Alex Kahng, George Zhang

CS 141

Simulation Description (Lab 2: Part 1)

**Implementation**

For each of the bitwise operators AND, OR, and XOR, we implemented a module by using the analogous bitwise operator in Verilog. For NOR, we implemented a module by negating the OR output. For ADD, we created a one-bit full adder in a separate module and chained these modules together to create a ripple-carry adder. For the multiplexer (mux), we nested ternary operations that called the appropriate operation (module) depending on the given op\_code.

**Testing**

For each of the bitwise operators, we set X, Y, to all permutations of 0x00000000 and 0xFFFFFFFF. This allowed us to test the operation of each bit and make sure our bitwise operators worked correctly. We also tested over random input to further validate beyond the corner cases. To test the EQUAL and ZERO flags, we checked if X = Y or if Z = 0 and made sure the appropriate flags were set. To test ADD, we wrote a one-bit full adder test and an adder test, both of which we did not include in test\_alu.v. In test\_alu.v, we created a test to make sure that the full adders were correctly hooked up. Specifically, we repeatedly bit-shifted X and Y, added them, and verified that the next significant bit changed correctly due to the carry bit. We tested our mux by varying the op\_codes.