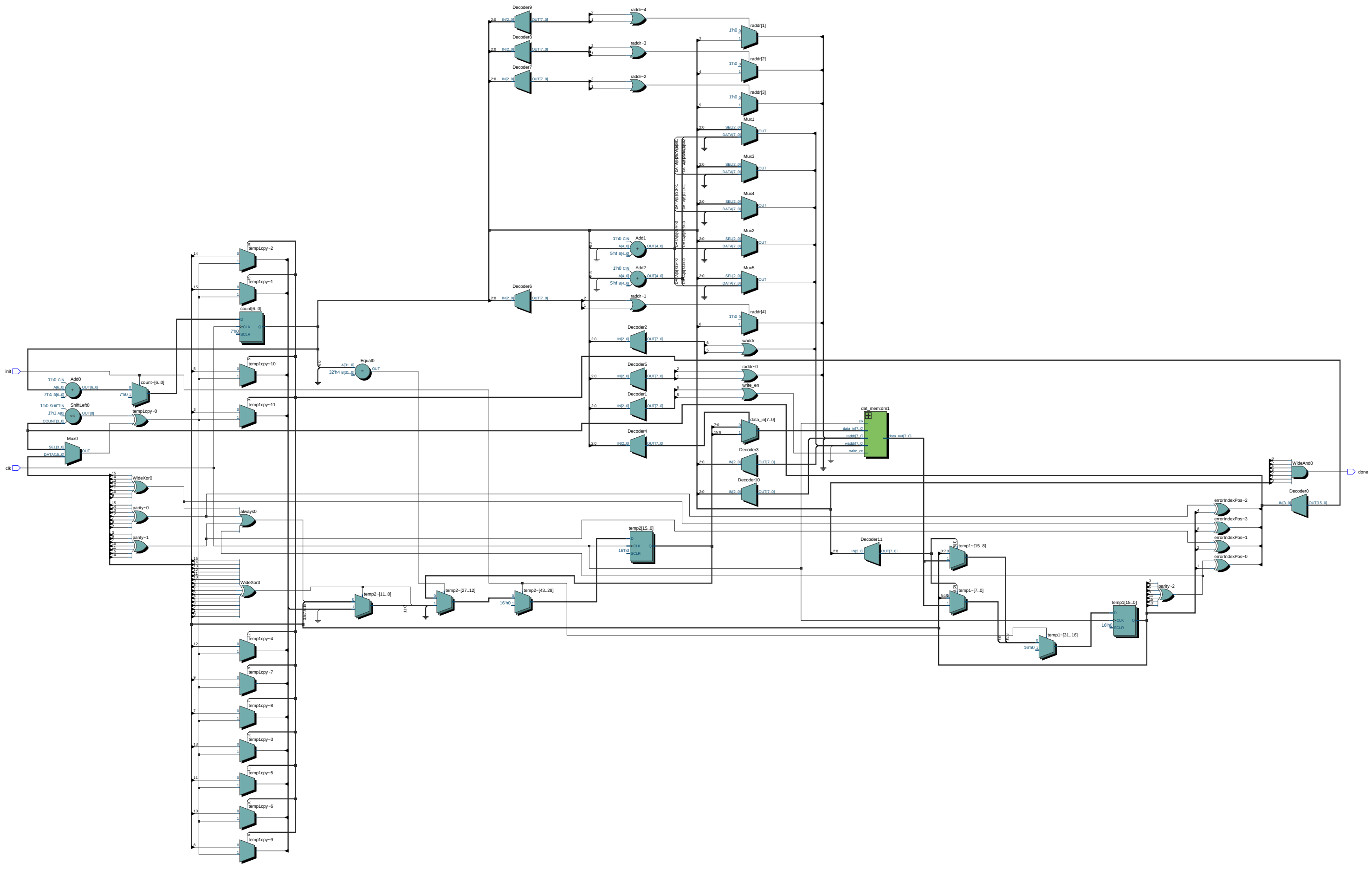
introduction_lab5.

What worked / what didn't.

For debugging, the testbench was modified to display a walking 1 bit. This modification was made to for debugging one bit errors. My written program was eventually functional for all test cases of walking ones. However, when returning the testbench to randomization, it failed to perform two bit and zero bit error checking. In the randomization of corrupted messages, the program that worked for walking 1 bits, only yielded 7/15 result. The program's issue was narrowed down to a delay in assignment between "temp1" and "temp1cpy", i.e. the bit array holding corrupted message, and its cloned bit array (also initalized with corrupted message from temp1).

After modifying the program, we were able to achieve a score of 8/15 at the expense of failure at every 1 bit error checking case. But succeeding in multiple zero bit error and two bit error (or more) checks.



rst
clk
done
Add0
count~[6..0]
ShiftLeft0
temp1cpy~0
Mux0
count[6..0]
Equal0
WideXor0
parity~0
parity~1
always0
WideXor3
temp1cpy~2
temp1cpy~1
temp1cpy~10
temp1cpy~11
temp1cpy~4
temp1cpy~7
temp1cpy~8
temp1cpy~3
temp1cpy~5
temp1cpy~6
temp1cpy~9
temp2~[11..0]
temp2~[27..12]
temp2~[43..28]
temp2[15..0]
Decoder9
Decoder8
Decoder7
Decoder6
raddr~4
raddr~3
raddr~2
raddr~1
raddr~0
raddr[1]
raddr[2]
raddr[3]
raddr[4]
Mux1
Mux3
Mux4
Mux2
Mux5
Add1
Add2
Decoder2
waddr
Decoder5
Decoder1
write_en
Decoder4
data_in[7..0]
Decoder3
Decoder10
dat_mem:dm1
Decoder11
temp1~[15..8]
temp1~[7..0]
temp1~[31..16]
temp1[15..0]
parity~2
errorIndexPos~2
errorIndexPos~3
errorIndexPos~1
errorIndexPos~0
WideAnd0
Decoder0

```
# Compile of dat_mem.sv was successful.
# Compile of lab5_tb.sv was successful.
# Compile of top_level.sv was successful.
# 3 compiles, 0 failed with no errors.
restart
# ** Note: (vsim-12125) Error and warning message counts have been reset to '0'
because of 'restart'.
# Loading work.lab5_tb
# Loading work.top_level
# Loading work.dat_mem
run -all
# good = 0000000000000000 case           0 flip   0000
# bad  = 0000000000000001                100100
#
# good = 0000000000000000 case           1 flip   0001
# bad  = 0000000000000000                000001
#
# good = 0000000000000000 case           2 flip   0010
# bad  = 0000001000000100                001001
#
# good = 0000000000000000 case           3 flip   0011
# bad  = 0000000000001000                100011
#
# good = 0000000000000000 case           4 flip   0100
# bad  = 0010000000010000                001101
#
# good = 0000000000000000 case           5 flip   0101
# bad  = 0010000000100000                001101
#
# good = 0000000000000000 case           6 flip   0110
# bad  = 0000000001000000                100101
#
# good = 0000000000000000 case           7 flip   0111
# bad  = 0000000010000000                010010
#
# good = 0000000000000000 case           8 flip   1000
# bad  = 0000000100000010                000001
#
# good = 0000000000000000 case           9 flip   1001
# bad  = 0010001000000000                001101
#
# good = 0000000000000000 case          10 flip   1010
# bad  = 0000010000000000                110110
#
# good = 0000000000000000 case          11 flip   1011
# bad  = 0000100000000000                111101
#
# good = 0000000000000000 case          12 flip   1100
# bad  = 0001000000000000                101101
#
```

```
# good = 0000000000000000 case          13 flip   1101
# bad  = 0011000000000000                001100
#
# good = 0000000000000000 case          14 flip   1110
# bad  = 0100000000000000                111001
#
#
# start lab5
#
# flip  =   0000
# flip2 = 100100
# 0100000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0000000000000001  Corrupted Message
# 0100000000000000  Recovered Message
# we have a match
#
# flip  =   0001
# flip2 = 000001
# 0000000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0000000000000000  Corrupted Message
# 0100000000000000  Recovered Message
# no error, but flaged as one
#
# flip  =   0010
# flip2 = 001001
# 0000000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0000001000000100  Corrupted Message
# 0100000000000000  Recovered Message
# double error injected here
# missed the double error
#
# flip  =   0011
# flip2 = 100011
# 0100000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0000000000001000  Corrupted Message
# 0100000000000000  Recovered Message
# we have a match
#
# flip  =   0100
# flip2 = 001101
# 0000000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0010000000010000  Corrupted Message
# 0100000000000000  Recovered Message
# double error injected here
# missed the double error
```

```
#
# flip  =   0101
# flip2 = 001101
# 0000000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0010000000100000  Corrupted Message
# 0100000000000000  Recovered Message
# double error injected here
# missed the double error
#
# flip  =   0110
# flip2 = 100101
# 0100000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0000000001000000  Corrupted Message
# 0100000000000000  Recovered Message
# we have a match
#
# flip  =   0111
# flip2 = 010010
# 0100000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0000000010000000  Corrupted Message
# 0100000000000000  Recovered Message
# we have a match
#
# flip  =   1000
# flip2 = 000001
# 0000000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0000000100000010  Corrupted Message
# 0100000000000000  Recovered Message
# double error injected here
# missed the double error
#
# flip  =   1001
# flip2 = 001101
# 0000000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0010001000000000  Corrupted Message
# 0100000000000000  Recovered Message
# double error injected here
# missed the double error
#
# flip  =   1010
# flip2 = 110110
# 0100000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0000010000000000  Corrupted Message
# 0100000000000000  Recovered Message
# we have a match
#
# flip  =   1011
# flip2 = 111101
# 0100000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0000100000000000  Corrupted Message
# 0100000000000000  Recovered Message
# we have a match
#
# flip  =   1100
# flip2 = 101101
# 0100000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0001000000000000  Corrupted Message
# 0100000000000000  Recovered Message
# we have a match
#
# flip  =   1101
# flip2 = 001100
# 0000000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0011000000000000  Corrupted Message
# 0100000000000000  Recovered Message
# double error injected here
# missed the double error
#
# flip  =   1110
# flip2 = 111001
# 0100000000000000  Original Message
# 0000000000000000  Message w/ parity
# 0100000000000000  Corrupted Message
# 0100000000000000  Recovered Message
# we have a match
#
# score =           8/15
# ** Note: $stop    : C:/Users/Albert/Desktop/Course Related/cse
140L/Project_Folder_qPrime_modelsim/Lab_4.5/testbench_and_starter_verilog/lab5_tb.sv
(109)
#    Time: 1435 ns  Iteration: 0  Instance: /lab5_tb
# Break in Module lab5_tb at C:/Users/Albert/Desktop/Course Related/cse
140L/Project_Folder_qPrime_modelsim/Lab_4.5/testbench_and_starter_verilog/lab5_tb.sv
line 109
```