Josh Monson

Aaron Stoddard

EE 620

**Chapter 8 HW**

**Overview:**

In this assignment we created our first full-blown test bench (minus a DUT). This test bench generated IPV4 packets and calls a scoreboard to determine if they were properly formed. We also added call back hooks into our scoreboard that allowed us to create a new test that corrupted packets without modifying our test bench environments. Additionally, we extend our packet class to create a bad\_packet class that corrupted the CRC checksum on 2% percent of the packets.

Furthermore, we created 3 different tests, one that transmits 1000 good packets, one that uses a call back to corrupt the version of 1% of the packets, and one that uses corrupts the checksum (as mentioned above). We created a test registry to run the different tests via command line arguments. Below is given directions on running our solution including command line commands, as well as the transcript windows for each of the three tests.

The source code for our project is included in this document.

**Directions on Running Solution:**

To execute our test programs run the following commands within our project directory:

1. ***make factory***
2. ***vsim -c test -do "run -all" +TESTNAME=TestGood***
3. ***vsim -c test -do "run -all" +TESTNAME=Test\_v3***
4. ***vsim -c test -do "run -all" +TESTNAME=TestBad***

**Transcript Windows:** bash console output showing compilation of project and running each project without recompiling. To enable you to better see where each step begins and ends we’ve bolded, highlighted and italicized each command that was run on the bash console.

***[joshuas2@manticore Homework9]$ make factory***

vlib work

vmap work work

Modifying modelsim.ini

vlog -mfcu -sv packet\_pkg.sv scoreboard\_pkg.sv package\_test.sv factory\_pkg.sv test\_program.sv

QuestaSim-64 vlog 10.1c Compiler 2012.07 Jul 27 2012

-- Compiling package packet\_pkg

\*\* Warning: packet\_pkg.sv(85): (vlog-LRM-7047) Return type of virtual method 'copy' in class 'packet' is a subtype of the method return type in the superclass. For strict LRM compliance the return types must match.

-- Compiling package scoreboard\_pkg

-- Importing package packet\_pkg

-- Compiling package package\_test

-- Importing package scoreboard\_pkg

-- Compiling package factory\_pkg

-- Compiling package packet\_pkg\_sv\_unit

-- Importing package factory\_pkg

-- Importing package package\_test

-- Importing package packet\_pkg

-- Importing package scoreboard\_pkg

-- Compiling program test

Top level modules:

test

***[joshuas2@manticore Homework9]$ vsim -c test -do "run -all" +TESTNAME=TestGood***

Reading /net/fpga2/questasim10.1c/questasim/tcl/vsim/pref.tcl

# 10.1c

# vsim +TESTNAME=TestGood -do {run -all} -c test

# \*\* Note: (vsim-3812) Design is being optimized...

# \*\* Warning: packet\_pkg.sv(85): (vopt-LRM-7047) Return type of virtual method 'copy' in class 'packet' is a subtype of the method return type in the superclass. For strict LRM compliance the return types must match.

# \*\* Warning: factory\_pkg.sv(22): (vopt-2250) Function "register" has no return value assignment.

# // 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.

# //

# Loading sv\_std.std

# Loading work.packet\_pkg(fast)

# Loading work.scoreboard\_pkg(fast)

# Loading work.package\_test(fast)

# Loading work.factory\_pkg(fast)

# Loading work.packet\_pkg\_sv\_unit(fast)

# Loading work.test(fast)

# run -all

# Factory:

# The name: TestBad maps to the class: '{me:@registry\_\_3@1}

# The name: TestGood maps to the class: '{me:@registry\_\_1@1}

# The name: Test\_v3 maps to the class: '{me:@registry\_\_2@1}

# factory\_pkg.factory.get\_test found TESTNAME=TestGood

# Create\_object called with TestGood which maps to key TestGood and class class {Environment env;}/packet\_pkg\_sv\_unit::TestGood extends class {}/factory\_pkg::component

# Running Good Test:

# 10000:Test Finished with 1000 comparisons.

# Break in Task packet\_pkg\_sv\_unit/TestGood::run\_test at TestGood.sv line 17

# Stopped at TestGood.sv line 17

VSIM 2> quit

***[joshuas2@manticore Homework9]$ vsim -c test -do "run -all" +TESTNAME=Test\_v3***

Reading /net/fpga2/questasim10.1c/questasim/tcl/vsim/pref.tcl

# 10.1c

# vsim +TESTNAME=Test\_v3 -do {run -all} -c test

# // 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.

# //

# Loading sv\_std.std

# Loading work.packet\_pkg(fast)

# Loading work.scoreboard\_pkg(fast)

# Loading work.package\_test(fast)

# Loading work.factory\_pkg(fast)

# Loading work.packet\_pkg\_sv\_unit(fast)

# Loading work.test(fast)

# run -all

# Factory:

# The name: TestBad maps to the class: '{me:@registry\_\_3@1}

# The name: TestGood maps to the class: '{me:@registry\_\_1@1}

# The name: Test\_v3 maps to the class: '{me:@registry\_\_2@1}

# factory\_pkg.factory.get\_test found TESTNAME=Test\_v3

# Create\_object called with Test\_v3 which maps to key Test\_v3 and class class {Environment env;}/packet\_pkg\_sv\_unit::Test\_v3 extends class {}/factory\_pkg::component

# Running v3 Test:

# 510: ERROR for version, expected=0x3 != actual=0x4

# 510: ERROR for header checksum, expected=0x9554 != actual=0xe554

# 770: ERROR for version, expected=0x3 != actual=0x4

# 770: ERROR for header checksum, expected=0xe8d6 != actual=0x98d6

# 1150: ERROR for version, expected=0x3 != actual=0x4

# 1150: ERROR for header checksum, expected=0x1357 != actual=0x6357

# 2150: ERROR for version, expected=0x3 != actual=0x4

# 2150: ERROR for header checksum, expected=0xd77e != actual=0xa77e

# 2270: ERROR for version, expected=0x3 != actual=0x4

# 2270: ERROR for header checksum, expected=0x85bb != actual=0xf5bb

# 3370: ERROR for version, expected=0x3 != actual=0x4

# 3370: ERROR for header checksum, expected=0x8a1a != actual=0xfa1a

# 4540: ERROR for version, expected=0x3 != actual=0x4

# 4540: ERROR for header checksum, expected=0xdb0a != actual=0xab0a

# 8630: ERROR for version, expected=0x3 != actual=0x4

# 8630: ERROR for header checksum, expected=0xc14c != actual=0xb14c

# 10000: Corrupted 8 Packets

# 10000:Test Finished with 1000 comparisons.

# Break in Task packet\_pkg\_sv\_unit/Test\_v3::run\_test at Test\_v3.sv line 20

# Stopped at Test\_v3.sv line 20

VSIM 2> quit

***[joshuas2@manticore Homework9]$ vsim -c test -do "run -all" +TESTNAME=TestBad***

Reading /net/fpga2/questasim10.1c/questasim/tcl/vsim/pref.tcl

# 10.1c

# vsim +TESTNAME=TestBad -do {run -all} -c test

# // 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.

# //

# Loading sv\_std.std

# Loading work.packet\_pkg(fast)

# Loading work.scoreboard\_pkg(fast)

# Loading work.package\_test(fast)

# Loading work.factory\_pkg(fast)

# Loading work.packet\_pkg\_sv\_unit(fast)

# Loading work.test(fast)

# run -all

# Factory:

# The name: TestBad maps to the class: '{me:@registry\_\_3@1}

# The name: TestGood maps to the class: '{me:@registry\_\_1@1}

# The name: Test\_v3 maps to the class: '{me:@registry\_\_2@1}

# factory\_pkg.factory.get\_test found TESTNAME=TestBad

# Create\_object called with TestBad which maps to key TestBad and class class {Environment env;}/packet\_pkg\_sv\_unit::TestBad extends class {}/factory\_pkg::component

# Running Bad Test:

# 1160: ERROR for header checksum, expected=0xd2f5 != actual=0x2d0a

# 1490: ERROR for header checksum, expected=0x4d96 != actual=0xb269

# 1950: ERROR for header checksum, expected=0xf2ba != actual=0xd45

# 2250: ERROR for header checksum, expected=0x4dd1 != actual=0xb22e

# 3000: ERROR for header checksum, expected=0xb4f8 != actual=0x4b07

# 3050: ERROR for header checksum, expected=0xcab4 != actual=0x354b

# 3330: ERROR for header checksum, expected=0xf55e != actual=0xaa1

# 3440: ERROR for header checksum, expected=0x3d12 != actual=0xc2ed

# 4350: ERROR for header checksum, expected=0x103 != actual=0xfefc

# 4360: ERROR for header checksum, expected=0x83f8 != actual=0x7c07

# 4480: ERROR for header checksum, expected=0x4d78 != actual=0xb287

# 5550: ERROR for header checksum, expected=0x3e63 != actual=0xc19c

# 7440: ERROR for header checksum, expected=0x354a != actual=0xcab5

# 9170: ERROR for header checksum, expected=0x612b != actual=0x9ed4

# 10000:Send 14 Bad Packets

# 10000:Test Finished with 1000 comparisons.

[joshuas2@manticore Homework9]$

**Source Code**

**makefile:**

TOPLEVEL**=**test

VERILOG\_FILES**=** packet\_pkg.sv scoreboard\_pkg.sv package\_test.sv factory\_pkg.sv test\_program.sv

.PHONY**:** TestGood

TestGood**:** factory

vsim -c test -do "run -all" +TESTNAME=TestGood

.PHONY**:** TestBad

TestBad**:** factory

vsim -c test -do "run -all" +TESTNAME=TestBad

.PHONY**:** Test\_v3

Test\_v3**:** factory

vsim -c test -do "run -all" +TESTNAME=Test\_v3

.PHONY**:** factory

factory**:** ${VERILOG\_FILES} clean

vlib work

vmap work work

vlog -mfcu -sv ${VERILOG\_FILES}

clean**:**

@rm -rf work transcript vsim.wlf

**packet\_pkg.sv:**

package packet\_pkg**;**

**parameter** VERSION **=** 4**;**

**parameter** IHL **=** 5**;**

**parameter** TOTAL\_LENGTH **=** 224**;**

class header\_class**;**

rand bit**[**3**:**0**]** version**;**

rand bit**[**3**:**0**]** ihl**;**

rand bit**[**15**:**0**]** total\_length**;**

rand bit**[**7**:**0**]** type\_of\_service**;**

rand bit**[**15**:**0**]** identification**;**

rand bit**[**2**:**0**]** flags**;**

rand bit**[**12**:**0**]** fragment\_offset**;**

rand bit**[**7**:**0**]** time\_to\_live**;**

rand bit**[**7**:**0**]** protocol**;**

bit**[**15**:**0**]** header\_checksum**;**

rand bit**[**31**:**0**]** destination\_ip\_address**;**

rand bit**[**31**:**0**]** source\_ip\_address**;**

constraint flag\_c **{**

flags**[**0**]** **==** 0**;**

**}**

constraint constants **{**

version **==** 4**;** ihl **==** 5**;** total\_length **==** 224**;**

**}**

**function** new**();**

**endfunction**

**function** header\_class copy**();**

copy **=** new**();**

copy**.**version **=** version**;**

copy**.**ihl **=** ihl**;**

copy**.**total\_length **=** total\_length**;**

copy**.**type\_of\_service **=** type\_of\_service**;**

copy**.**identification **=** identification**;**

copy**.**flags **=** flags**;**

copy**.**fragment\_offset **=** fragment\_offset**;**

copy**.**time\_to\_live **=** time\_to\_live**;**

copy**.**protocol **=** protocol**;**

copy**.**header\_checksum **=** header\_checksum**;**

copy**.**destination\_ip\_address **=** destination\_ip\_address**;**

copy**.**source\_ip\_address **=** source\_ip\_address**;**

return copy**;**

**endfunction**

**function** void calc\_header\_checksum**();**

header\_checksum **=** **{**version**,** ihl**,** type\_of\_service**}** **^**

total\_length **^** identification **^**

**{**flags**,** fragment\_offset**}** **^** **{**time\_to\_live**,** protocol**}** **^**

source\_ip\_address**[**31**:**16**]** **^** source\_ip\_address**[**15**:**0**]** **^**

destination\_ip\_address**[**31**:**16**]** **^** destination\_ip\_address**[**15**:**0**];**

**endfunction**

endclass // header\_class

class data\_class**;**

rand bit**[**63**:**0**]** payload**;**

**function** new**();**

**endfunction**

**function** data\_class copy**();**

copy **=** new**();**

copy**.**payload **=** payload**;**

return copy**;**

**endfunction**

endclass // data\_class

virtual class base\_packet**;**

rand header\_class header**;**

rand data\_class data**;**

static int count**;**

int id**;**

**function** new**();**

id **=** count**++;**

header **=** new**();**

data **=** new**();**

**endfunction** // new

pure virtual **function** base\_packet copy**();**

pure virtual **function** void display**();**

pure virtual **function** void calc\_header\_checksum**();**

endclass**;** // base\_packet

class packet extends base\_packet**;**

**function** new**();**

super**.**new**();**

**endfunction**

**function** packet copy**();**

copy **=** new**();**

copy**.**header **=** header**.**copy**();**

copy**.**data **=** data**.**copy**();**

return copy**;**

**endfunction**

**function** void display**();**

$display**(**"Transaction ID: %d"**,**id**);**

**endfunction**

**function** void calc\_header\_checksum**();**

header**.**calc\_header\_checksum**();**

**endfunction**

endclass // packet

endpackage // packet\_pkg

**package\_test.sv:**

package package\_test**;**

import packet\_pkg**::\*;**

import scoreboard\_pkg**::\*;**

`define SV\_RAND\_CHECK**(**r**)\**

do **begin** \

**if** **(!(**r**))** begin\

$display**(**"%s:%0d Randomization failed \"%s\""**,** \

`\_\_FILE\_\_**,** `\_\_LINE\_\_**,** `"r`"**);** \

**end** \

**end** **while(**0**)**

virtual class Driver\_cbs**;**

virtual **task** pre\_tx**(**ref packet pkt**);**

//Callback does nothing

**endtask** // pre\_tx

virtual **task** post\_tx**(**ref packet pkt**);**

**endtask** // post\_tx

endclass // Driver\_cbs

class Driver\_cbs\_scoreboard extends Driver\_cbs**;**

Scoreboard scb**;**

**function** new**();**

this**.**scb **=** new**();**

**endfunction**

virtual **task** post\_tx**(**ref packet pkt**);**

scb**.**compare\_expected**(**pkt**);**

**endtask** // post\_tx

endclass

class Driver\_cbs\_v3 extends Driver\_cbs**;**

int corruption\_cnt**;**

**function** new **();**

corruption\_cnt **=** 0**;**

**endfunction** // new

virtual **task** pre\_tx**(**ref packet pkt**);**

int rval **=** $urandom\_range**(**0**,**99**);**

**if(**rval **==** 0**)** **begin**

pkt**.**header**.**version **=** 3**;**

corruption\_cnt**++;**

**end**

**endtask**

endclass

class Driver**;**

mailbox **#(**packet**)** gen2drv**;**

packet p**;**

Driver\_cbs cbs**[**$**];**

**function** new**(input** mailbox **#(**packet**)** gen2drv**);**

this**.**gen2drv **=** gen2drv**;**

**endfunction** // new

**task** run**(input** int count**);**

**repeat(**count**)** **begin**

gen2drv**.**get**(**p**);**

foreach **(**cbs**[**i**])** cbs**[**i**].**pre\_tx**(**p**);**

transmit**(**p**);**

foreach **(**cbs**[**i**])** cbs**[**i**].**post\_tx**(**p**);**

**end**

**endtask** // run

**task** transmit**(**base\_packet pkt**);**

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

**endtask**

endclass

class Generator**;**

mailbox **#(**packet**)** gen2drv**;**

packet blueprint**;**

**function** new**(input** mailbox **#(**packet**)** gen2drv**);**

this**.**gen2drv **=** gen2drv**;**

blueprint **=** new**();**

**endfunction**

**task** run**(input** int count**);**

**repeat(**count**)** **begin**

`SV\_RAND\_CHECK**(**blueprint**.**randomize**());**

blueprint**.**calc\_header\_checksum**();**

gen2drv**.**put**(**blueprint**.**copy**());** // requires base\_packet class copy to have been implemented by a child class or will not compile!

**end**

**endtask**

endclass

class Environment**;**

Generator gen**;**

Driver drv**;**

mailbox **#(**packet**)** gen2drv**;**

int count**;**

**function** new**(**int count**);**

this**.**count **=** count**;**

**endfunction;**

**function** void build**();**

gen2drv **=** new**();**

gen **=** new**(**gen2drv**);**

drv **=** new**(**gen2drv**);**

**endfunction**

**task** run**();**

**fork**

gen**.**run**(**count**);**

drv**.**run**(**count**);**

**join**

**endtask**

endclass

endpackage

**factory\_pkg.sv:**

package factory\_pkg**;**

virtual class component**;**

pure virtual **task** run\_test**();**

endclass // component

virtual class wrapper**;**

pure virtual **function** string get\_type\_name**();**

pure virtual **function** component create\_object**(**string name**);**

endclass // wrapper

class factory**;**

static wrapper type\_names**[**string**];**

static factory inst**;**

static **function** factory get**();**

**if** **(**inst **==** null**)** inst **=** new**();**

return inst**;**

**endfunction** // get

static **function** register**(**wrapper c**);**

type\_names**[**c**.**get\_type\_name**()]** **=** c**;**

**endfunction** // register

static **function** component get\_test**();**

string name**;**

wrapper test\_wrapper**;**

component test\_component**;**

**if(** **!**$value$plusargs**(**"TESTNAME=%s"**,**name**))begin**

$display**(**"FATAL +TESTNAME not found"**);**

$finish**;**

**end**

$display**(**"%m found TESTNAME=%s"**,**name**);**

test\_wrapper **=** factory**::**type\_names**[**name**];**

$cast**(**test\_component**,** test\_wrapper**.**create\_object**(**name**));**

return test\_component**;**

**endfunction** // get\_test

static **function** void printFactory**();**

$display**(**"Factory:"**);**

foreach **(**type\_names**[**s**])** $display**(**"The name: %s maps to the class: %p"**,** s**,** type\_names**[**s**]);**

**endfunction** // printFactory

endclass // factory

class registry **#(**type T**,** string Tname**)** extends wrapper**;**

typedef registry **#(**T**,**Tname**)** this\_type**;**

local static this\_type me **=** get**();**

static **function** this\_type get**();**

**if(**me **==** null**)** **begin**

factory f **=** factory**::**get**();**

me **=** new**();**

void'**(**f**.**register**(**me**));**

**end**

return me**;**

**endfunction** // get

virtual **function** string get\_type\_name**();**

return Tname**;**

**endfunction** // get\_type\_name

virtual **function** component create\_object**(**string name**);**

T toReturn**;**

$display**(**"Create\_object called with %s which maps to key %s and class %s"**,**

name**,**

Tname**,**

$typename**(**T**));**

toReturn **=** new**();**

return toReturn**;**

**endfunction** // create\_object

endclass

endpackage // factory\_pkg

**test\_program.sv**

import factory\_pkg**::\*;**

import package\_test**::\*;**

import packet\_pkg**::\*;**

`include "TestGood.sv"

`include "Test\_v3.sv"

`include "TestBad.sv"

program test**;**

**initial** **begin**

component c**;**

factory**::**printFactory**();**

c **=** factory**::**get\_test**();**

c**.**run\_test**();**

**end**

endprogram // test

**TestGood.sv**

class TestGood extends component**;**

typedef registry **#(**TestGood**,** "TestGood"**)** type\_id**;**

Environment env**;**

virtual **task** run\_test**();**

Driver\_cbs\_scoreboard sb\_callback**;**

$display**(**"Running Good Test:"**);**

env **=** new**(**1000**);**

env**.**build**();**

**begin**

sb\_callback **=** new**();**

env**.**drv**.**cbs**.**push\_back**(**sb\_callback**);**

**end**

env**.**run**();**

$display**(**"%0t:Test Finished with %d comparisons."**,**$time**,** sb\_callback**.**scb**.**num\_compared**);**

$stop**;**

**endtask** // run\_test

endclass // TestGood

**TestBad.sv**

class bad\_packet extends packet**;**

static int corruption\_cnt**;**

virtual **function** automatic void calc\_header\_checksum**();**

int corrupt**;**

corrupt **=** $urandom\_range**(**0**,**99**);**

header**.**calc\_header\_checksum**();**

**if(**corrupt **<** 2**)** **begin**

header**.**header\_checksum **=** **~**header**.**header\_checksum**;**

corruption\_cnt**++;**

**end**

**endfunction**

endclass // bad\_packet

class TestBad extends component**;**

typedef registry **#(**TestBad**,** "TestBad"**)** type\_id**;**

Environment env**;**

virtual **task** run\_test**();**

Driver\_cbs\_scoreboard sb\_callback**;**

bad\_packet badp**;**

$display**(**"Running Bad Test:"**);**

env **=** new**(**1000**);**

env**.**build**();**

**begin**

sb\_callback **=** new**();**

badp **=** new**();**

env**.**gen**.**blueprint **=** badp**;**

env**.**drv**.**cbs**.**push\_back**(**sb\_callback**);**

**end**

env**.**run**();**

$display**(**"%0t:Send %0d Bad Packets"**,**$time**,** bad\_packet**::**corruption\_cnt**);**

$display**(**"%0t:Test Finished with %d comparisons."**,**$time**,** sb\_callback**.**scb**.**num\_compared**);**

**endtask** // run\_test

endclass // TestBad

**Test\_v3.sv**

class Test\_v3 extends component**;**

typedef registry **#(**Test\_v3**,** "Test\_v3"**)** type\_id**;**

Environment env**;**

virtual **task** run\_test**();**

Driver\_cbs\_scoreboard sb\_callback**;**

Driver\_cbs\_v3 dcs**;**

$display**(**"Running v3 Test:"**);**

env **=** new**(**1000**);**

env**.**build**();**

**begin**

dcs **=** new**();**

sb\_callback **=** new**();**

env**.**drv**.**cbs**.**push\_back**(**dcs**);**

env**.**drv**.**cbs**.**push\_back**(**sb\_callback**);**

**end**

env**.**run**();**

$display**(**"%0t: Corrupted %0d Packets"**,**$time**,** dcs**.**corruption\_cnt**);**

$display**(**"%0t:Test Finished with %d comparisons."**,**$time**,** sb\_callback**.**scb**.**num\_compared**);**

$stop**;**

**endtask** // run\_test

endclass // Test\_v3