Laboratory Assignment #2

Due: 01/12/2020, 23:59

Trade-offs in Combinational Circuit Design: 15-bit Adder-Subtractor with Overflow Detector Circuit

Requirements:

You are required to design a circuit that can both add and subtract two signed 15-bit integers and realize whether there is an overflow or not. You will design two different adder-subtractors and compare their performances in terms of propagation time and area requirements. You should design and implement your adder-subtractor circuits on the schematic, simulate the designs and verify their correctness using Xilinx ISE tool. Finally, you should write a report for your lab. These two adder-subtractors are as follows (you can create two different projects for two adder-subtractor designs):

1. 15-bit ripple-carry adder-subtractor using full adders
2. 15-bit hybrid adder-subtractor using five 3-bit carry lookahead adders (CLAs).

Both adder-subtractors must detect overflow.

You are also required to use hierarchical design method for these adder-subtractors, in which small building blocks are used to construct the entire circuitry. For example, to design an 15-bit ripple-carry adder-subtractor, you first design a full-adder, then turn it to a symbol with three-bit input and two-bit output, and finally duplicate the symbol as many times as needed (For hierarchical design, see Xilinx ISE Design Tool Guide (Schematic) page 58-68).

You are also required to report the performance (area and speed) of your designs. After you finish your design (and verify its correctness), you must synthesize and implement your design for BASYS FPGA board (Spartan3E, XC3S100E, TQ144, -4 – see page 7 of Xilinx ISE Design Tool Guide (Schematic)). Then, you should include your implementation results in your report in terms of *time and area*. The time will be in terms of nano-seconds and the area will be expressed as amount of resources on the FPGA device (i.e., number of LUTs). For reading synthesize and implementation results, see the document Xilinx ISE Design Tool Guide (Synthesis-Implementation) on SUCourse+. Finally, compare two designs by answering following questions in your report using implementation results:

1. Which one of the two is better in terms of area?
2. Which one of the two is better in terms of time?
3. Define a new metric to measure the time-area tradeoff in two designs by multiplying the number of LUTs and time. Which one of the two designs is better in terms of this new metric?
4. State the requirements of a *good* design in terms of area, time and the new metric you've defined.

You are also required to write a report which should include (see Sample Lab Report on SUCourse+):

1. Screenshots of the schematics of your designs (both created symbol and top design)
2. For each design, include the screenshots of your simulation waveform window showing that your design is calculating correct output (for at least 4 different input combinations). For example:
    - Input A: 784, Input B: 53, Operation: Addition → Output C: 837, Output Overflow: 0
    - Input A: -4, Input B: -70, Operation: Subtraction → Output C: 66, Output Overflow: 0
    - ...
3. Include your implementation results and answer the questions above.

The following summarizes what you should do before and during the lab session.

Before the deadline

1. Work on your designs.
2. Enter your designs using schematic editor.
3. Simulate your design using as many as input combinations you can to make sure that your design is working correctly. For better visualization, you can group the inputs and represent them as signed decimal numbers.
4. Synthesize and implement your designs.
5. Write your report.
6. Submit your work (zip your project directory and your report), as much as you've done, through SUCourse+ before the deadline.

During the demo

7. Simulate your designs for the input combinations given by your TA.
8. Demonstrate Step 7 for the TA.

Notes:
In order to save time and avoid delays on demo schedule, please use the following input and output port names in your designs.

Inputs:
For the augend: A14, A13, A12, ... , A2, A1, A0 (A14 is the MSB,A0 is the LSB)
For the addend: B14, B13, B12, ... , B2, B1, B0 (B14 is the MSB,B0 is the LSB)
For addition/subtraction input: SUB

Outputs:
For the result: C14, C13, C12, ... , C2, C1, C0 (C14 is the MSB,C0 is the LSB)
For the overflow: OVF