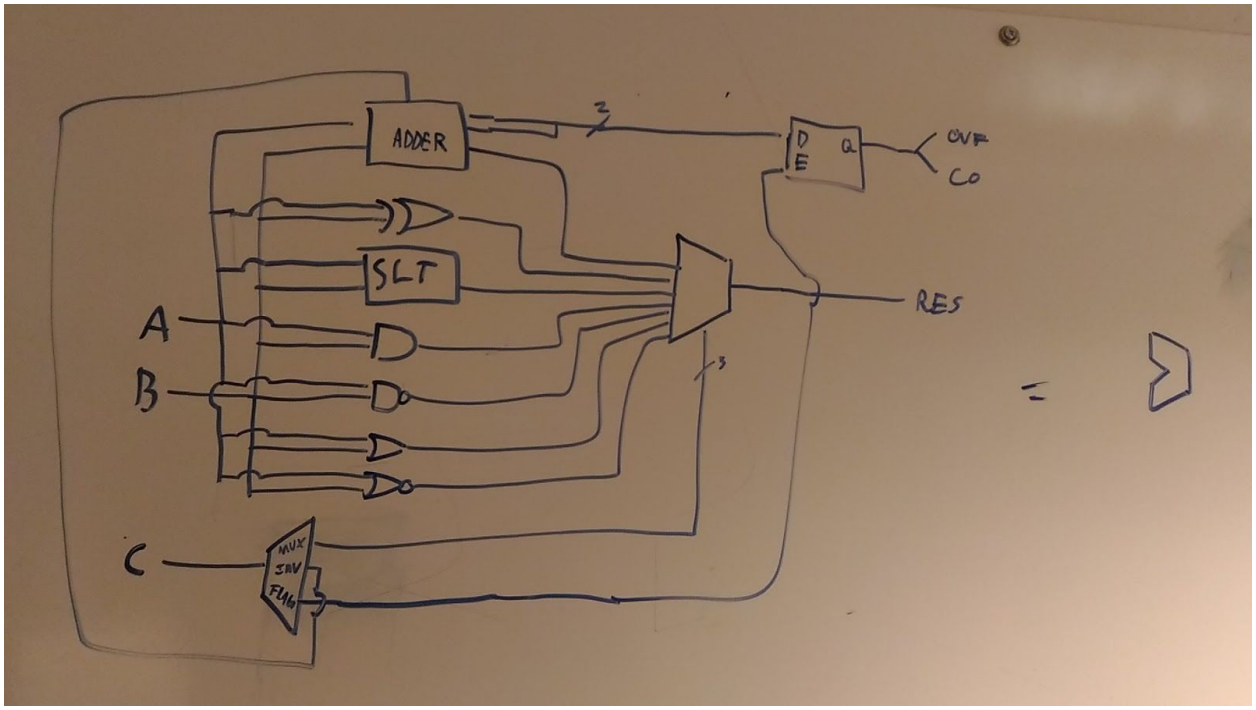
CompArch Lab 1

Ian Paul
Shruti Iyer

Implementation



We based our designs off the suggestions in the lab. We used a LUT to choose between the operations. We also calculated results for all the operations and then used an 8 input multiplexer (actually used 7 of those inputs) to choose the final answer for the output of the ALU. We used a bit-slice and control style, computing the operations and then combining the relevant ones at the end. One inopportune design choice we made was using behavioral verilog instead of a multiplexer to switch between addition and subtraction in our adder. We will likely switch this in an update.

Test Results

CompArch Lab 1

Ian Paul
Shruti Iyer

For our addition and subtraction model we tested adding and subtracting such that we would overflow high and low, sum to zero, and get precise values for each bit. These tests span the full range of what this operation should do, including verifying the flags. Also, it is worth noting that we tested on other values, but only kept a set of relevant ones in the test bench.

We tested all of the logic operations using the same values: both inputs all high, both inputs all low, one input high, one input low, and a mix of high and low in different places for both inputs. These tests should check that the operation works over its entire range and that it can process different kinds of inputs to both positions.

To test the SLT we only chose 3 test cases because it heavily reuses the adder and subtractor code which is heavily tested. We used a case where a<b, a=b, and a>b. Assuming that the subtractor worked well, these would test the full range of what the SLT needed to do.

We found 2 cases where the tests revealed flaws. The first was when testing the adder, we noticed that it failed on summing to 0. This turned out to be an error in our test bench rather than the operation. We used this as an opportunity to increase the delays on all our test benches. The second error was in XOR. We discovered that we wired one of the gates wrong and changed it.

Overall, we found that the test cases we thought of in our initial meeting were sufficient. In fact, we initially overestimated the number of test cases we needed, and chose to use

fewer to no redundant cases. We didn't find a case where we needed to add tests, just change them as our ALU structure changed.

Timing Analysis

We analyzed our ALU and found that the worst case delay comes from any case where we need to use the SLT. We calculated that it takes 1670 nanoseconds (with a 10 nanosecond estimate for fundamental gates) for the longest delay to propagate through the SLT. Since the ALU control was done using behavioral verilog, we don't have a timing estimate for how long changes there take, but our 8 input multiplexer(made of smaller multiplexers), takes 150 nanoseconds at worst to propagate. Overall, the delay in our ALU is 1820 nanoseconds.

Work Plan Reflection

Compare how long each unit work actually took to how long you predicted it would take. This will help you better schedule future labs.

Most of the work estimate happened as expected except for Fixing Code (i.e. Integration of various parts). This took slightly longer, but mostly because other class workload occupying way more time. We underestimated the work/time that goes into putting together different parts of the ALU.

CompArch Lab 1

Ian Paul
Shruti Iyer

Thing	Time Estimate	Due	Actual Time
Finish Plan:	30 Minutes	2016-10-01	30 mins
Control Logic:	30 minutes	2016-10-01	30 mins
Test Cases:	2 Hours	2016-10-04	1.5 hours
Functions:	2 Hours	2016-10-04	2.5 hours
Fixing Code:	2 Hours	2016-10-05	4 hours
Writing the Report:	90 Minutes	2016-10-06	30 mins