Mark McMurtury and Cody Butler
Lab 3

Description:

We were tasked with creating an RPN Calculator in System Verilog using a stack and a register file. The register file should two 32 bit read ports and associated 5 bit read addresses, as well as a single 32 bit write port and a 5 bit write address. If the write signal is enabled, the program should read from either read address port and write that data to a specified write address. The RPN calculator should contain a stack implemented using our register file. The calculator should use switch values and key values to determine which operation code will be executed.

Partial Simulation Code:

```
...
#pushes values to sim
force mode 2'b00        //sets the mode
run                     //run for 1 cycle
run
force key 4'hf          //sets the key value to all ones to set our go signal
run
force key 4'b11         //sets the key value 1 and 1, execute push operation
run
run
force key 4'hf
run
force val 16'h2         //pushes a value of 2 onto the stack
run
force key 4'b11
run
run
force key 4'hf
force val 16'h3
run
run
force key 4'b11
run
force key 4'hf
force val 16'h4
run
run
force key 4'b11
run
force key 4'hf
force val 16'h6
run
force key 4'b11
run
force key 4'hf
force val 16'h2
run
```

```
…
#runs the operations
force mode 2'b10        //reset the mode for testing
run
run
force key 4'hf
run
run
run
force key 4'b1          //sets the key values to 0 and 1, execute NOR operation
run
force key 4'hf
run
run
run
force key 4'b10         //sets the key values to 1 and 0, execute OR operation
run
force key 4'hf
run
run
run
force key 4'b11         //sets the key values to 1 and 1, execute AND operation
run
force key 4'hf
run
run
run
force key 4'b0          //sets the key values to 0 and 0, execute XOR operation
run
run
run
… etc ...
```

Results and Discussion:

For test case 1, our mode was 2'b00 and our key value of 4'b11, which executes the push operation which pushes a value to the stack and increment our counter. We initially use this case to push our values onto the stack that we wish to test. We use a push bypass within our other commands to push computed values onto our stack, with the register write-back enabled.

Test case 2, our mode was 2'b00 and our key value of 4'b10, which executes the pop operation which pops a value from the stack and decrements the counter. We call this operation within our other commands to remove the values from our stack.

Test case 3, our mode was 2'b00 and our key value of 4'b01, which executes the add operation which pushes the sum value to the stack. Test case 4, our mode was 2'b00 and our key value of 4'b00, which executes the subtract operation which pushes the difference of the values to the stack. Test case 5, our mode was 2'b01 and our key value of 4'b11, which executes the unsigned multiply operation which pushes the lower 32 bit value to the stack. Test case 6, our mode was 2'b00 and our key value of 4'b10, which executes the shift left logical operation which pushes the topmost value that has been shifted to the left by the second value to the stack. Test case 7, our mode was 2'b00 and our key value of 4'b01, which executes the shift right logical operation which pushes the topmost value that has been

shifted to the right by the second value to the stack. Test case 8, our mode was 2’b00 and our key value of 4’b00, which executes the comparison operation which compares the topmost and second to top, determines if the topmost value is less than the second to top value, then pushes either 0 or 1 to the stack. Test case 9, our mode was 2’b00 and our key value of 4’b01, which executes the shift right logical operation which pushes the topmost value that has been shifted to the right by the second value to the stack. Test case 10, our mode was 2’b10 and our key value of 4’b11, which executes the bitwise AND operation which pushes the value that has been anded together to the stack. Test case 11, our mode was 2’b10 and our key value of 4’b10, which executes the bitwise Or operation which pushes the value that has been or-ed together to the stack. Test case 12, our mode was 2’b10 and our key value of 4’b01, which executes the bitwise NOR operation which pushes the value that has been nor-ed together to the stack. Test case 13, our mode was 2’b10 and our key value of 4’b00, which executes the bitwise XOR operation which pushes the value that has been xor-ed together to the stack. Test case 14, our mode was 2’b11 and our key value of 4’b11, which executes the reverse operation which pops the topmost values, swaps them using two temporary variables and stalls to hold the data, and push the resulting swap to the stack.