Introduction to Digital Systems Laboratory 4b

adder / subtractor

ABSTRACT

In this lab students will build an adder / subtractor circuit using an adder and a selective 2’s complement circuit.

Introduction

In this lab students will be building an 6-bit adder/subtractor with an overflow bit. Ten switches will be used to input two 5-bit unsigned numbers (A and B), and three selective 2's complement circuits to implement the following operations:

A + B, B - A, A - B, and -A - B

The resulting sum will be displayed in hexadecimal on three seven segment displays where the third display will just be a sign (nothing or -).

Theory

In order to properly perform subtraction or addition of negative numbers which may yield a negative result a system must have a method of representing negative numbers. In digital systems 2's complement format is the preferred method of representing negative numbers since it allows for simple addition of negative numbers. For example if the goal is to implement A – B it will be the same as A + (2's complement of B) provided the system ignores the overflow bit and has enough bits to perform the operation.

A 5-bit unsigned binary integer can represent decimal values between 0 and 31 where a 5-bit signed binary integer can represent decimal values between -16 and 15. Thus is it important when designing a digital system which performs operations on numbers to ensure there are enough bits to handle all input combinations or there is a method of detecting an overflow.

The introduction specifies the inputs are two 5-bit unsigned numbers which will represent decimal values from 0 to 31. Below is shown the minimum and maximum values for each of the operations to be implemented:

Operation	Operation with 2's Complement	Minimum	Maximum
A + B	A + B	0	62
B - A	(2's complement of A) + B	-31	31
A - B	A + (2's complement of B)	-31	31
-A - B	(2's comp of A) + (2's comp of B)	-62	0

Table 1 – List of operations

A 6-bit adder will be used in conjunction with 2's complementers to perform these operations which will result in a 6-bit signed output (see Table 1). A 6-bit signed binary number can represent decimal numbers from -32 to 31 thus there will be input cases where the output will overflow. This could be avoided by using a 7+ bit adder; however, the overflow is intentional in order for students to learn about overflow conditions.

Implementation

In order to complete this lab several Verilog modules must be created with the following specifications:

Selective 2's Complementer
There shall be a 6-bit input value and a 6-bit output value as well as a single select bit. When the select bit is 0 the output shall be equivalent to the input. When the select bit is 1 the output shall be equivalent to the 2's complement of the input. Since the output is specified to be 6-bits the C_6 overflow/carry bit can be ignored. If implementing with schematics it will be helpful to design a single bit selective 2's complementer first then use six copies to implement this block. You can use the technique of fixed input simplification. Since the goal is to output either the input, A, or -A which is A'+1 we can use an adder with the B input fixed to 0. When an input is fixed it allows for simplification of a circuit. For example, A0 = 0, A+0 = A, $A\oplus 0$ = A, A1 = A, A+1 = 1, and $A\oplus 1$ = A'. To select between A and A'+1 you can make a A'+1 circuit then use a multiplexer to select between A and A'+1 based on the select input bit.

Adder
There shall be two 6-bit input values, a carry input, a carry output, and a 6-bit output value equal to the binary sum of the inputs. An adder can be implemented in Verilog or the schematic Adder from part A of the lab can be extended then converted to Verilog automatically (File → Create / Update → Create HDL Design File from Current File… then select Verilog HDL for the language).

Seven Segment Hexadecimal Display Code Converter
There shall be a 4-bit input and a 7-bit seven segment output where the least significant bit (LSb) of the output will represent segment A and the most significant bit (MSb) will represent segment G.

Dual Seven Segment Hex Display
There shall be a 6-bit input and two 7-bit seven segment outputs. This can be implemented using two of the Seven Segment Hexadecimal Display Code Converters described above.

In order to implement all the operations a selective 2's complementer should be connected to each input. In order to display a useful value to the user another selective 2's complement should be connected to the output such that it complements the output when the output is negative. This can be achieved by connecting the MSb of the adder output to the select bit of the 2's complementer. The MSb of the adder can also be used to indicate when the value is negative.

If using the DE0 the 5-bit A input will be SW[4:0], the 5-bit B input will be SW[9:5], the select for the A 2's complementer will be BUTTON[0], the select for the B 2's complementer will be BUTTON[1]. If using the DE2 the select inputs will be KEY[0] and KEY[1] respectively. Figure 1 shows a schematic implementation of the top level block diagram; however, you are required to implement the top level using Verilog.

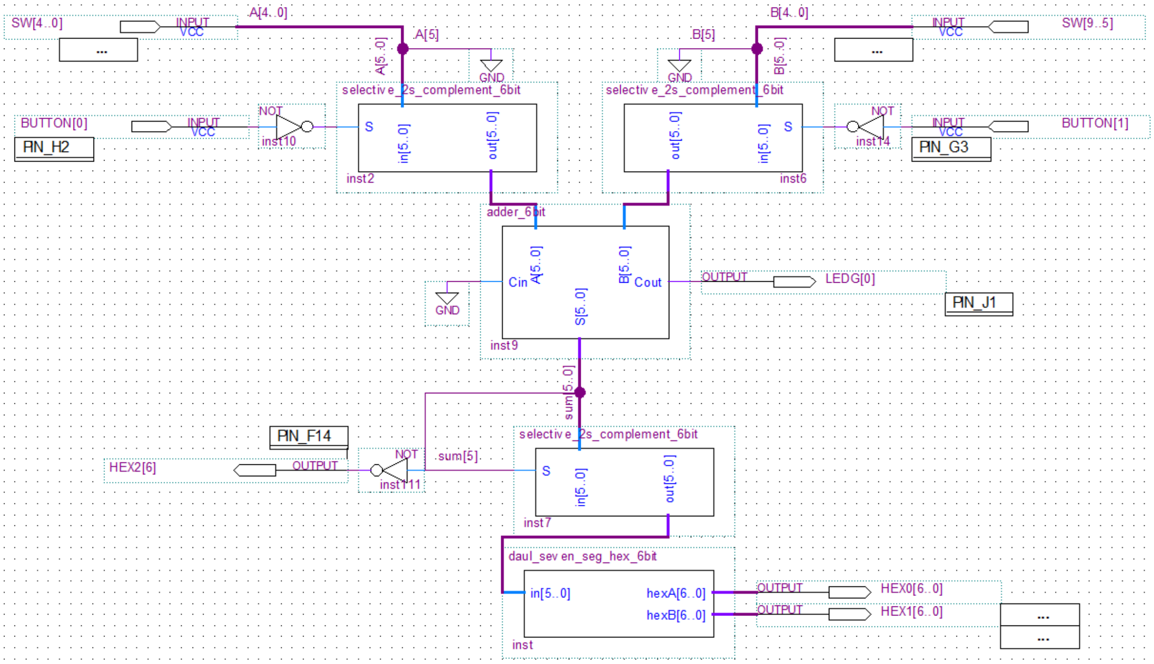

Figure 1 – Top level block diagram

Since the input buses are 5-bits and the input to the selective 2's complementer is 6-bits it is necessary to name the bus connecting to the switches (e.g. A[4..0]) and the bus connecting to the 2's complementer (e.g. A[5..0]) such that Quartus II will know how to connect the two buses of different widths together. A node should also be connected to the MSb of the 6-bit bus and connected to ground as shown in Figure 1. Note: since the BUTTON/KEY inputs are normally 1 and 0 when active inverters are needed on the BUTTON/KEY inputs to attain the desired functionality.

Extra Credit

(+1 points) When the output can be represented by one digit hide the second digit and use that position as the sign.

Deliverable

Demonstration of adder/subtractor functionality.

Grade Scale

Demonstration Shows:	Grade:
Nothing	0 / 10
Tried something	5 / 10
Functions with bugs	8 / 10
Functions properly	10 / 10