Andrew Sandberg

As3264

3410 Project 1

Comparators:

For the comparators I broke it up into two branches, one where bit 2 of Op was a 0, and one where it was a 1. The 0 corresponds to less than or equal to or greater than and the one corresponds to equal or not equal. Then each branch had another multiplexer that selected, using bit one, which of the functions it was. To determine equality I simply ran both inputs through an xor gate and then inverted the result. My logic was if the bits were different the xor would return one which then inverted would be a zero. In the opposite case if both bits were the same then the output would be one. I then took all new 32 bits, split them, and ran them through an and because if they were then all one I knew the numbers were then equal as every bit pair would have been the same and resulted in a 0 through the xor and then a one through the inverter. Then I just inverted the final result and put it into the multiplexor so I could choose between equal and not equal.

The other half of the circuit I checked whether or not the numbers were less than or equal to zero, or greater than zero. To do this I ran every bit in A but the most significant bit through an or gate. Then I knew if there were any ones I would now have a 1, then I compared this new bit to the most significant bit and drew a truth table. There were three cases were the number was less than or equal to zero; letting the processed bit be called p they were, (p and MSB) +(!p and MSB) +(!p and !MSB). With this I was able to construct a simple circuit with three and gates and an or gate. Then I ran this result to the multiplexer and ran its inverse to the multiplexer so I had both options.

Logicals:

The logicals were the easiest. All I needed was one of each type of gate and to split the results between three multiplexers that would select the result I wanted in accordance with the operator. Bit 1 decided between and and or, as well as between nor and xor. Then bit 2 decided between the two groups.

Shifter:

For this I was luckily able to use my shifter from class with some modifications. I built an inverter circuit that just made the first bit the last and the last the first and so on because I knew that shifting to the right was the same as shifting the reverse of the input to the left, and then flipping the order again. With that knowledge, I needed a mux to choose between the reverse of the input and the normal order, a mux to choose between logical and arithmetic, and a mux to choose between reverse input and normal order at the end (because I had to flip the result after passing it through the shifter but if it had been left shift originally, I did not want the flipped result).

Adder:

Again, I was mostly able to reuse my adder from lab but with a few caveats. I needed a carryout and I needed to be able to also do subtraction. So I decided if bit 2 was 1 then the operation was subtraction and I wanted the inverse of B and for Cin to be 1, so I used a mux and simply had bit 2 be passed as Cin. To fix the problem of the carryout, I added a carryout to the four bit adder, then in the sixteen bit adder I only used the carryout of the last four bit adder in the circuit, and repeated. The logic was that I only cared whether or not the Cin and the Cout at the position of the most significant bit were different. If they were then I knew there was overflow.