Joshua S. Monson
ECEn 620
Assignment 7.1

Overview

The purpose of this assignment was to learn how to use threads in system Verilog. The test bench created in this assignment validates an arbiter module. For this test bench I create a gold reference design and instantiated it along-side a design under test (DUT). The outputs (grant signals) of the golden and the DUT are compared on the negative edge of the clock. Initially, I believed that there might be a bug in the dut; however, I ran the test bench for more than 100,000 clock cycles of the randomly generated data and even tried constraining the randomness such that requests were almost always enabled and requesting more often than not. Even with these efforts I did not find a bug. So, the design is either bug free or both my golden and the dut have the same bug.

The remainder of this report contains:

1. Source Code for top, package, golden, and test.
2. Waveform of showing directed test and 10 cycles of random test.
3. Transcripts showing testbench operation with and without errors.

Source Code for Top

```
`include "arb_if.sv"
module top();
   //Clock Generator
   integer errors = 0;

   bit clk;
   wire rst;
   always #10ns clk = ~clk;

   arb_if arb0(), arb1(), arb2();
   arb_if darb0(), darb1(), darb2();

   test  t(clk,
           rst,
           arb0.Test,
           arb1.Test,
           arb2.Test,
         darb0.Test,
         darb1.Test,
         darb2.Test,
         errors);

   golden_arb golden(.clk(clk),
                     .reset(rst),
                     .req0(arb0.req),
                     .req1(arb1.req),
                 .req2(arb2.req),
                 .en0(arb0.en),
                 .en1(arb1.en),
                 .en2(arb2.en),
                 .grant0(arb0.grant),
                 .grant1(arb1.grant),
                 .grant2(arb2.grant)
                );

   arbiter arb(.clk(clk),
               .reset(rst),
                     .req0(darb0.req),
                     .req1(darb1.req),
                 .req2(darb2.req),
                 .en0(darb0.en),
                 .en1(darb1.en),
                 .en2(darb2.en),
                 .grant0(darb0.grant),
                 .grant1(darb1.grant),
                 .grant2(darb2.grant)
                );
   always @ (negedge clk) begin
      if(arb0.grant != darb0.grant) begin
      $display("@%0d: Grant0 Mismatch!!", $time);
       errors++;
      end
      if(arb1.grant != darb1.grant) begin
      $display("@%0d: Grant1 Mismatch!!", $time);
       errors++;
      end
      if(arb2.grant != darb2.grant) begin
      $display("@%0d: Grant2 Mismatch!!", $time);
       errors++;
      end
   end


endmodule // top
```

Source Code for Test

```
import tb_pkg::*;

`define SV_RAND_CHECK(r) \
   do begin \
      if (!(r)) begin \
       $display("%s:%0d: Randomization failed \"%s\"", \
              `__FILE__, `__LINE__, `"r`");\
       $finish; \
      end \
   end while(0)

program automatic test(input bit clk,
                       output bit    rst,
                             arb_if a0,
                             arb_if a1,
                             arb_if a2,
                             arb_if da0,
                             arb_if da1,
                             arb_if da2,

                   input integer errors);

  initial begin
     Transaction t0, t1, t2;
     int n = 100;

     //*************************
     //      Directed Test
     //*************************
     rst = 1;
     da0.req = 0;
     da1.req = 0;
     da2.req = 0;
     da0.en = 0;
     da1.en = 0;
     da2.en = 0;
     a0.req = 0;
     a1.req = 0;
     a2.req = 0;
     a0.en = 0;
     a1.en = 0;
     a2.en = 0;
     @(posedge clk);
     @(negedge clk);
     rst = 0;
     @(posedge clk);
     da0.req = 1;
     da1.req = 1;
     da2.req = 1;
     da0.en = 1;
     da1.en = 1;
     da2.en = 1;
     a0.req = 1;
     a1.req = 1;
     a2.req = 1;
     a0.en = 1;
     a1.en = 1;
     a2.en = 1;
     @(posedge clk);
     @(posedge clk);
     @(posedge clk);
     da0.req = 0;
     a0.req = 0;
     @(posedge clk);
     @(posedge clk);
     @(posedge clk);
     da1.req = 0;
     da2.req = 0;
     a1.req = 0;
     a2.req = 0;
```

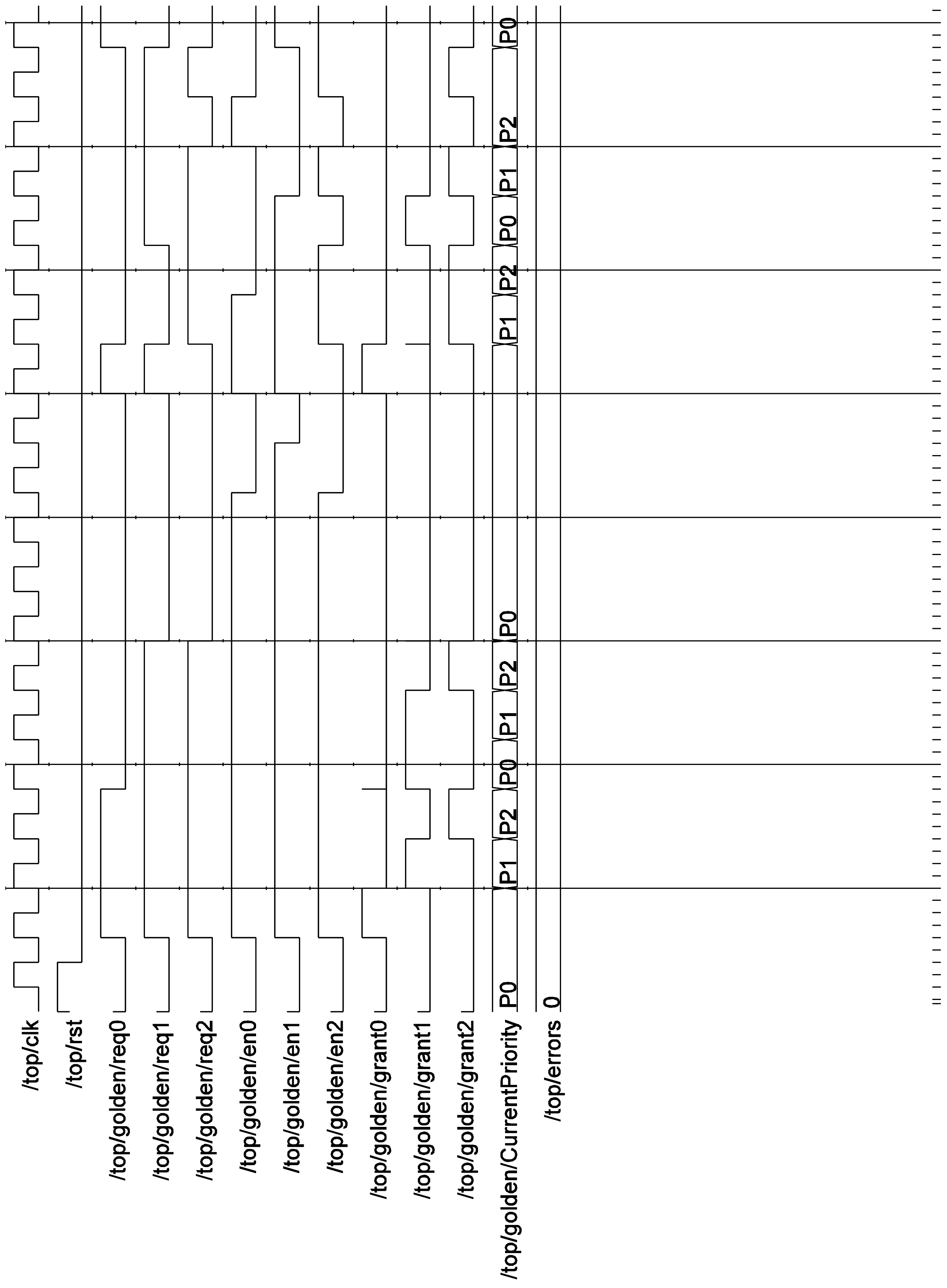
```
      @(posedge clk);
      @(posedge clk);
      @(posedge clk);

      //*************************
      //      Random Tests
      //*************************

      fork
       repeat (n) begin
          t0 = new();
          `SV_RAND_CHECK(t0.randomize());
          a0.req = t0.req;
          a0.en  = t0.en;
          da0.req= t0.req;
          da0.en = t0.en;
          @(posedge clk);
       end
       repeat (n) begin
          t1 = new();
          `SV_RAND_CHECK(t1.randomize());
          a1.req = t1.req;
          a1.en  = t1.en;
          da1.req= t1.req;
          da1.en = t1.en;
          @(posedge clk);
       end
       repeat (n) begin
          t2 = new();
          `SV_RAND_CHECK(t2.randomize());
          a2.req = t2.req;
          a2.en  = t2.en;
          da2.req= t2.req;
          da2.en = t2.en;
          @(posedge clk);
       end
      join

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

      $display("Ran for %0d cycles", n);

      $display("Found %0d Errors.", errors);

  end

endprogram
```

Source Code for Package

```
package tb_pkg;

class Transaction;
   rand bit req;
   rand bit en;

   /*constraint creq_dist {
      req dist { 0 :/ 30, 1 :/ 70};
   }

   constraint cen_dist {
      en dist { 0 :/ 5, 1 :/95 };
   }*/

endclass // Transaction

endpackage
```

Source Code for Golden

```
// Model for a 3-port arbiter using a round robin approach
// to issuing grant.
// 1 Priority is maintained in a round robin format, i.e. port 0->port 1-
>port 2->port 0â€¦
// 2 At reset the priority is port 0.
// 3 Priority will be incremented whenever any grant is asserted
// 4 Grant is combinatorial
// 5 If a port with priority is not enabled or not currently requesting the
next port will
// be given the grant given that port is enabled and requesting the bus, and
so on.
`default_nettype none
 module golden_arb( input wire clk,
                      input wire reset,
                    input wire req0,
                    input wire req1,
                    input wire req2,
                    input wire en0,
                    input wire en1,
                    input wire en2,
                    output reg grant0,
                    output reg grant1,
                    output reg grant2);

   typedef enum       { P0, P1, P2 } PRIORITY;

   PRIORITY CurrentPriority, NextPriority;
   logic              grant_asserted;

   //Grant Logic
   always_comb begin
      grant0 = 0;
      grant1 = 0;
      grant2 = 0;
      case (CurrentPriority)
      P0:
        if(req0 && en0)
          grant0 = 1;
        else if (req1 && en1)
          grant1 = 1;
        else if (req2 && en2)
          grant2 = 1;
      P1:
        if (req1 && en1)
          grant1 = 1;
        else if (req2 && en2)
          grant2 = 1;
        else if(req0 && en0)
          grant0 = 1;
      P2:
        if (req2 && en2)
          grant2 = 1;
        else if(req0 && en0)
          grant0 = 1;
        else if (req1 && en1)
          grant1 = 1;
      endcase
   end

   //Next Priority logic
   always_comb begin
    grant_asserted = (grant0 || grant1 || grant2);
    unique case (CurrentPriority)
      P0:
      if(grant_asserted)
        NextPriority = P1;
        else
        NextPriority = P0;
      P1:
      if(grant_asserted)
        NextPriority = P2;
        else
        NextPriority = P1;
      P2:
      if(grant_asserted)
        NextPriority = P0;
        else
        NextPriority = P2;
    endcase
   end

   //Priority Register
   always_ff @(posedge clk iff reset == 0 or posedge reset) begin
    if (reset == 1)
      CurrentPriority <= P0;
    else
      CurrentPriority <= NextPriority;
   end
endmodule
```


/top/clk
/top/rst
/top/golden/req0
/top/golden/req1
/top/golden/req2
/top/golden/en0
/top/golden/en1
/top/golden/en2
/top/golden/grant0
/top/golden/grant1
/top/golden/grant2
/top/golden/CurrentPriority
P0
P1
P2
/top/errors
0

Transcript Without Errors

```
# view wave
# .main_pane.wave.interior.cs.body.pw.wf
# do wave.do
# run -all
# Ran for 100 cycles
# Found 0 Errors.
# 1
# Simulation stop requested.
```

Transcript With Errors

```
# run -all
# @1360: Grant0 Mismatch!!
# @1360: Grant2 Mismatch!!
# @1440: Grant1 Mismatch!!
# @1440: Grant2 Mismatch!!
# @1480: Grant0 Mismatch!!
# @1480: Grant2 Mismatch!!
# @1700: Grant0 Mismatch!!
# @1700: Grant2 Mismatch!!
# @1720: Grant0 Mismatch!!
# @1720: Grant1 Mismatch!!
# @1740: Grant0 Mismatch!!
# @1740: Grant1 Mismatch!!
# @1800: Grant0 Mismatch!!
# @1800: Grant2 Mismatch!!
# @1820: Grant0 Mismatch!!
# @1820: Grant2 Mismatch!!
# @1860: Grant1 Mismatch!!
# @1860: Grant2 Mismatch!!
# @1880: Grant0 Mismatch!!
# @1880: Grant1 Mismatch!!
# @1940: Grant0 Mismatch!!
# @1940: Grant2 Mismatch!!
# @2040: Grant0 Mismatch!!
# @2040: Grant2 Mismatch!!
# Ran for 100 cycles
# Found 24 Errors.
```