ECE438 Advanced Logic Design

30 September 2013

**Goal: Create A 32-bit ALU**

For this project I was asked to create a 32-bit ALU that would have nine functions. These nine functions would be controlled by four bits. With this information I decided to use signals to decide which vector I would output. For example when the control bits were in the pattern for an XOR, I would have an internal signal assert an XOR line high, otherwise it would be set low. Once I had this I could simple use a conditional signal assignment statement to select which vector I was outputting. Seven of the nine operations were very easy. The hard part was constructing a 32-bit Look Ahead Carry Adder. Next was constructing a Set On Less Than function. This too was challenging but for different reasons.

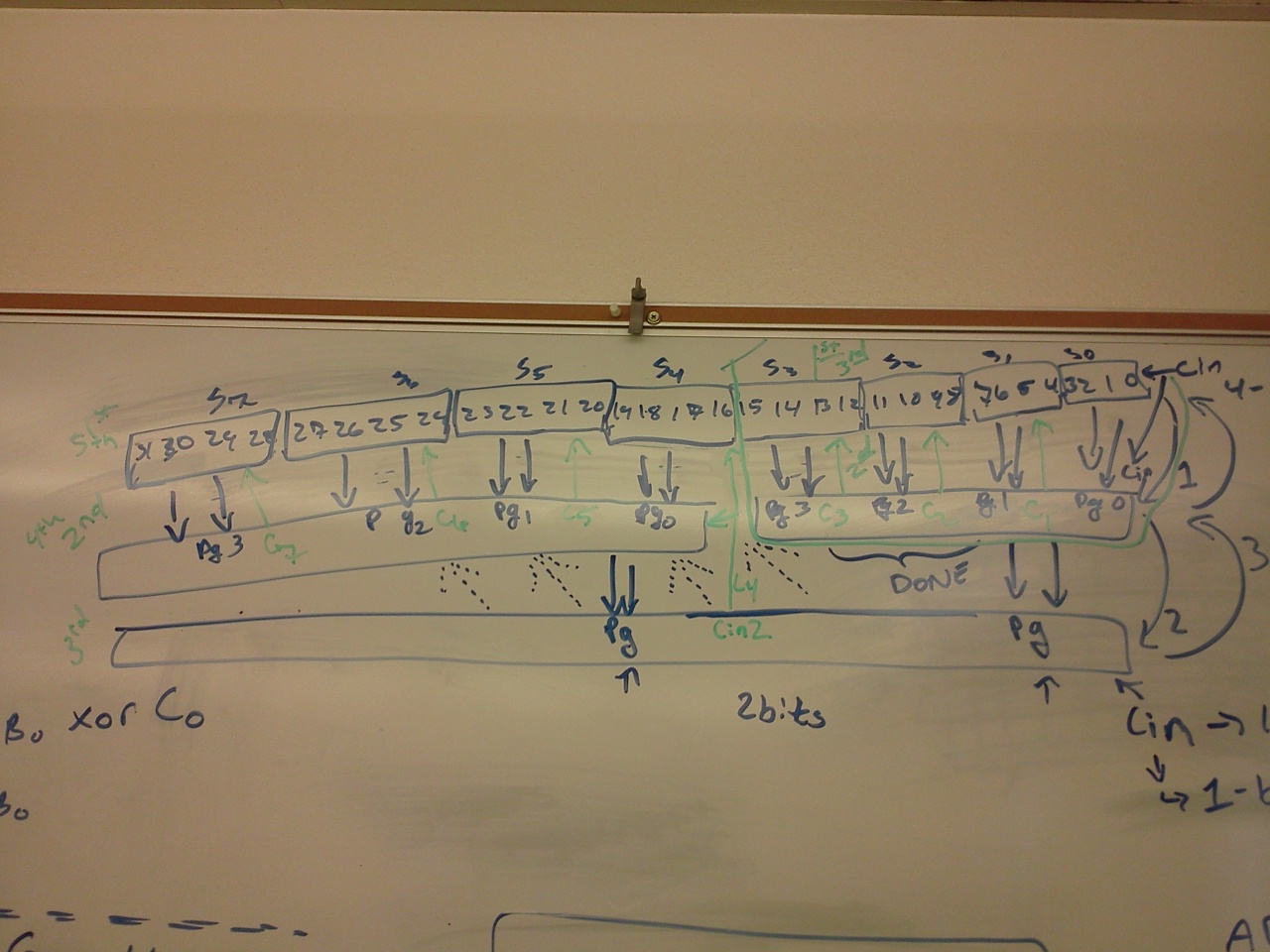
**Results:**

My results were overall very well. There are a few outlying cases when my SLT function is incorrect, but given the time to complete this I feel that this is an acceptable loss. The adder/subtraction module works exactly as it was modeled as well. All the other functions are extremely easy to create, given the boundaries that it was to be done bitwise instead of the entire vector at once. Timing for this is completely dependent upon the gate delay, for our simulation we used 1 nano second. This is reasonable for all of the functions except for the adder/subtracter module. This approximation doesn’t work because we used very wide gates to get our 1 ns gate delay. This would be extremely expensive to actually implement, and so our adding/subtracting time would be greatly extended if we were to try to make this ALU cost effective. The entire adding module is constructed with single bit full adders and propagate generate units connected to those and more to those that are connected to the full adders. This method cuts down on time that would have been spent on a ripple carry adder. In order to combine addition and subtraction, when my subtraction line was asserted, I inverted the subtraction and forced carry in to be a one, this adding the twos complement and getting the subtraction answer from an addition answer.

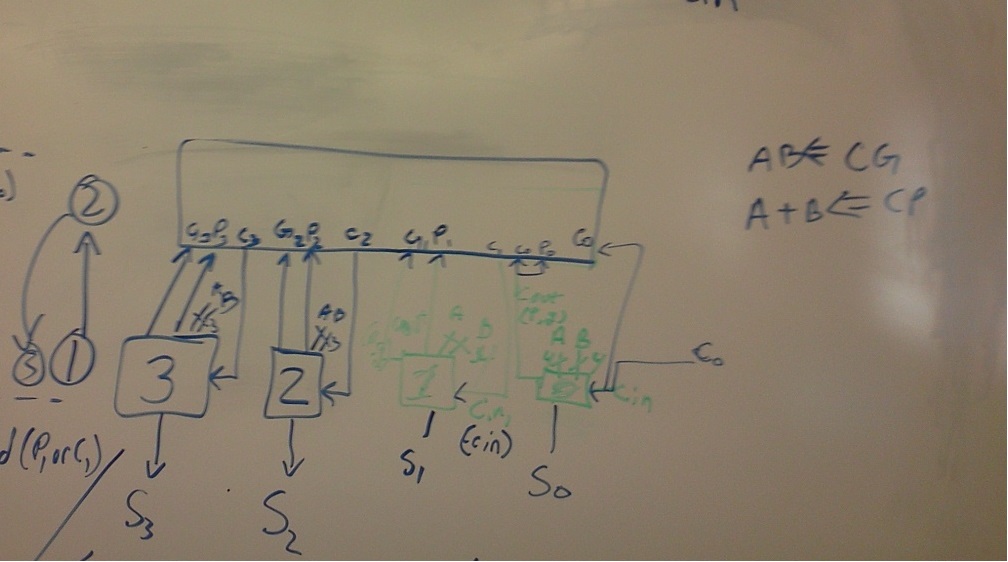
**Final Thoughts:**

This project was not very hard, until I had to construct the adder module. However, once I realized that I could construct all of the adders at the same time and that the signals would be connected in the way that I desired them to be, it became incredibly easy. This thought came to me when I realized that the carry propagate units were essentially registers so to speak, and that the carries were being calculated inside of them and then spit out to the adders after the adders constructed the propagate and generate values.

Here is a block diagram of what my adder module looks like conceptually:



There are seven LACU’s that all feed into two, then those two feed into one. This big one then cascade’s the carries into the full adders to get the sum.

Here is how the full adders interact with the LACU’s and vice versa:  
  


As shown here the LCAU calculates the carries for each adder and then pushes them out back to the adder. The adders first must calculate the propagate and generate.