Lab 6 – Arithmetic Logic Unit

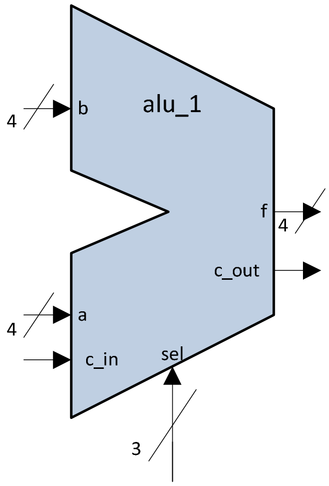
Introduction

In this lab you will be designing an Arithmetic Logic Unit (ALU). Its name is quite descriptive of what it is: it is *a block of logic that performs various arithmetic operations*. It is a central component of most all processor designs: large and small processors, digital signal processors, etc. We will use it later in the course as we develop a skeleton design of a small processor.

Your design will behave like a 4-bit calculator with a small set of arithmetic operations. You will select two 4-bit values on switches 0-7. Switch 8 will be a carry in/borrow in signal. The two push buttons (key0 and key1) and switch 9 and will select an arithmetic function as per the specification. The resultant 4-bit value will be displayed on one of your seven segment displays. The carry out result will be displayed on an LED.

Detailed Specification

The design specification centers on the particular arithmetic functions of the ALU. A block diagram of the module ***alu_1*** is shown below. Note the two 4-bit input signals, ***a*** and ***b***, and the 4-bit output signal ***f***. There are also single bit carry in/borrow in (***c_in***) and carry out/borrow out (***c_out***) signals. Seven arithmetic operations for this lab are described in the table below.



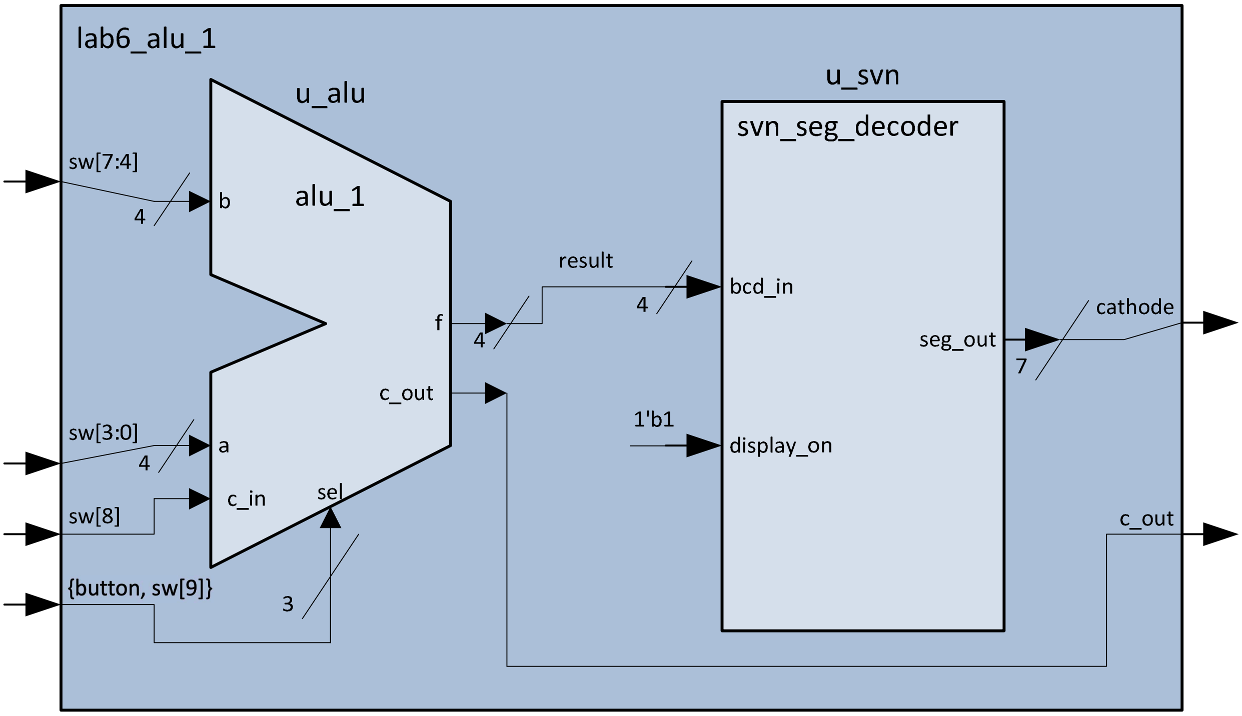
Select Operation (signal: sel)	Arithmetic Operation
3'b000	f = b with c_out = 0
3'b001	f = a + b + c_in with c_out being set
3'b010	f = a + b with c_out being set
3'b011	f = a - b - c_in with c_out being set
3'b100	f = a - b with c_out being set
3'b101	f = a + 1 with c_out being set
3'b110	f = a - 1 with c_out being set
3'b111	f = a & b with c_out = 0

Design Implementation

There are two design steps to complete this lab.

- Design an ***alu_1*** module as per the specification above. You will be provided skeleton code in file: **alu_1.sv**. The preferred implementation is to use a case statement to decode the select signal.

- In the ***lab_6_alu_1*** module, you will instantiate the ***alu_1*** module and your seven-segment display driver from Lab 3. You will connect them so that the output from the ALU is connected to the input of the display driver. Thus, when implemented, you will see the output of the ALU on your seven-segment display. A block diagram of the ***lab_6_alu_1*** module is shown below. The carry out signal is connected to LED 8.



Simulation/Verification

There are two simulation sets that match the two design steps above.

There are two simulation command files (right-click each of the *.m_sim files and Run) associated with each of the testbenches:

- **alu_1.m_sim**, to simulate and test ALU module alone. The testbench reads the text file, **tb_alu_1.txt**, which tests all 2048 combinations of the input buttons and switches.
- **lab6_alu_1.m_sim**, to simulate and test the integration of the ALU and seven segment decoder into module lab6_alu_1. It uses the same test vectors as the first simulation.

Remember to save a screen shot of each simulation log file for your reports. There will be two different simulation log files you will need.

There are two different ***vcd*** waveform files that are generated from each simulation.

- **alu_1.vcd**

- **lab6_alu_1.vcd**

Download and view these with ***WaveTrace*** as needed to debug your design.

Synthesis

Generate a binary file as you did in past labs (right-click and Run on: **lab6_top.qsf**). For this lab it will be called: **lab6_top.sof**. Copy it to your shared folder to load and run on your DE10 board. Try several combinations of button/switch selections and verify your observations based on the design specification.