Digital IC Verification

Project One

Name : mohamed samy fathy

Under supervision of:

Eng . Hassan Khaled

1 - Snippets for the implemented code.

//TOP

```
`include "interface.sv"
`include "axi4.sv"
`include "Testbench.sv"

module TOP();
    parameter   DATA_WIDTH   = 32;
    parameter   ADDR_WIDTH   = 16;
    parameter   MEMORY_DEPTH = 1024;

    bit ACLK = 0;
    always #10 ACLK = ~ACLK;


    axi4_if     #(DATA_WIDTH,ADDR_WIDTH,MEMORY_DEPTH)   axi4if(ACLK);
    axi4        #(DATA_WIDTH,ADDR_WIDTH,MEMORY_DEPTH)   D1(axi4if.DUT);
    axi4_TB     #(DATA_WIDTH,ADDR_WIDTH,MEMORY_DEPTH)   T1(axi4if.TEST);
    assertion                                           A1(axi4if.ASSERT);


endmodule

```

// interface

```
interface axi4_if #(
    parameter DATA_WIDTH = 32,
    parameter ADDR_WIDTH = 16,
    parameter MEMORY_DEPTH = 1024
)(input bit ACLK);


    logic                   ARESETn;

    logic [ADDR_WIDTH-1:0]  AWADDR, ARADDR;
    logic [7:0]             AWLEN, ARLEN;
    logic [2:0]             AWSIZE, ARSIZE;
    logic                   AWVALID, AWREADY;

    logic [DATA_WIDTH-1:0]  WDATA, RDATA;
    logic                   WVALID, WREADY, WLAST;

    logic [1:0]             BRESP, RRESP;
    logic                   BVALID, BREADY ;
    logic                   ARVALID, ARREADY;
    logic                   RVALID, RLAST, RREADY;


    clocking cb @(posedge ACLK);
        default input #1step output negedge;
            output AWADDR,AWLEN,AWSIZE,AWVALID,WDATA,WVALID,WLAST,BREADY,ARADDR,ARLEN,ARSIZE,RREADY,ARVALID;
            input  AWREADY,WREADY,BRESP,BVALID,ARREADY,RDATA,RRESP,RLAST,RVALID ;
    endclocking


    modport TEST(clocking cb ,input ACLK, output ARESETn);

    modport DUT(input ACLK,ARESETn,AWADDR,AWLEN,AWSIZE,AWVALID,WDATA,WVALID,WLAST,BREADY,ARADDR,ARLEN,ARSIZE,RREADY,ARVALID,
                output AWREADY,WREADY,BRESP,BVALID,ARREADY,RDATA,RRESP,RLAST,RVALID);


    modport ASSERT(input ACLK,ARESETn,AWADDR,AWLEN,AWSIZE,AWVALID,WDATA,WVALID,WLAST,BREADY,ARADDR,ARLEN,ARSIZE,RREADY,ARVALID,
                   AWREADY,WREADY,BRESP,BVALID,ARREADY,RDATA,RRESP,RLAST,RVALID);

    endinterface

```

// Testbench

```
`timescale 1ns/1ps
`include "axi4_packet.sv"

module axi4_TB #(
    parameter DATA_WIDTH = 32,
    parameter ADDR_WIDTH = 16,
    parameter MEMORY_DEPTH = 1024
) (axi4_if.TEST axi4if);

    // // // // // // //  Local Variables and Objects  // // // // // // // // //

    axi4_packet pkt;

    // Queues for write operation results
    logic [1:0] Wactual_queue[$]; logic [1:0] Wexpected_queue[$];

    // Queues for read operation results
    logic [1:0] Ractual_queue[$]; logic [1:0] Rexpected_queue[$];


    initial begin
        pkt = new();
        pkt.clk = axi4if.ACLK;


        axi4if.ARESETn = 0;


        axi4if.cb.AWADDR  <= 0;
        axi4if.cb.AWLEN   <= 0;
        axi4if.cb.AWSIZE  <= 2;
        axi4if.cb.AWVALID <= 0;

        axi4if.cb.WDATA   <= 0;
        axi4if.cb.WVALID  <= 0;
        axi4if.cb.WLAST   <= 0;

        axi4if.cb.BREADY  <= 0;

        axi4if.cb.ARADDR  <= 0;
        axi4if.cb.ARLEN   <= 0;
        axi4if.cb.ARSIZE  <= 2;
        axi4if.cb.ARVALID <= 0;

        axi4if.cb.RREADY  <= 0;

        #15;
        axi4if.ARESETn = 1;
```

```
        // WRITE Transactions

        repeat (3000) begin
            generate_stimulus_w(pkt);
            drive_stim_W(pkt);
        end


        // READ Transactions

        repeat (3000) begin
            generate_stimulus_r(pkt);
            drive_stim_r(pkt);
        end

        $assertoff;

        @(posedge axi4if.ACLK);

        repeat(200)begin
            run_all_coverage_cornar_tests();
        end

        drive_stim_W_IDEL(pkt);

        $asserton;

        #50;
        axi4if.ARESETn = 0;
        #50;
        $display("AXI4 Testbench Completed Successfully");
        $stop;
    end


//////////  WRITE OPERATION TASKS ///////////////


    task automatic generate_stimulus_w(ref axi4_packet pkt);
        assert(pkt.randomize()) else $fatal("Randomization failed for WRITE packet");

    endtask


    task automatic drive_stim_W(ref axi4_packet pkt);
        @(posedge axi4if.ACLK);
```

```
100
101         axi4if.cb.AWADDR  <= pkt.AWADDR_rand;
102         axi4if.cb.AWLEN   <= pkt.AWLEN_rand;
103         axi4if.cb.AWVALID <= 1;
104
105
106         @(posedge axi4if.ACLK);
107         wait (axi4if.cb.AWREADY == 1);
108
109
110         if (axi4if.cb.AWREADY == 1) begin
111             axi4if.cb.AWVALID <= 0;
112             @(posedge axi4if.ACLK);
113         end
114
115         //$display("[%0t] WRITE ADDR: 0x%0h | AWLEN=%0d", $time, pkt.AWADDR_rand, pkt.AWLEN_rand);
116
117
118         for (int beat = 0; beat <= pkt.AWLEN_rand; beat++) begin
119             axi4if.cb.WDATA  <= pkt.WDATA_rand[beat];
120             axi4if.cb.WVALID <= 1;
121             axi4if.cb.WLAST  <= (beat == pkt.AWLEN_rand);
122
123             pkt.cg.sample();
124
125             //$display("[%0t] WDATA[%0d] = %0h | WLAST = %0b",$time, beat, pkt.WDATA_rand[beat], axi4if.cb.WLAST);
126
127             @(posedge axi4if.ACLK);
128         end
129
130
131         axi4if.cb.BREADY <= 1;
132         do @(posedge axi4if.ACLK);
133         while (axi4if.cb.BVALID != 1);
134
135         $display("[%0t] Write Response Received BRESP = %0h",$time, axi4if.cb.BRESP);
136         collect_output_data_W(axi4if.cb.BRESP);
137         golden_model_W();
138         check_results_W();
139
140         @(posedge axi4if.ACLK);
141         axi4if.cb.BREADY <= 0;
142
143
144
145     endtask
146
147
148     task automatic golden_model_W();
149         Wexpected_queue.push_back('b00);
150     endtask
151
152     task automatic collect_output_data_W(input logic [1:0] Resp);
153         Wactual_queue.push_back(Resp);
154     endtask
155
156     task automatic check_results_W();
157         for (int i = 0; i < Wexpected_queue.size(); i++) begin
158             if (Wactual_queue[i] == Wexpected_queue[i])
159                 $display("[PASS] WRITE[%0d] Expected=%0b Got=%0b",i, Wexpected_queue[i], Wactual_queue[i]);
160             else
161                 $display("[FAIL] WRITE[%0d] Expected=%0b Got=%0b",i, Wexpected_queue[i], Wactual_queue[i]);
162         end
163
164         Wexpected_queue.delete();
165         Wactual_queue.delete();
166     endtask
167
168
169
170 //////////////  READ OPERATION TASKS ////////////////
171
172
173     task automatic generate_stimulus_r(ref axi4_packet pkt);
174         assert(pkt.randomize()) else $fatal("Randomization failed for READ packet");
175          pkt.cg.sample();
176     endtask
177
178     task automatic drive_stim_r(ref axi4_packet pkt);
179         @(posedge axi4if.ACLK);
180
181         axi4if.cb.ARADDR  <= pkt.ARADDR_rand;
182         axi4if.cb.ARLEN   <= pkt.ARLEN_rand;
183         axi4if.cb.ARVALID <= 1;
184
185
186         do @(posedge axi4if.ACLK);
187         while (axi4if.cb.ARREADY == 0);
188
189         axi4if.cb.ARVALID <= 0;
190         //$display("[%0t] READ ADDR: 0x%0h | ARLEN=%0d", $time, pkt.ARADDR_rand, pkt.ARLEN_rand);
191
192
193         axi4if.cb.RREADY <= 1;
194
195         do begin
```

```
196
197
198
199             wait (axi4if.cb.RVALID == 1);
200
201             @(posedge axi4if.ACLK);
202                 //$display("[%0t] READ DATA Received %0d ", $time, axi4if.cb.RDATA);
203                 pkt.cg.sample();
204
205         end while (axi4if.cb.RLAST == 0);
206
207         collect_output_data_R();
208         golden_model_R();
209         check_results_R();
210
211         //$display("[%0t] READ COMPLETE  Last Data Beat Received", $time);
212         axi4if.cb.RREADY <= 0;
213
214     endtask
215
216
217     task automatic golden_model_R();
218         Rexpected_queue.push_back('b00);
219     endtask
220
221     task automatic collect_output_data_R();
222         Ractual_queue.push_back(axi4if.cb.RRESP);
223     endtask
224
225     task automatic check_results_R();
226         for (int i = 0; i < Rexpected_queue.size(); i++) begin
227         if (Ractual_queue[i] == Rexpected_queue[i])
228             $display("[PASS] READ[%0d] Expected=%0b Got=%0b",i, Rexpected_queue[i], Ractual_queue[i]);
229         else
230             $display("[FAIL] READ[%0d] Expected=%0b Got=%0b", i, Rexpected_queue[i], Ractual_queue[i]);
231         end
232         Ractual_queue.delete();
233         Rexpected_queue.delete();
234     endtask
235
236
237     //////////  WRITE VIOLATION OPERATION TASKS //////////////
238
239     task drive_stim_write_boundary_violation();
240         $display("[%0t] >>> Starting WRITE boundary violation stimulus", $time);
241
242         axi4if.cb.AWADDR  <= 16'hFFF0;
243         axi4if.cb.AWLEN   <= 8'd20;
```

```
244          axi4if.cb.AWVALID <= 1;
245
246          // Wait until AWREADY goes high
247          @(posedge axi4if.ACLK);
248
249          axi4if.cb.AWVALID <= 0;
250          //$display("[%0t] WRITE ADDR: 0x%0h | AWLEN=%0d", $time, pkt.AWADDR_rand, pkt.AWLEN_rand);
251
252          for (int beat = 0; beat <= axi4if.cb.AWLEN ; beat++) begin
253              axi4if.cb.WDATA  <= $random;
254              axi4if.cb.WVALID <= 1;
255              axi4if.cb.WLAST  <= (beat == axi4if.cb.AWLEN );
256
257              @(posedge axi4if.ACLK);
258          end
259
260          axi4if.cb.BREADY <= 1;
261          @(posedge axi4if.ACLK);
262
263          $display("[%0t] Write Response Received BRESP = %0h",$time, axi4if.cb.BRESP);
264
265          @(posedge axi4if.ACLK);
266          axi4if.cb.BREADY <= 0;
267
268          $display("[%0t] <<< WRITE boundary violation complete", $time);
269      endtask
270
271
272      task drive_stim_invalid_write_addr();
273          $display("[%0t] >>> Starting INVALID WRITE address stimulus", $time);
274
275          @(posedge axi4if.ACLK);
276
277          axi4if.cb.AWADDR  <= 16'h5000;
278          axi4if.cb.AWLEN   <= 8'd4;
279          axi4if.cb.AWVALID <= 1;
280
281          @(posedge axi4if.ACLK);
282
283          axi4if.cb.AWVALID <= 0;
284          //$display("[%0t] WRITE ADDR: 0x%0h | AWLEN=%0d", $time, pkt.AWADDR_rand, pkt.AWLEN_rand);
285
286
287          for (int beat = 0; beat <= axi4if.cb.AWLEN ; beat++) begin
288              axi4if.cb.WDATA  <= $random;
289              axi4if.cb.WVALID <= 1;
290              axi4if.cb.WLAST  <= (beat == axi4if.cb.AWLEN );
291
```

```
292             @(posedge axi4if.ACLK);
293         end
294
295         axi4if.cb.BREADY <= 1;
296
297         $display("[%0t] Write Response Received BRESP = %0h",$time, axi4if.cb.BRESP);
298
299         @(posedge axi4if.ACLK);
300         axi4if.cb.BREADY <= 0;
301
302         $display("[%0t] <<< INVALID WRITE address complete", $time);
303     endtask
304
305
306     task drive_stim_wlast_missing();
307         $display("[%0t] >>> Starting WLAST missing stimulus", $time);
308
309             @(posedge axi4if.ACLK);
310
311             axi4if.cb.AWADDR  <= 16'h0100;
312             axi4if.cb.AWLEN   <= 8'd4;
313             axi4if.cb.AWVALID <= 1;
314
315             @(posedge axi4if.ACLK);
316             wait (axi4if.cb.AWREADY == 1);
317
318             axi4if.cb.AWVALID <= 0;
319             //$display("[%0t] WRITE ADDR: 0x%0h | AWLEN=%0d", $time, pkt.AWADDR_rand, pkt.AWLEN_rand);
320
321             for (int beat = 0; beat <= axi4if.cb.AWLEN ; beat++) begin
322                 axi4if.cb.WDATA  <= $random;
323                 axi4if.cb.WVALID <= 1;
324                 //axi4if.cb.WLAST  <= (beat == axi4if.cb.AWLEN );
325
326                 @(posedge axi4if.ACLK);
327             end
328
329             axi4if.cb.BREADY <= 1;
330             do @(posedge axi4if.ACLK);
331             while (axi4if.cb.BVALID != 1);
332
333             $display("[%0t] Write Response Received BRESP = %0h",$time, axi4if.cb.BRESP);
334
335             @(posedge axi4if.ACLK);
336             axi4if.cb.BREADY <= 0;
337
338         $display("[%0t] <<< WLAST missing stimulus complete", $time);
339     endtask

341
342     task drive_stim_bready_hold_low();
343         $display("[%0t] >>> Starting BREADY toggle stimulus", $time);
344
345         @(posedge axi4if.ACLK);
346
347         axi4if.cb.AWADDR  <= 16'h0100;
348         axi4if.cb.AWLEN   <= 8'd4;
349         axi4if.cb.AWVALID <= 1;
350
351         @(posedge axi4if.ACLK);
352         wait (axi4if.cb.AWREADY == 1);
353
354         axi4if.cb.AWVALID <= 0;
355         //$display("[%0t] WRITE ADDR: 0x%0h | AWLEN=%0d", $time, pkt.AWADDR_rand, pkt.AWLEN_rand);
356
357         for (int beat = 0; beat <= axi4if.cb.AWLEN ; beat++) begin
358             axi4if.cb.WDATA  <= $random;
359             axi4if.cb.WVALID <= 1;
360             axi4if.cb.WLAST  <= (beat == axi4if.cb.AWLEN );
361
362             @(posedge axi4if.ACLK);
363         end
364
365         axi4if.cb.BREADY <= 1;
366         do  begin
367         @(negedge axi4if.ACLK);
368         axi4if.cb.BREADY <= ~axi4if.cb.BREADY;
369         end while (axi4if.cb.BVALID != 1);
370
371         $display("[%0t] Write Response Received BRESP = %0h",$time, axi4if.cb.BRESP);
372
373         @(posedge axi4if.ACLK);
374         axi4if.cb.BREADY <= 0;
375
376         $display("[%0t] <<< BREADY toggle stimulus complete", $time);
377     endtask
378
379
380
381
382
383     task drive_stim_write_107();
384         $display("[%0t] >>> Starting WRITE boundary violation stimulus", $time);
385
386         @(posedge axi4if.ACLK);
387
388         axi4if.cb.AWADDR  <= 16'h0FFF;
```

```
        axi4if.cb.AWLEN   <= 8'd20;
        axi4if.cb.AWVALID <= 0;

        @(posedge axi4if.ACLK);

        axi4if.cb.AWVALID <= 0;
        //$display("[%0t] WRITE ADDR: 0x%0h | AWLEN=%0d", $time, pkt.AWADDR_rand, pkt.AWLEN_rand);

        for (int beat = 0; beat <= axi4if.cb.AWLEN ; beat++) begin
            axi4if.cb.WDATA  <= $random;
            axi4if.cb.WVALID <= 1;
            axi4if.cb.WLAST  <= (beat == axi4if.cb.AWLEN );

            @(posedge axi4if.ACLK);
        end

        axi4if.cb.BREADY <= 1;

        $display("[%0t] Write Response Received BRESP = %0h",$time, axi4if.cb.BRESP);

        @(posedge axi4if.ACLK);
        axi4if.cb.BREADY <= 0;

        $display("[%0t] <<< WRITE boundary violation complete", $time);
    endtask


    //////////////  READ VIOLATION OPERATION TASKS ////////////////

    task drive_stim_read_boundary_violation();
        $display("[%0t] >>> Starting READ boundary violation stimulus", $time);

        axi4if.cb.ARADDR  <= 16'hFFF0;
        axi4if.cb.ARLEN   <=  8'd20;
        axi4if.cb.ARVALID <= 1;

         @(posedge axi4if.ACLK);

        axi4if.cb.ARVALID <= 0;
        //$display("[%0t] READ ADDR: 0x%0h | ARLEN=%0d", $time, pkt.ARADDR_rand, pkt.ARLEN_rand);

        axi4if.cb.RREADY <= 1;

        //$display("[%0t] READ COMPLETE  Last Data Beat Received", $time);
        axi4if.cb.RREADY <= 0;

        $display("[%0t] <<< READ boundary violation complete", $time);
    endtask
```

```
    task drive_stim_invalid_read_addr();
        $display("[%0t] >>> Starting INVALID READ address stimulus", $time);

        axi4if.cb.ARADDR  <= 16'h5000;
        axi4if.cb.ARLEN   <=  8'd8;
        axi4if.cb.ARVALID <= 1;

        @(posedge axi4if.ACLK);

        axi4if.cb.ARVALID <= 0;
        //$display("[%0t] READ ADDR: 0x%0h | ARLEN=%0d", $time, pkt.ARADDR_rand, pkt.ARLEN_rand);
        axi4if.cb.RREADY <= 1;

        //$display("[%0t] READ COMPLETE  Last Data Beat Received", $time);
        axi4if.cb.RREADY <= 0;

        $display("[%0t] <<< RREADY toggle stimulus complete", $time);
    endtask


    task drive_stim_rready_hold_low();
        $display("[%0t] >>> Starting BREADY toggle stimulus", $time);


        axi4if.cb.ARADDR  <= 16'h1500;
        axi4if.cb.ARLEN   <=  8'd8;
        axi4if.cb.ARVALID <= 1;

        do @(posedge axi4if.ACLK);
        while (axi4if.cb.ARREADY == 0);

        axi4if.cb.ARVALID <= 0;
        //$display("[%0t] READ ADDR: 0x%0h | ARLEN=%0d", $time, pkt.ARADDR_rand, pkt.ARLEN_rand);

        axi4if.cb.RREADY <= 1;

        do begin
        @(negedge axi4if.ACLK);
        axi4if.cb.RREADY <= ~axi4if.cb.RREADY;
        wait (axi4if.cb.RVALID == 1);

        @(posedge axi4if.ACLK);
        //$display("[%0t] READ DATA Received %0d ", $time, axi4if.cb.RDATA);
        pkt.cg.sample();
        end while (axi4if.cb.RLAST == 0);

        //$display("[%0t] READ COMPLETE  Last Data Beat Received", $time);
```

```
    task run_all_coverage_cornar_tests();
        $display("[%0t] >>> RUNNING ENHANCED COVERAGE TESTS", $time);

        drive_stim_write_boundary_violation();
        drive_stim_read_boundary_violation();
        drive_stim_invalid_write_addr();
        drive_stim_invalid_read_addr();
        drive_stim_wlast_missing();

        $display("[%0t] <<< ENHANCED COVERAGE TESTS COMPLETE", $time);
    endtask


    task automatic drive_stim_W_IDEL(ref axi4_packet pkt);
        @(posedge axi4if.ACLK);

        axi4if.cb.AWADDR  <= 'h02f0;
        axi4if.cb.AWLEN   <= 'd12;
        axi4if.cb.AWVALID <= 1;
        axi4if.cb.WVALID <= 0;

        @(posedge axi4if.ACLK);
        wait (axi4if.cb.AWREADY == 1);

        if (axi4if.cb.AWREADY == 1) begin
            axi4if.cb.AWVALID <= 0;
            @(posedge axi4if.ACLK);
        end

        axi4if.cb.AWVALID <= 0;
        //$display("[%0t] WRITE ADDR: 0x%0h | AWLEN=%0d", $time, pkt.AWADDR_rand, pkt.AWLEN_rand);


        for (int beat = 0; beat <= pkt.AWLEN_rand; beat++) begin
            axi4if.cb.WDATA  <= pkt.WDATA_rand[beat];
            axi4if.cb.WVALID <= 0;
            axi4if.cb.WLAST  <= (beat == pkt.AWLEN_rand);
            //$display("[%0t] WDATA[%0d] = %0h | WLAST = %0b",$time, beat, pkt.WDATA_rand[beat], axi4if.cb.WLAST);
            @(posedge axi4if.ACLK);
        end



    endtask

endmodule
```

//axi4_packet

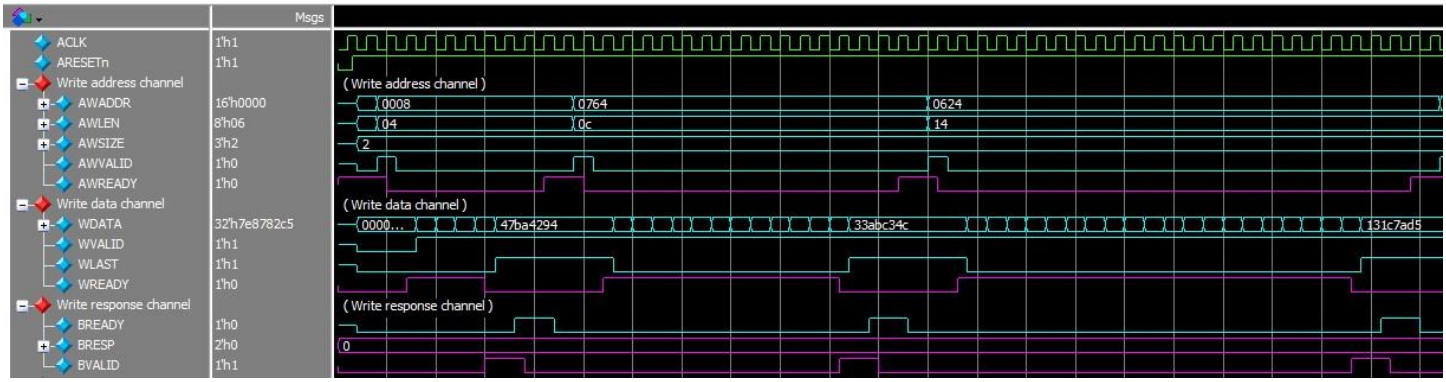
```
class axi4_packet;

    localparam DATA_WIDTH   = 32;
    localparam ADDR_WIDTH   = 16;
    localparam MEMORY_DEPTH = 1024;
    localparam NUM_OF_Bytes_PER_BEAT  = 4;

    bit    clk;
    rand logic  [ADDR_WIDTH-1:0] AWADDR_rand, ARADDR_rand;
    rand logic [7:0]             AWLEN_rand, ARLEN_rand;
    rand logic [DATA_WIDTH-1:0] WDATA_rand[];


    constraint len_c {
        AWLEN_rand inside {[0:20]};
        WDATA_rand.size() == (AWLEN_rand + 1);
    }


    constraint WADDR_C {

        (AWADDR_rand % NUM_OF_Bytes_PER_BEAT) == 0;
        AWADDR_rand >= 0;
        AWADDR_rand <= (MEMORY_DEPTH*NUM_OF_Bytes_PER_BEAT - (AWLEN_rand + 1) * NUM_OF_Bytes_PER_BEAT);
       // ((AWADDR_rand & 'hFFFF) + ((AWLEN_rand + 1) * NUM_OF_Bytes_PER_BEAT)) <= 12'h1000;
    }


    constraint ARLEN_C {
        ARLEN_rand <= AWLEN_rand;
    }


    constraint RADDR_C {
        (ARADDR_rand % NUM_OF_Bytes_PER_BEAT) == 0;
        ARADDR_rand >= AWADDR_rand;
        ARADDR_rand <= AWADDR_rand + (AWLEN_rand * NUM_OF_Bytes_PER_BEAT);
    }

    constraint data_c {
        foreach (WDATA_rand[i]) {
            WDATA_rand[i] dist { '0 := 1, '1 := 1, [1:(2**DATA_WIDTH - 2)] := 8 };
        }
    }

    // covergroup
    covergroup cg;
        cp_addr : coverpoint AWADDR_rand {
            bins low_addr = { [0:255] };
            bins mid_addr = { [256:2047] };
            bins high_addr = { [2048:(MEMORY_DEPTH*NUM_OF_Bytes_PER_BEAT - 1)] };
        }

        cp_awlen : coverpoint AWLEN_rand {
            bins len_short = { [0:3] };
            bins len_medium = { [4:10] };
            bins len_long = { [11:20] };
        }

        cp_raddr : coverpoint ARADDR_rand {
            bins low_addr = { [0:255] };
            bins mid_addr = { [256:2047] };
            bins high_addr = { [2048:(MEMORY_DEPTH*NUM_OF_Bytes_PER_BEAT - 1)] };
        }

        cp_arlen : coverpoint ARLEN_rand {
            bins len_short = { [0:3] };
            bins len_medium = { [4:10] };
            bins len_long = { [11:20] };
        }
    endgroup

    function new();
        cg = new();
    endfunction

endclass
```

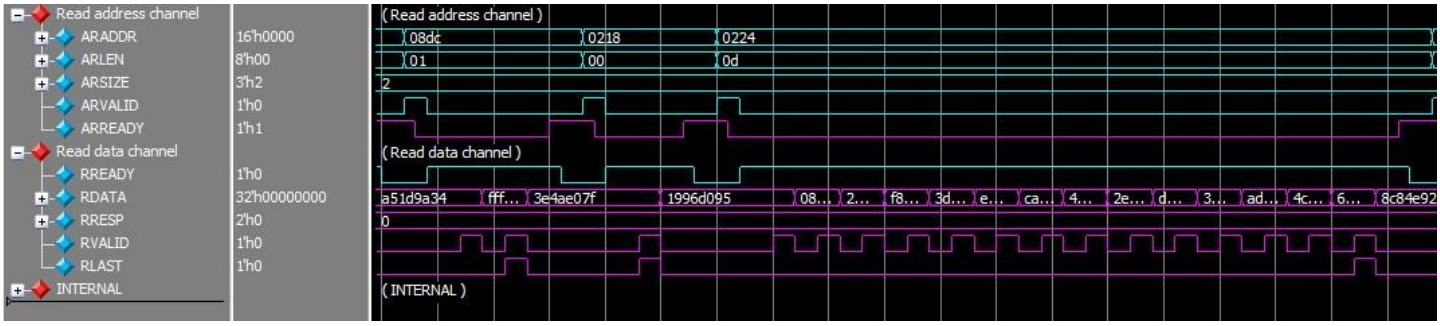
et

// Assertion

```
1   module assertion(axi4_if.ASSERT axi4if);
2
3
4       // WRITE RESPONSE ASSERTION
5        property write_response_ok;
6           @(posedge axi4if.ACLK)
7           disable iff (!axi4if.ARESETn)
8               (axi4if.BVALID && axi4if.BREADY) |=> (axi4if.BRESP == 2'b00);
9       endproperty
10
11      A1 : assert property(write_response_ok)
12          $display("[%0t] WRITE RESPONSE PASSED : BRESP = %0b", $time, axi4if.BRESP);
13      C1 : cover property(write_response_ok);
14
15
16
17      // READ RESPONSE ASSERTION
18      property read_response_ok;
19          @(posedge axi4if.ACLK)
20          disable iff (!axi4if.ARESETn)
21          (axi4if.RLAST && axi4if.RREADY) |=> (axi4if.RRESP == 2'b00);
22      endproperty
23
24      A2 : assert property(read_response_ok)
25          $display("[%0t] READ RESPONSE PASSED : RRESP = %0b", $time, axi4if.RRESP);
26      C2 : cover property(read_response_ok);
27
28  endmodule
```

2 – Snippets for the waveform shows the values for created variables





3-Snippets for Functional Coverage achieving

```
882  ============================================================================
883  === Instance: /Testbench_sv_unit
884  === Design Unit: work.Testbench_sv_unit
885  ============================================================================
886
887  Covergroup Coverage:
888      Covergroups                     1        na        na   100.00%
889          Coverpoints/Crosses         4        na        na        na
890              Covergroup Bins        12        12         0   100.00%
891  ----------------------------------------------------------------------------------------------------------------
892  Covergroup                                          Metric       Goal       Bins    Status
893
894  ----------------------------------------------------------------------------------------------------------------
895   TYPE /Testbench_sv_unit/axi4_packet/cg             100.00%        100          -    Covered
896      covered/total bins:                                  12         12          -
897      missing/total bins:                                   0         12          -
898      % Hit:                                          100.00%        100          -
899      Coverpoint cp_addr                              100.00%        100          -    Covered
900          covered/total bins:                               3          3          -
901          missing/total bins:                               0          3          -
902          % Hit:                                      100.00%        100          -
903          bin low_addr                                   4682          1          -    Covered
904          bin mid_addr                                  38163          1          -    Covered
905          bin high_addr                                 42500          1          -    Covered
906      Coverpoint cp_awlen                             100.00%        100          -    Covered
907          covered/total bins:                               3          3          -
908          missing/total bins:                               0          3          -
909          % Hit:                                      100.00%        100          -
910          bin len_short                                  2388          1          -    Covered
911          bin len_medium                                13666          1          -    Covered
912          bin len_long                                  69291          1          -    Covered
913      Coverpoint cp_raddr                             100.00%        100          -    Covered
914          covered/total bins:                               3          3          -
915          missing/total bins:                               0          3          -
916          % Hit:                                      100.00%        100          -
917          bin low_addr                                   4060          1          -    Covered
918          bin mid_addr                                  37999          1          -    Covered
919          bin high_addr                                 43286          1          -    Covered
920      Coverpoint cp_arlen                             100.00%        100          -    Covered
921          covered/total bins:                               3          3          -
922          missing/total bins:                               0          3          -
923          % Hit:                                      100.00%        100          -
924          bin len_short                                 15337          1          -    Covered
925          bin len_medium                                31119          1          -    Covered
926          bin len_long                                  38889          1          -    Covered
927  Statement Coverage:
928      Enabled Coverage              Bins      Hits    Misses  Coverage
929      ----------------              ----      ----    ------  --------
930      Statements                       1         1         0   100.00%
```

4-Snippets for Code Coverage achieving

```
================================================================================
=== Instance: /TOP/D1/mem_inst
=== Design Unit: work.axi4_memory
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         5         5         0   100.00%

================================Branch Details================================

Branch Coverage for instance /TOP/D1/mem_inst

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File axi_memory.v
------------------------------------IF Branch------------------------------------
    24                                     52453     Count coming in to IF
    24              1                          2
    26              1                      26225
                                           26226     All False Count
Branch totals: 3 hits of 3 branches = 100.00%

------------------------------------IF Branch------------------------------------
    27                                     26225     Count coming in to IF
    27              1                       3201
    29              1                      23024
Branch totals: 2 hits of 2 branches = 100.00%


Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                       4         4         0   100.00%

```

```
============================Statement Details============================

Statement Coverage for instance /TOP/D1/mem_inst --

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File axi_memory.v
    23              1                      52453
    25              1                          2
    28              1                       3201
    30              1                      23024
Toggle Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Toggles                        156       156         0   100.00%

============================Toggle Details============================

Toggle Coverage for instance /TOP/D1/mem_inst --

                                       Node      1H->0L      0L->1H  "Coverage"
                                       ----------------------------------------


Total Node Count     =         78
Toggled Node Count   =         78
Untoggled Node Count =          0

Toggle Coverage      =     100.00% (156 of 156 bins)

================================================================================
=== Instance: /TOP/D1
=== Design Unit: work.axi4
================================================================================
Branch Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Branches                         16        16         0   100.00%

============================Branch Details============================

Branch Coverage for instance /TOP/D1

    Line         Item                      Count     Source
    ----         ----                      -----     ------
  File axi4.sv
------------------------------------IF Branch------------------------------------
    69                                    121058     Count coming in to IF
    69              1                          5
    99              1                     121053
```

```
================================Condition Details================================

Condition Coverage for instance /TOP/D1 --

  File axi4.sv
----------------Focused Condition View-------------------
Line       151 Item    1  (axi4if.WLAST || ((write_burst_cnt - 1) == 0))
Condition totals: 2 of 2 input terms covered = 100.00%

----------------Focused Condition View-------------------
Line       235 Item    1  (read_burst_cnt > 0)
Condition totals: 1 of 1 input term covered = 100.00%


Expression Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Expressions                      1         1         0   100.00%

================================Expression Details================================

Expression Coverage for instance /TOP/D1 --

  File axi4.sv
----------------Focused Expression View-----------------
Line       229 Item    1  (read_burst_cnt == 0)
Expression totals: 1 of 1 input term covered = 100.00%


FSM Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    FSM States                       7         7         0   100.00%
    FSM Transitions                 10        10         0   100.00%

================================FSM Details================================

FSM Coverage for instance /TOP/D1 --

FSM_ID: write_state
    Current State Object : write_state
    ----------------------
    State Value MapInfo :
    ---------------------
Line           State Name               Value
----           ----------               -----
 107               W_IDLE                   0
 127               W_ADDR                   1
```

```
----------------------------------CASE Branch----------------------------------
    106                                 121053     Count coming in to CASE
    107             1                    74154
    127             1                     3601
    146             1                    36584
    168             1                     6714
Branch totals: 4 hits of 4 branches = 100.00%

----------------------------------IF Branch----------------------------------
    131                                   3601     Count coming in to IF
    139             1                      400
Branch totals: 1 hit of 1 branch = 100.00%

----------------------------------IF Branch----------------------------------
    148                                  36584     Count coming in to IF
    161             1                        1
Branch totals: 1 hit of 1 branch = 100.00%

----------------------------------IF Branch----------------------------------
    151                                  36583     Count coming in to IF
    151             1                     3200
    156             1                    33383
Branch totals: 2 hits of 2 branches = 100.00%

----------------------------------CASE Branch----------------------------------
    187                                 121053     Count coming in to CASE
    188             1                    71605
    208             1                     3400
    225             1                    46048
Branch totals: 3 hits of 3 branches = 100.00%

----------------------------------IF Branch----------------------------------
    210                                   3400     Count coming in to IF
    216             1                      400
Branch totals: 1 hit of 1 branch = 100.00%

----------------------------------IF Branch----------------------------------
    235                                  23024     Count coming in to IF
    235             1                    20024
    243             1                     3000
Branch totals: 2 hits of 2 branches = 100.00%


Condition Coverage:
    Enabled Coverage              Bins   Covered    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Conditions                       3         3         0   100.00%
```

```
 127              W_ADDR                   1
 146              W_DATA                   2
 168              W_RESP                   3
    Covered States :
    ----------------
                      State           Hit_count
                      -----           ---------
                     W_IDLE                3606
                     W_ADDR                3601
                     W_DATA                3201
                     W_RESP                3200
    Covered Transitions :
    --------------------
Line            Trans_ID           Hit_count          Transition
----            --------           ---------          ----------
 123                   0                3601          W_IDLE -> W_ADDR
 140                   1                 400          W_ADDR -> W_IDLE
 133                   2                3201          W_ADDR -> W_DATA
 162                   3                   1          W_DATA -> W_IDLE
 153                   4                3200          W_DATA -> W_RESP
 174                   5                3200          W_RESP -> W_IDLE


    Summary                       Bins      Hits    Misses  Coverage
    -------                       ----      ----    ------  --------
        FSM States                   4         4         0   100.00%
        FSM Transitions              6         6         0   100.00%
FSM_ID: read_state
    Current State Object : read_state
    ---------------------
    State Value MapInfo :
    ---------------------
Line          State Name               Value
----          ----------               -----
 188              R_IDLE                   0
 208              R_ADDR                   1
 225              R_DATA                   2
    Covered States :
    ----------------
                      State           Hit_count
                      -----           ---------
                     R_IDLE                3405
                     R_ADDR                3400
                     R_DATA                3000
    Covered Transitions :
    --------------------
Line            Trans_ID           Hit_count          Transition
----            --------           ---------          ----------
 204                   0                3400          R_IDLE -> R_ADDR
```

```
 204                   0                3400          R_IDLE -> R_ADDR
 217                   1                 400          R_ADDR -> R_IDLE
 214                   2                3000          R_ADDR -> R_DATA
 245                   3                3000          R_DATA -> R_IDLE


    Summary                       Bins      Hits    Misses  Coverage
    -------                       ----      ----    ------  --------
        FSM States                   3         3         0   100.00%
        FSM Transitions              4         4         0   100.00%
Statement Coverage:
    Enabled Coverage              Bins      Hits    Misses  Coverage
    ----------------              ----      ----    ------  --------
    Statements                      77        77         0   100.00%

================================Statement Details================================

Statement Coverage for instance /TOP/D1 --

    Line         Item                     Count     Source
    ----         ----                     -----     ------
  File axi4.sv
    68              1                    121058
    71              1                         5
    72              1                         5
    73              1                         5
    74              1                         5
    76              1                         5
    77              1                         5
    78              1                         5
    79              1                         5
    80              1                         5
    82              1                         5
    83              1                         5
    84              1                         5
    85              1                         5
    86              1                         5
    87              1                         5
    89              1                         5
    90              1                         5
    91              1                         5
    92              1                         5
    93              1                         5
    94              1                         5
    95              1                         5
    96              1                         5
    100             1                    121053
    101             1                    121053
    108             1                     74154
```

```
333      101            1                  121053
334      108            1                   74154
335      109            1                   74154
336      110            1                   74154
337      114            1                    3601
338      115            1                    3601
339      116            1                    3601
340      122            1                    3601
341      123            1                    3601
342      128            1                    3601
343      133            1                    3201
344      134            1                    3201
345      135            1                    3201
346      136            1                    3201
347      137            1                    3201
348      140            1                     400
349      141            1                     400
350      152            1                    3200
351      153            1                    3200
352      154            1                    3200
353      157            1                   33383
354      158            1                   33383
355      162            1                       1
356      163            1                       1
357      172            1                    3200
358      173            1                    3200
359      174            1                    3200
360      189            1                   71605
361      190            1                   71605
362      191            1                   71605
363      195            1                    3400
364      196            1                    3400
365      197            1                    3400
366      203            1                    3400
367      204            1                    3400
368      212            1                    3000
369      213            1                    3000
370      214            1                    3000
371      215            1                    3000
372      217            1                     400
373      218            1                     400
374      219            1                     400
375      227            1                   46048
376      228            1                   46048
377      229            1                   46048
378      233            1                   23024
379      236            1                   20024
380      237            1                   20024
381      240            1                   20024
```

```
384      245            1                    3000
385  Toggle Coverage:
386      Enabled Coverage              Bins      Hits    Misses  Coverage
387      ----------------              ----      ----    ------  --------
388      Toggles                          8         8         0   100.00%
389
390  =================================Toggle Details=================================
391
392  Toggle Coverage for instance /TOP/D1 --
393
394                                                    Node      1H->0L      0L->1H  "Coverage"
395                                                    ----------------------------------------
396
397  Total Node Count     =          4
398  Toggled Node Count   =          4
399  Untoggled Node Count =          0
400
401  Toggle Coverage      =     100.00% (8 of 8 bins)
402
```

5-Snippets for Assertion achieving

```
803  ================================================================================
804  === Instance: /TOP/A1
805  === Design Unit: work.assertion
806  ================================================================================
807
808  Assertion Coverage:
809      Assertions                        2         2         0   100.00%
810  --------------------------------------------------------------------
811  Name                 File(Line)                   Failure      Pass
812                                                    Count        Count
813  --------------------------------------------------------------------
814  /TOP/A1/A1           Assertion.sv(11)                0          1
815  /TOP/A1/A2           Assertion.sv(24)                0          1
816
817  Directive Coverage:
818      Directives                        2         2         0   100.00%
819
820  DIRECTIVE COVERAGE:
821  --------------------------------------------------------------------------------------------
822  Name                                  Design Design   Lang File(Line)      Hits Status
823                                        Unit   UnitType
824  --------------------------------------------------------------------------------------------
825  /TOP/A1/C1                            assertion Verilog  SVA  Assertion.sv(13)3000 Covered
826  /TOP/A1/C2                            assertion Verilog  SVA  Assertion.sv(26)3000 Covered
827
828  ================================================================================
```

6- Snippets for the logs show the all printed values.

```
# [PASS] READ[0] Expected=0 Got=0
# [1864950000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1865150000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1865350000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1865510000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1865670000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1866590000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1867190000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1867870000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1868110000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1868310000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1868430000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1868950000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1869110000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1869230000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1869470000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1869870000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1870390000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1870710000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1871230000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1871670000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1871830000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1872470000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1872710000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1873070000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1873190000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1873350000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1874070000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1874710000] READ RESPONSE PASSED : RRESP = 0
# [PASS] READ[0] Expected=0 Got=0
# [1875070000] READ RESPONSE PASSED : RRESP = 0
```