that will be used in this lab.



The signed numbers will use two's complement encoding. Recall that adding two's complement numbers is identical to adding two unsigned numbers. The main difference is how we interpret and display the numbers. You are going to first create a **new decoder** component called *twos_comp_decocer.vhd*. This decoder will take in a 4-bit two's complement number and drive two HEX displays. One display will show the sign of the number by either lighting segment 6 to indicate a negative number or not lighting any segment to indicate a positive number. The other display will show the magnitude of the two's complement number. Note that a 4-bit two's complement number can represent the decimal set {-8, -7, -6, -5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5, 6, 7}. Your subsystem should have the following entity definition:

```
entity twos_comp_decoder is
  port (TWOS_COMP_IN : in  std_logic_vector(3 downto 0);
        HEX_OUT      : out std_logic_vector(6 downto 0);
        SIGN_OUT     : out std_logic_vector(6 downto 0));
end entity;
```

The two inputs for your signed adder will come from the SW(7:4) and SW(3:0) switches. You will display the decoded version of the inputs on the HEX3/2 and HEX1/0 displays. You'll also display the inputs on the LEDR(7:4) and LEDR(3:0). You will display the decoded version of your sum on the HEX5/4 displays.

Your system will also calculate whether a carry and two's complement overflow occured. The Carry signal will be displayed on LEDR(9) and the Overflow signal will be displayed on LEDR(8).

Hint 1: You can create the adder logic using either a process or a sequential signal assignment. You will want to use the "+" to implement your adder, so you will need to include the numeric_std package. Note that this package only supports the "+" operation on types *unsigned* or *signed*. This means you'll need to create internal vectors of

type *signed* to implement the inputs and outputs of your adder. You will first need to type cast your inputs (SW(7:4) and SW(3:0)) into your new signed vectors. You can then perform the addition with a resulting sum and carry that are of type signed. You'll then need to type cast back from signed to std_logic_vector to drive the HEX and LEDR outputs.

Hint 2: The conditions for a two's complement overflow during addition is whether:

1) The sum of two positive numbers yields a negative number

or

2) The sum of two negative numbers yields a positive number.

The logic for this can be implemented using a process and an if/then construct. Recall that you can check whether a number is positive or negative by looking at its *sign bit* (i.e., the MSB bit).

A) Design and compile your system. Fix any Errors.

B) Download and test your design. Fix any Errors.

C) Demonstrate your design.

DEMO – Show the lab instructor your system. Once checked off, upload your top.vhd file to the lab DropBox.

Lab Grading

Demo 1 – Proper operation of your adder		_____ **/ 90**
Review of your top.vhd file		_____ **/ 10** (uploaded to DropBox)
	Total	_____ **/ 100**

Post Lab Survey (Just for your own knowledge. This is not graded)

1) Can you create a SIGNED adder using the "+" operator from the numeric_std package?
2) Can you use type casting to move back and forth between signed and std_logic_vector types?
3) Can you compute whether two's complement overflow occurred during an addition?