Digital Logic
CSC 244L
Laboratory 6
Introduction to Addition and Subtraction Hardware

1 Objectives

1. Experimentally verify that the hardware used for "unsigned binary" addition can be used in 2's complement addition;
2. Experimentally verify that a simple "adder" circuit can be modified to perform subtraction; and
3. Design and verify a SystemVerilog version of the adder/subractor circuit using your DE10-Lite board.

2 Pre-Laboratory Procedure

Illegible work and files that do not open in the D2L dropbox will result in reduced grades. It is your responsibility to ensure that what you turn in is readable by the TA. Please read the submission instructions and follow them to ensure you receive all points. The circuit diagrams for pre-lab may be *neatly* hand drawn. To be completed *before* your lab meets (individually):

2.1 Read the entire lab procedure, and Chapters 1.5, and 5.1–5.2 (through subtraction, skipping carry-lookahead and prefix adders) in your book;

2.2 The following will be turned in, individually, to a D2L dropbox in your lab section before 2 pm on The week of October 24. Complete these items (in this lab manual) for your pre-lab:

- 4.1, 4.2, 4.3
- 5.1, 5.2, 5.3

2.3 Submit your pre-lab work to the Laboratory 6 dropbox in your CSC 244L lab section. **Clearly label your pre-lab work using the numbering convention in 2.2.**

For your SV, you must use **separate modules** with **one .txt file per SystemVerilog module** for the various logic functions and memory elements. **Do not .zip your pre-lab files, and do not turn in your SV in '.sv' file format.** You must turn these in as a '.txt' file to receive pre-lab credit, '.sv' files are not readable in D2L.

By now you should have downloaded and installed Quartus Prime Lite. You should come to lab with **SystemVerilog modules that compile the very first time**. You will ensure that your files compile correctly by compiling them at home with Quartus. A proactive student would also test that the compiled SV works on their DE10-Lite board.

3 7-segment Display and 2's Complement Numbers

3.1 Create a SV file named "decimal7decoder.sv" that contains one SV `module` named decimal7decoder. Using procedural SV (i.e., `always_comb`), create a 7-segment decoder that

takes a 4-bit 2's complement number as an input, and outputs the full range *in decimal* from -8 to +7 on an active-low 7-segment display. You will need to use HEX digits for this, one for the numeral and one for the sign.

3.2 Connect the 4-bit input to SW[3:0], and the 7-segment outputs to HEX1 and HEX0. Test all 16 combinations of inputs and verify the correct hexadecimal output. Be sure to double-check -8!

3.3 Show your working FPGA implementation to your TA, and have them sign off on your lab sheet.

4 Combinational Adder Procedure

Table 1: 2's-complement addition

A		B		Pre-lab Sum A+B				DE10 Sum A+B			
decimal	binary	decimal	binary	V	C	binary	decimal	V	C	binary	decimal
1	0001	1	0001	No	0	0010	2				
3	0011	1	0001	0	0	0100	4				
0	0000	-8	1000			1000	-8				
-4	0100	5	0101	0		0001	1				
6	1010	2	0010	1		1000	8				
-8	1000	7	0111	0		1111	-1				
-6	0110	-2	0010			1000	-8				

4.1 As part of pre-lab, complete the "pre-lab" column of Table 1. Use the "V" column to indicate whether or not *overflow* occurred, and use the "C" column to indicate whether or not there was a carry-out in the most significant bit.

4.2 Create a SV file named "fulladder.sv" that contains one SV `module` named fulladder. Using any SV you wish (e.g., behavioral, procedural), create a 1-bit full adder that implements $C_{in} + A + B$ and provides the answer as a sum bit S and carry-out C_{out}.

4.3 Create a second SV file named "adder4.sv" that contains one SV `module` named adder4. Using ***structural*** SV, connect four fulladder modules together to make a 4-bit ripple-carry adder (RCA). The RCA should replicate the 74x83, except rename the sum output Σ as S. The RCA should have two 4-bit addend inputs ($A_{3:0}, B_{3:0}$), a carry-in input (C_0), a 4-bit sum output ($S_{3:0}$), and two outputs to indicate whether a CARRY-OUT (C_4) or OVERFLOW (V) occurred.

4.4 Create a top-level SV file. Connect your A and B inputs to SW[7:4] and SW[3:0], respectively, as well as through your decimal7decoder decoder to HEX5:4 and HEX3:2, respectively. Connect your S outputs to LEDR[3:0] and HEX1:0, and your C_4 and V outputs to LEDR[9] and LEDR[8], respectively. The carry-in, C_0, should have a logic-0 as the input.

4.5 Connect the 74x83 on the breadboard. Use switches as your A and B inputs and ground your C_0 input. Connect LEDs to your sum and C_4 outputs. Use this adder circuit to determine the

values required to complete Table 1. Verify that the experimental data matches the pre-lab. If they do not, either debug your circuit, or recalculate your pre-lab (or both!).

4.6 Load your top-level SV module to your FPGA. Verify the operation of your SV module by checking that the outputs match your pre-lab and breadboard circuit in Table 1.

4.7 Show your working FPGA implementation and table to your TA, and have them sign off on your lab sheet.

5 Combinational Subtraction Procedure

Table 2: 2's-complement subtraction

A		B		Pre-lab Difference A-B				DE10 Difference A-B			
decimal	binary	decimal	binary	V	C	binary	decimal	V	C	binary	decimal
1	0001	1	0001	No	0	0000	0				
3	0011	1	0001	0			2				
0	0000	-8	1000	1			8				
-4	0100	5	0101	1			-9				
6	1010	2	0010	0			4				
-8	1000	7	0111	1			-16				
-6	0110	-2	0010	0			-4				

5.1 As part of pre-lab, complete the "pre-lab" column of Table 2.

5.2 Create a SV file named "addsub4.sv" that contains one SV `module` named addsub4. Use your adder4 module and whatever other SV you wish to create a circuit that performs both addition and subtraction based on a control signal (active-high subtraction, active-low addition). Your addsub4 should have two additional outputs that indicate whether a CARRY-OUT (C) or OVERFLOW (V) occurred.

5.3 Make a copy of your top-level SV file from the previous section. Replace the adder4 module with the addsub4 module, and connect the add/sub control signal to SW[9]. You may use this module to demo the previous section, with the control signal in ADD mode.

5.4 Using your already loaded top-level SV module on your FPGA, verify the operation of your SV module by changing the control signal to subtraction and checking that the outputs match your pre-lab in Table 2.

5.5 Show your working FPGA implementation, and table to your TA, and have them sign off on your lab sheet.

Laboratory 6 Signoff Sheet

TA Signature	Section
	3.3
	4.7
	5.5