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
```