**Joshua Monson  
ECEN 620   
12/1/14  
Assertions Lab**

**Assertions Lab Overview**

The goal of this lab is to gain experience using assertions to debug a design. In this lab we were to add assertions to debug a simple FIFO design and then move those assertions into a bind file. The rest of this report demonstrates that I have completed this assignment and includes the following:

1. “debugged” fifo1.sv – bug fixes have been annotated with comments.
2. Bindfile.sv and tb1.sv – contains my assertions and demonstrates that it was properly instantiated.
3. Screen Shot of assertions Window (All Pass) – demonstrates that I corrected the fifo design sufficiently to get all assertions to pass
4. Transcript (Working) – demonstrates that my corrections to the fifo design caused the self-checking test bench to pass.
5. Screen Shot of non-working assertions – demonstrating that my assertions fired and failed.

**Debugged fifo.sv**

//----------------------------------------------------------------------

// This File: fifo1.sv

//

// Copyright 2003-2014 Sunburst Design, Inc.

//

// Sunburst Design (Beaverton, OR):

// cliffc@sunburst-design.com

// www.sunburst-design.com

//----------------------------------------------------------------------

**module** fifo1 **(**

**output** logic **[**7**:**0**]** dout**,**

**output** logic full**,** empty**,**

**input** logic write**,** read**,** clk**,** rst\_n**,**

**input** logic **[**7**:**0**]** din**);**

logic **[**7**:**0**]** fifomem **[**0**:**15**];**

logic **[**3**:**0**]** wptr**,** rptr**;**

logic **[**4**:**0**]** cnt**;** //Bug 3: Count Cannot Represent 1-16 and 0 (changed range 3:0 -> 4:0)

always\_ff **@(posedge** clk **or** **negedge** rst\_n**)**

**if** **(!**rst\_n**)** **begin**

wptr **<=** '0**;**

rptr **<=** '0**;** //Bug 1: rptr was not reset (Added this line)

cnt **<=** '0**;**

empty **<=** '1**;**

full **<=** '0**;**

**end**

**else**

**case** **({**write**,** read**})**

2'b00**:** **;** // no fifo write or read

2'b01**:** **begin** // fifo read

**if(**cnt **>** 0**)** **begin** //Bug 2: Read Only Occurs when FIFO not Empty (Added if statement)

full **<=** '0**;**

rptr **<=** rptr **+** 1**;**

cnt **<=** cnt **-** 1**;**

**if** **(**cnt**==**1**)** empty **<=** '1**;** //Bug 8: Empty should only be set when reading the fifo with a count of 1

**end**

**end**

2'b10**:** **begin** // fifo write

empty **<=** '0**;**

**if** **(**cnt**<**16**)** **begin** //Bug 4: Write Should Only Occur when not full removed ==

wptr **<=** wptr **+** 1**;**

cnt **<=** cnt **+** 1**;**

**if** **(**cnt **==** 15**)** full **<=** '1**;** //Bug 7: Full Should be set when writing the 16th fifo word

**end**

**end**

2'b11**:** // fifo write & read

**if** **(**full**)** **begin**

rptr **<=** rptr **+** 1**;**

cnt **<=** cnt **-** 1**;**

full **<=** '0**;** //Bug 5: Must Lower Full Signal

**end**

**else** **if** **(**empty**)** **begin**

wptr **<=** wptr **+** 1**;**

cnt **<=** cnt **+** 1**;**

empty **<=** '0**;** //Bug 6: Must Lower Empty Signal

**end**

**else** **begin**

wptr **<=** wptr **+** 1**;**

rptr **<=** rptr **+** 1**;**

**end**

**endcase**

// FIFO synchronous memory write operation

always\_ff **@(posedge** clk**)**

**if** **(**write **&&** **((**cnt **<**16**)** **||** **((**cnt**==**16**)** **&&** read**)))**

fifomem**[**wptr**]** **<=** din**;**

**assign** dout **=** fifomem**[**rptr**];**

endmodule

**bindfile.sv**

**module** bindfile**(input** clk**,** rst\_n**,**

**input** full**,** empty**,** read**,** write**,**

**input** **[**4**:**0**]** cnt**,**

**input** **[**3**:**0**]** rptr**,** **[**3**:**0**]** wptr**);**

`ifndef ASSERT\_MACROS

`define ASSERT\_MACROS

`define assert\_clk**(**arg**,** ck**=**clk**)** \

assert property **(@(posedge** ck**)** **disable** iff **(!**rst\_n**)** arg**)**

`define assert\_async\_rst**(**arg**,** ck**=**clk**)** \

assert property **(@(posedge** ck**)** arg**)**

`endif

ERROR\_FIFO\_RESET\_SHOULD\_CAUSE\_EMPTY1\_FULL0\_RPTR0\_WPTR0\_CNT0**:**

`assert\_async\_rst**(!**rst\_n **|->** **(**rptr**==**0 **&&** wptr**==**0 **&&** empty**==**1 **&&** full**==**0 **&&** cnt**==**0**));**

ERROR\_FIFO\_SHOULD\_BE\_FULL**:**

`assert\_clk**(**cnt**>**15 **|->** full**);**

ERROR\_FIFO\_SHOULD\_NOT\_BE\_FULL**:**

`assert\_clk**(**cnt**<**16 **|->** **!**full**);**

ERROR\_FIFO\_DID\_NOT\_GO\_FULL**:**

`assert\_clk**(**cnt**==**15 **&&** write **&&** **!**read **|->** **##**1 full**);**

ERROR\_FIFO\_SHOULD\_BE\_EMPTY**:**

`assert\_clk**(**cnt**==**0 **|->** empty**);**

ERROR\_FIFO\_SHOULD\_NOT\_BE\_EMPTY**:**

`assert\_clk**(**cnt**>**0 **|->** **!**empty**);**

ERROR\_FIFO\_DID\_NOT\_GO\_EMPTY**:**

`assert\_clk**(**cnt**==**1 **&&** read **&&** **!**write **|->** **##**1 empty**);**

ERROR\_FIFO\_FULL\_WRITE\_CAUSED\_WPTR\_TO\_CHANGE**:**

`assert\_clk**((**full **&&** write **&&** **!**read**)** **|->** **##**1 $stable**(**wptr**));**

ERROR\_FIFO\_FULL\_WRITE\_CAUSED\_FULL\_FLAG\_TO\_CHANGE**:**

`assert\_clk**((**full **&&** write **&&** **!**read**)** **|->** **##**1 $stable**(**full**));**

ERROR\_FIFO\_EMPTY\_READ\_CAUSED\_EMPTY\_FLAG\_TO\_CHANGE**:**

`assert\_clk**((**empty **&&** read **&&** **!**write**)** **|->** **##**1 $stable**(**empty**));**

ERROR\_FIFO\_EMPTY\_READ\_CAUSED\_RPTR\_TO\_CHANGE**:**

`assert\_clk**((**empty **&&** read **&&** **!**write**)** **|->** **##**1 $stable**(**rptr**));**

ERROR\_FIFO\_WORD\_COUNTER\_IS\_NEGATIVE**:**

`assert\_clk**((**cnt **>=**0**));**

ERROR\_FIFO\_READWRITE\_ILLEGAL\_FIFO\_FULL\_OR\_EMPTY**:**

`assert\_clk**(**read **&&** write **&&** **!**full **&&** **!**empty **|->** **##**1 $stable**(**full**)** **&&** $stable**(**empty**));**

**endmodule** // bindfile

**tb1.sv**

//----------------------------------------------------------------------

// This File: tb1.sv

//

// Copyright 2003-2014 Sunburst Design, Inc.

//

// Sunburst Design (Beaverton, OR):

// cliffc@sunburst-design.com

// www.sunburst-design.com

//----------------------------------------------------------------------

`timescale 1ns**/**1ns

**module** tb1**;**

logic **[**7**:**0**]** dout**;**

logic **[**7**:**0**]** din**;**

logic empty**,** full**;**

logic read**,** write**;**

logic rst\_n**;**

SIMUTIL T **(.**clk**(**clk**));**

`ifdef FIFOGOOD

fifogood1 u1 **(.\*);**

`elsif FIFOBAD

fifobad1 u1 **(.\*);**

`else

fifo1 u1 **(.\*);**

`endif

bind fifo1 bindfile b1**(.**wptr**(**wptr**),** **.**rptr**(**rptr**),** **.**cnt**(**cnt**),** **.\*);**

**initial** **begin** // Stimulus

initialize**;**

fiforead**;**

**repeat** **(**16**)** fifowrite **(**8'hAA**);**

**repeat** **(** 2**)** fifowriteread**(**8'hFF**);**

**repeat** **(**16**)** fiforead**;**

**repeat** **(** 8**)** fifowrite **(**8'h99**);**

**repeat** **(** 5**)** fiforead**;**

reset**;**

**repeat** **(** 8**)** fifowrite **(**8'h66**);**

**repeat** **(** 6**)** fifowrite **(**8'h55**);**

**repeat** **(** 9**)** fiforead**;**

**repeat** **(**10**)** fifowrite **(**8'hFF**);**

**repeat** **(** 2**)** fifowriteread**(**8'hFF**);**

**repeat** **(** 4**)** fifowrite **(**8'hFF**);**

**repeat** **(**17**)** fiforead**;**

**end**

**initial** **begin** // Verification

$timeformat**(-**9**,**0**,**"ns"**,**10**);**

T**.**STARTSIM**;**

**repeat(** 2**)** fifoexpect**(**8'hxx**,**0**,**1**);**

**repeat(**15**)** fifoexpect**(**8'hAA**,**0**,**0**);**

**repeat(** 1**)** fifoexpect**(**8'hAA**,**1**,**0**);**

**repeat(**15**)** fifoexpect**(**8'hAA**,**0**,**0**);**

**repeat(** 1**)** fifoexpect**(**8'hFF**,**0**,**0**);**

**repeat(** 2**)** fifoexpect**(**8'hxx**,**0**,**1**);**

**repeat(**12**)** fifoexpect**(**8'h99**,**0**,**0**);**

**repeat(** 2**)** fifoexpect**(**8'hxx**,**0**,**1**);**

**repeat(**21**)** fifoexpect**(**8'h66**,**0**,**0**);**

**repeat(**14**)** fifoexpect**(**8'h55**,**0**,**0**);**

**repeat(** 4**)** fifoexpect**(**8'h55**,**1**,**0**);**

**repeat(** 2**)** fifoexpect**(**8'h55**,**0**,**0**);**

**repeat(**13**)** fifoexpect**(**8'hFF**,**0**,**0**);**

**repeat(**30**)** fifoexpect**(**8'hxx**,**0**,**1**);**

T**.**STOPSIM**;**

T**.**FINISH**;**

**end**

**task** initialize**;**

rst\_n **<=** 0**;**

din **<=** 8'hff**;**

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

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

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

**@(negedge** clk**)** rst\_n **=** 1**;**

**endtask**

**task** reset**;**

**@(negedge** clk**)** rst\_n **=** 0**;**

**@(negedge** clk**)** rst\_n **=** 1**;**

**endtask**

**task** cycle\_delay**;**

**input** **[**31**:**0**]** dlycnt**;**

**repeat** **(**dlycnt**)** **@(negedge** clk**);**

**endtask**

**task** fifowrite**;**

**input** **[**7**:**0**]** wdata**;**

**@(negedge** clk**)** write **=** 1**;**

din **=** wdata**;**

**@(negedge** clk**)** write **=** 0**;**

**endtask**

**task** fiforead**;**

**@(negedge** clk**)** read **=** 1**;**

**@(negedge** clk**)** read **=** 0**;**

**endtask**

**task** fifowriteread**;**

**input** **[**7**:**0**]** wdata**;**

**@(negedge** clk**)** write **=** 1**;**

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

din **=** wdata**;**

**@(negedge** clk**)** write **=** 0**;**

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

**endtask**

**task** fifoexpect**;**

**input** **[**7**:**0**]** exdata**;**

**input** exfull**,** exempty**;**

**repeat** **(**2**)** **@(posedge** clk**);** **#(**`CYCLE**-**1**);**

**if** **((**full **===** exfull**)** **&&** **(**empty **===** exempty**)** **&&** **(**dout **===** exdata**))**

T**.**PASS**;**

`ifndef BUG

**else** **if** **((**full **===** exfull**)** **&&** **(**empty **===** exempty**)** **&&** **(**empty **===** 1'b1**))** // Correction

T**.**PASS**;** // Correction

`endif

**else** **begin**

T**.**ERROR**;**

$display**(**"%t: FIFO: data=%h full=%b empty=%b"**,** $time**,** dout**,** full**,** empty**);**

$display**(**" EXPECTED: data=%h full=%b empty=%b\n\n"**,** exdata**,** exfull**,** exempty**);**

**end**

**endtask**

`ifdef VCD

**initial** **begin**

$dumpfile**(**"dump.svcd"**);**

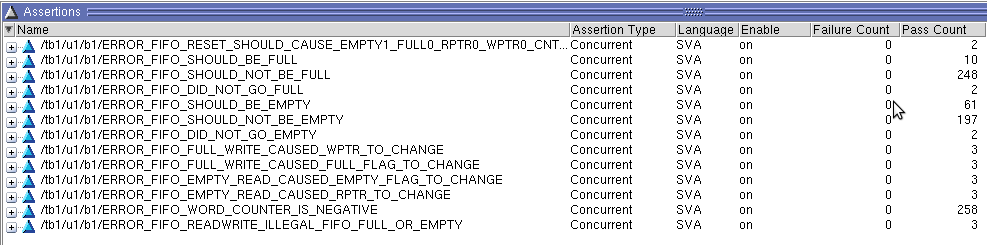
$dumpvars**;**

**end**

`endif

endmodule

**Screen-Shot of Assertions Window (Assertions Passing)**

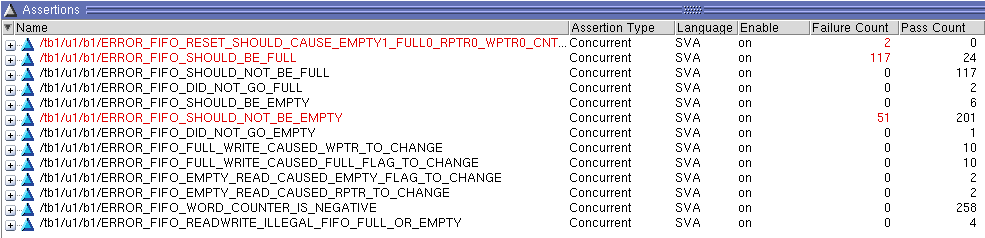
****

**Transcript of Testbench Passing**

// Questa Sim-64  
# // Version 10.1c linux\_x86\_64 Jul 27 2012  
# //  
# // Copyright 1991-2012 Mentor Graphics Corporation  
# // All Rights Reserved.  
# //  
# // THIS WORK CONTAINS TRADE SECRET AND PROPRIETARY INFORMATION  
# // WHICH IS THE PROPERTY OF MENTOR GRAPHICS CORPORATION OR ITS  
# // LICENSORS AND IS SUBJECT TO LICENSE TERMS.  
# //  
# vsim -assertdebug -do {view wave;do wave.do; run 26000 ns} -novopt tb1   
# Refreshing /net/fpga1/users/joshuas2/ECEn620/Assignments/AssertionsLab/work.tb1  
# Loading sv\_std.std  
# Loading work.tb1  
# Refreshing /net/fpga1/users/joshuas2/ECEn620/Assignments/AssertionsLab/work.SIMUTIL  
# Loading work.SIMUTIL  
# Refreshing /net/fpga1/users/joshuas2/ECEn620/Assignments/AssertionsLab/work.fifo1  
# Loading work.fifo1  
# Refreshing /net/fpga1/users/joshuas2/ECEn620/Assignments/AssertionsLab/work.bindfile  
# Loading work.bindfile  
# view wave   
# .main\_pane.wave.interior.cs.body.pw.wf  
# do wave.do   
# run 26000 ns   
run 1000 ns  
#   
# TEST PASSED - test #1: 134 vectors - 134 passed  
#   
#   
# //--------------------------------------------------------  
# // SIMUTIL Inform: Ran simulation with TYPICAL delays  
# //--------------------------------------------------------  
#   
#

# \*\*\*TEST SUITE PASSED - 1 test - 1 passed  
# \*\* Note: Data structure takes 30525440 bytes of memory  
# Process time 0.00 seconds  
# $finish : simutil.v(236)  
# Time: 26900 ns Iteration: 1 Instance: /tb1  
# 1  
# Break in Task FINISH at simutil.v line 236

**Screen-Shot of Assertions Window (Assertions Failing)**

****