Adam Burnett  
ECEn 620  
Assignment 3  
  
This lab involved using tasks, functions and do-while loops to improve my solution from the chapter 2 homework. To improve my solution, I added checks to verify if both read and write data were asserted at the same time, a test that checks the checker, calculation of even parity in the memory module and a do-while loop. These improvements helped both in functionality and readability of the code.

my\_mem code:

`default\_nettype none

**module** my\_mem**(**clk**,**

write**,**

read**,**

data\_in**,**

address**,**

data\_out**);**

**input** logic clk**;**

**input** logic write**;**

**input** logic read**;**

**input** logic **[**7**:**0**]** data\_in**;**

**input** logic **[**15**:**0**]** address**;**

**output** logic **[**8**:**0**]** data\_out**;**

//This function calculates even parity

**function** automatic logic ev\_parity**(**logic **[**7**:**0**]** din**);**

return **(^**din**);**

**endfunction**

//Declare a 9-bit associative array using the logic data type

logic **[**8**:**0**]** mem\_array**[**shortint**];**

**always** **@(posedge** clk**)** **begin**

**if** **(**write**)**

mem\_array**[**address**]** **=** **{**ev\_parity**(**data\_in**),** data\_in**};**

**else** **if** **(**read**)**

data\_out **=** mem\_array**[**address**];**

**end**

endmodule

testbench code:

`default\_nettype none

`timescale 1ns**/**100ps

**module** testbench **();**

logic clk**;**

logic write**;**

logic read**;**

logic **[**7**:**0**]** data\_in**;**

logic **[**15**:**0**]** address**;**

logic **[**8**:**0**]** data\_out**;**

shortint address\_array**[];**

byte data\_to\_write\_array**[];**

bit **[**8**:**0**]** data\_read\_expect\_assoc**[**shortint**];**

int error\_counter**;**

bit **[**8**:**0**]** data\_read\_queue**[**$**];**

//Read Write Checker

**function** automatic logic ReadWriteChecker**(**logic rd**,** logic wr**);**

**if(**rd **==** 1 **&&** wr **==** 1**)** **begin**

$display**(**"Error@%d: Read and Write asserted at the same time\n"**,** $time**);**

error\_counter**++;**

**end**

**endfunction**

//Test Read Write Checker

always\_comb **begin**

ReadWriteChecker**(**read**,** write**);**

**end**

//Test Bench Code

**initial** **begin**

clk **=** 0**;**

error\_counter **=** 0**;**

read **=** 0**;**

address\_array **=** new**[**6**];**

data\_to\_write\_array **=** new**[**6**];**

//Test Checker

**@(negedge** clk**);**

read **=** 1**;**

write **=** 1**;**

**@(negedge** clk**);**

read **=** 0**;**

write **=** 0**;**

//Do While Loop

do

**begin**

$display**(**"Waiting..."**);**

**#**10ns**;**

**end**

**while** **(**$time **<** 60 **);**

foreach**(**address\_array**[**i**])** **begin**

address\_array**[**i**]** **=** $random**;**

data\_to\_write\_array**[**i**]** **=** $random**;**

data\_read\_expect\_assoc**[**address\_array**[**i**]]** **=** **{^**data\_to\_write\_array**[**i**],** data\_to\_write\_array**[**i**]};**

**end**

foreach**(**address\_array**[**i**])** **begin**

write\_to\_memory**(**address\_array**[**i**],** data\_to\_write\_array**[**i**]);**

**end**

address\_array**.**reverse**();**

foreach**(**address\_array**[**i**])** **begin**

**@(negedge** clk**)**

read **=** 1**;**

address **=** address\_array**[**i**];**

**@(posedge** clk**)**

**#**1

data\_read\_queue**.**push\_back**(**data\_out**);**

**if(**data\_read\_expect\_assoc**[**address**]** **!=** data\_out**)** **begin**

$display**(**"Found Error: %03X %03X"**,** data\_read\_expect\_assoc**[**address**],** data\_out**);**

error\_counter **=** error\_counter **+** 1**;**

**end**

**end**

$display**(**"Print Read Values"**);**

foreach**(**data\_read\_queue**[**i**])** **begin**

$display**(**"\t%03X"**,** data\_read\_queue**[**i**]);**

**end**

$display**(**"Error Count: %d"**,** error\_counter**);**

**end**

**task** write\_to\_memory**(input** shortint addr**,** byte data**);**

write **=** 1**;**

data\_in **=** data**;**

address **=** addr**;**

**@(posedge** clk**);**

write **=** 0**;**

**@(negedge** clk**);**

**endtask;**

**always** **begin**

**#**5 clk **=** **~**clk**;**

**end**

my\_mem mem**(**clk**,**

write**,**

read**,**

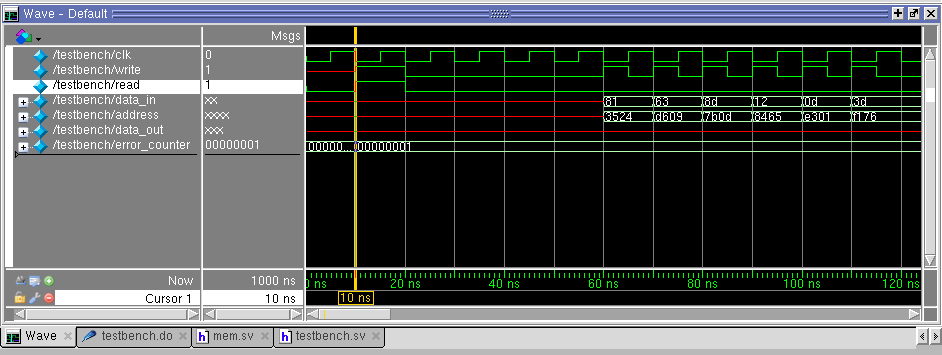
data\_in**,**

address**,**

data\_out**);**

endmodule

Waveform:

****

**Transcripts:**

**Error Free:**

# vsim -novopt testbench

# Refreshing /net/fpga1/users/adamrb/ECEn620/Assignments/Assignment3/work.testbench  
# Loading sv\_std.std  
# Loading work.testbench  
# Refreshing /net/fpga1/users/adamrb/ECEn620/Assignments/Assignment3/work.my\_mem  
# Loading work.my\_mem

# Waiting...  
# Waiting...  
# Waiting...  
# Waiting...  
# Waiting...  
# Waiting...

# Print Read Values

# 13d  
# 10d  
# 012  
# 08d  
# 063  
# 081

# Error Count: 0

**Read = 1 Write =1 Error**

# Refreshing /net/fpga1/users/adamrb/ECEn620/Assignments/Assignment3/work.testbench

# Loading sv\_std.std  
# Loading work.testbench  
# Refreshing /net/fpga1/users/adamrb/ECEn620/Assignments/Assignment3/work.my\_mem  
# Loading work.my\_mem  
**# Error@ 10: Read and Write asserted at the same time**#   
# Waiting...  
# Waiting...  
# Waiting...  
# Waiting...

# Print Read Values

# 13d  
# 10d  
# 012  
# 08d  
# 063  
# 081

# Error Count: 1

**Data Read Error**

# vsim -novopt testbench

# Refreshing /net/fpga1/users/adamrb/ECEn620/Assignments/Assignment3/work.testbench

# Loading sv\_std.std  
# Loading work.testbench  
# Refreshing /net/fpga1/users/adamrb/ECEn620/Assignments/Assignment3/work.my\_mem

# Loading work.my\_mem  
# Waiting...  
# Waiting...  
# Waiting...  
# Waiting...  
# Waiting...  
# Waiting...

**# Found Error: 13d 000  
# Found Error: 10d 000  
# Found Error: 012 000  
# Found Error: 08d 000  
# Found Error: 063 000  
# Found Error: 081 000**

# Print Read Values

# 000  
# 000  
# 000  
# 000  
# 000  
# 000

# Error Count: 6