Lab 3 Report

Maheedhar Mandapati
Joanna John

12/7/2019

Testing Strategy

Our testing technique was to form a testbench that utilized pre-defined vectors to rapidly test each of the operations that our calculator has. Each vector had a mode, key, and a value, as well as anticipated values for the top and next on the stack. Our objective was to rapidly check that each operation worked as expected. The vectors are designed such that each one is 56 bits long, or 14 hex digits.

The primary hex digit can be further broken down into three parts, the highest bit being the enabled flag. The next bit is a don't care. And finally, the final two bits are taken together and utilized as the mode within the rpn calculator. The second hex digit is utilized as the key within the rpn calculator, where key is active low. The lowest 4 hex digits represent the expected next value and the 4 hex digits before that represent the expected top value. The remaining 4 hex digits in the vectors is the value to be pushed onto the stack.

After each vector that requires the testbench to perform an operation we tell the testbench to idle, so the rpn calculator can catchup and return the correct result when required.

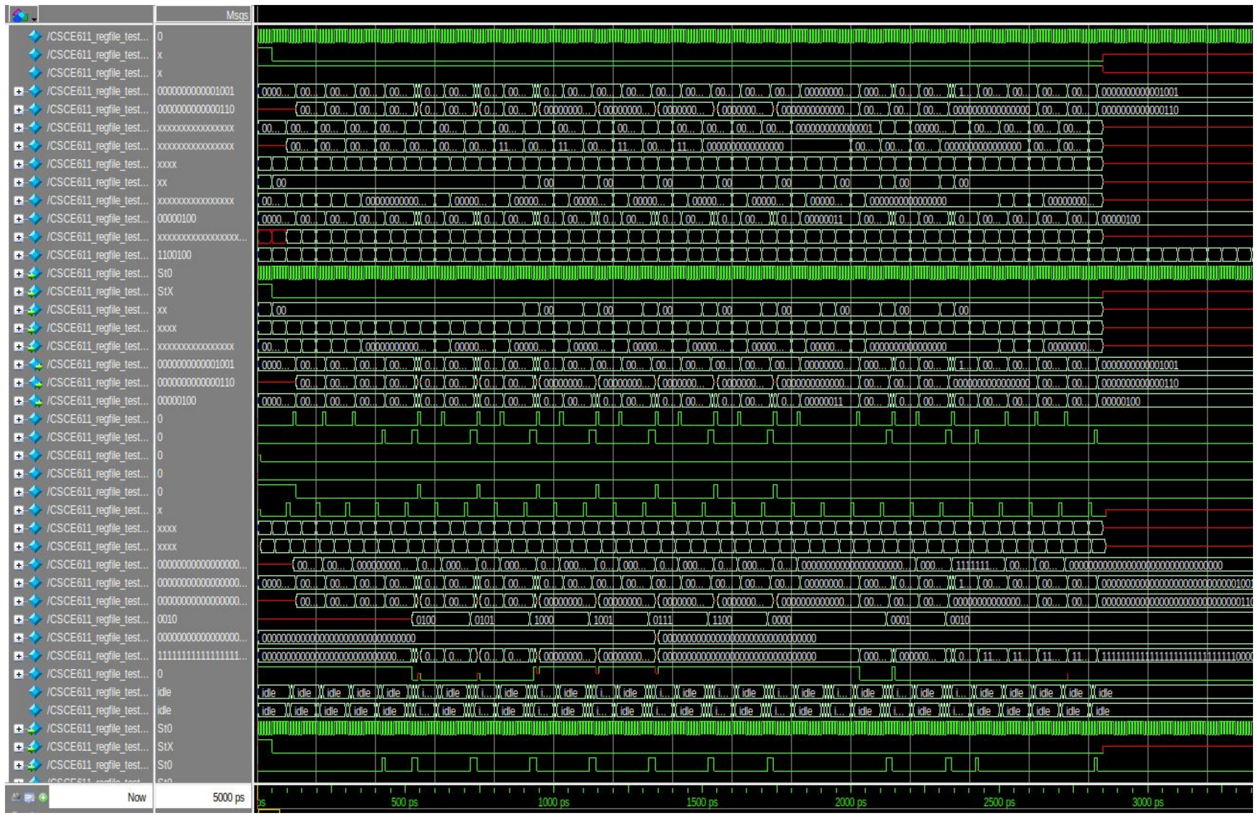
Justification/Analysis

The test vectors we created test all the operations of the rpn calculator, pushing, popping, and reverse. However when we ran the test vectors, we found that our add function didn't pop of enough values that it needed and that any vectors that followed the add one was throwing errors. We established that this is because of our decision to user multi state system in which you could differentiate between popping once/twice. We have tried to counteract this but to no avail.

Lab 3 Report

Maheedhar Mandapati
Joanna John

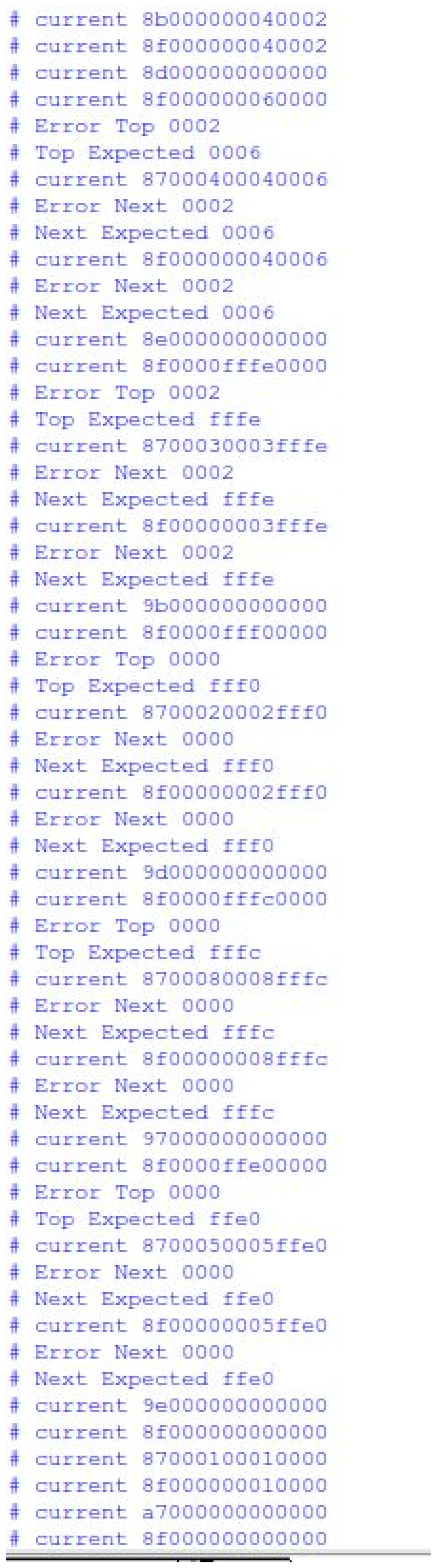
12/7/2019



Lab 3 Report

Maheedhar Mandapati
Joanna John

12/7/2019



```
# current 8b000000040002
# current 8f000000040002
# current 8d000000000000
# current 8f000000060000
# Error Top 0002
# Top Expected 0006
# current 87000400040006
# Error Next 0002
# Next Expected 0006
# current 8f000000040006
# Error Next 0002
# Next Expected 0006
# current 8e000000000000
# current 8f0000fffe0000
# Error Top 0002
# Top Expected fffe
# current 8700030003fffe
# Error Next 0002
# Next Expected fffe
# current 8f00000003fffe
# Error Next 0002
# Next Expected fffe
# current 9b000000000000
# current 8f0000fff00000
# Error Top 0000
# Top Expected fff0
# current 8700020002fff0
# Error Next 0000
# Next Expected fff0
# current 8f00000002fff0
# Error Next 0000
# Next Expected fff0
# current 9d000000000000
# current 8f0000fffc0000
# Error Top 0000
# Top Expected fffc
# current 8700080008fffc
# Error Next 0000
# Next Expected fffc
# current 8f00000008fffc
# Error Next 0000
# Next Expected fffc
# current 97000000000000
# current 8f0000ffe00000
# Error Top 0000
# Top Expected ffe0
# current 8700050005ffe0
# Error Next 0000
# Next Expected ffe0
# current 8f00000005ffe0
# Error Next 0000
# Next Expected ffe0
# current 9e000000000000
# current 8f000000000000
# current 87000100010000
# current 8f000000010000
# current a7000000000000
# current 8f000000000000
```

Lab 3 Report

Maheedhar Mandapati
Joanna John

12/7/2019

Test Vectors

```
//rst
b_d_0000_0000_xxxx
// idle
8_f_0000_0000_xxxx
//push
8_7_0002_0002_0000
// idle
8_f_0000_0002_0000
//push 2
8_7_0004_0004_0002
// idle
8_f_0000_0004_0002
//push 3
8_7_000a_000a_0004
// idle
8_f_0000_000a_0004
//pop
8_b_0000_0004_0002
// idle
8_f_0000_0004_0002
//add
8_d_0000_0000_0000
// idle
8_f_0000_0006_0000
//push 4
8_7_0004_0004_0006
// idle
8_f_0000_0004_0006
//sub
8_e_0000_0000_0000
// idle
8_f_0000_fffe_0000
//push 5
8_7_0003_0003_fffe
// idle
8_f_0000_0003_fffe
```

Lab 3 Report

Maheedhar Mandapati
Joanna John

12/7/2019

```
//sll
9_b_0000_0000_0000
// idle
8_f_0000_fff0_0000
//push 6
8_7_0002_0002_fff0
// idle
8_f_0000_0002_fff0
//srl
9_d_0000_0000_0000
// idle
8_f_0000_fffc_0000
//push 7
8_7_0008_0008_fffc
// idle
8_f_0000_0008_fffc
//multu
9_7_0000_0000_0000
// idle
8_f_0000_ffe0_0000
//push 8
8_7_0005_0005_ffe0
// idle
8_f_0000_0005_ffe0
//slt
9_e_0000_0000_0000
// idle
8_f_0000_0000_0000
//push 9
8_7_0001_0001_0000
// idle
8_f_0000_0001_0000
//AND
a_7_0000_0000_0000
// idle
8_f_0000_0000_0000
//push 10
8_7_0001_0001_0000
// idle
```

Lab 3 Report

Maheedhar Mandapati
Joanna John

12/7/2019

```
8_f_0000_0001_0000
//OR
a_b_0000_0001_0000
// idle
8_f_0000_0001_0000
//push 11
8_7_0001_0001_0001
// idle
8_f_0000_0001_0001
//NOR
a_d_0000_0000_0000
// idle
8_f_0000_0001_0000
//push 12
8_7_0000_0000_0001
// idle
8_f_0000_0000_0001
//XOR
a_e_0000_0000_0000
// idle
8_f_0000_fffe_0000
//pop
8_b_0000_0000_0000
// idle
8_f_0000_0000_0000
//push 13
8_7_0006_0006_0000
// idle
8_f_0000_0006_0000
//push 14
8_7_0009_0009_0006
// idle
8_f_0000_0009_0006
//reverse
c_7_0000_0000_0009
// idle
8_f_0000_0000_0009
//pop
8_b_0000_0009_0006
```

Lab 3 Report

Maheedhar Mandapati
Joanna John

12/7/2019
// idle