

يوسف أحمد محمد ابراهيم

Assignment 3
Extra

Question 1:

1. Design :

```
module tb1();
    int j;
    int q[$];

    initial begin
        j = 1;
        q = '{0, 2, 5};
        q.insert(1, j);
        $display("q = %p", q);
        q.delete(1);
        $display("q = %p", q);
        q.push_front(7);
        $display("q = %p", q);
        q.push_back(9);
        $display("q = %p", q);
        j = q.pop_front();
        $display("q = %p, j = %0d", q, j);
        j = q.pop_back();
        $display("q = %p, j = %0d", q, j);
        q.reverse();
        $display("reverse q = %p", q);
        q.rsort();
        $display("rsort q = %p", q);
        q.shuffle();
        $display("shuffle q = %p" , q);

    end
endmodule
```


2. Result:

```
# q = '{0, 1, 2, 5}'  
# q = '{0, 2, 5}'  
# q = '{7, 0, 2, 5}'  
# q = '{7, 0, 2, 5, 9}'  
# q = '{0, 2, 5, 9}', j = 7  
# q = '{0, 2, 5}', j = 9  
# reverse q = '{5, 2, 0}'  
# rsort q = '{5, 2, 0}'  
# shuffle q = '{2, 0, 5}'
```

Question 2:

1. RTL Design :

```
module adder (  
    input  clk,  
    input  reset,  
    input  signed [3:0] A, // Input data A in 2's complement  
    input  signed [3:0] B, // Input data B in 2's complement  
    output reg signed [4:0] C // Adder output in 2's complement  
);  
  
// Register output C  
always @(posedge clk or posedge reset) begin  
    if (reset)  
        C <= 5'b0;  
    else  
        C <= A + B;  
end  
endmodule
```


2. Testbench:

```
1  import adder_pkg::*;
2
3  module adder_tb();
4      bit signed [3:0] A;
5      bit signed [3:0] B;
6      bit signed [4:0] C;
7      bit          clk;
8      logic        rst;
9
10     integer error_count, correct_count;
11
12     localparam MAXPOS = 4'b0111;
13     localparam MAXNEG = 4'b1000;
14
15     adder dut(clk, rst, A, B, C);
16
17     initial begin
18         clk = 0;
19         forever
20             #1 clk = ~clk;
21     end
22
23     rand_input ri;
24
25     initial begin
26         ri = new(clk);
27         A = 0;
28         B = 0;
29         error_count = 0;
30         correct_count = 0;
31         assert_reset();
32
33         for (int i = 0; i < 200; i++) begin
34             assert(ri.randomize());
35             A = ri.A;
36             B = ri.B;
37             rst = ri.rst;
38             check_result(A + B);
39         end
40
41         $display("errors: %d, success: %d", error_count, correct_count);
42         $stop;
43     end
```

2. Testbench:

```
45  task assert_reset();
46      rst = 1;
47      check_result(0);
48      rst = 0;
49  endtask
50
51  task check_result(logic signed [4:0] C_exp);
52      if(rst) C_exp = 0;
53      @(negedge clk);
54      if(C != C_exp) begin
55          $display ("*** ERROR, A = %d, B = %d, C = %d", A, B, C, );
56          error_count = error_count + 1;
57      end
58      else
59          correct_count = correct_count + 1;
60  endtask
61
62  always @(posedge clk) begin
63      if(!rst) begin
64          ri.cg_A.sample();
65          ri.cg_B.sample();
66      end
67  end
68 endmodule
69
```

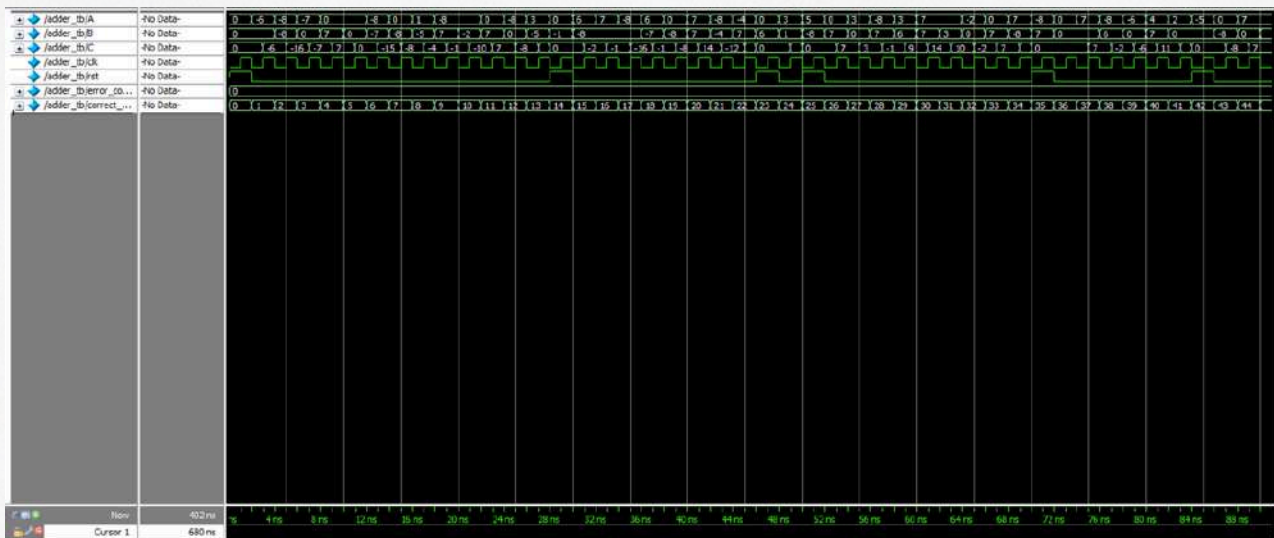

3. Verification Plan:

1	Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
2	RESET_1	When the reset is asserted, output C should be low regardless value of A, B.	Directed at the start and the end of the simulation	-	assert_reset() task checks that output is 0 during reset.
	RANDOMIZED	C always equal to a + b if no reset	using class to random values	two covergroups with 2 coverpoints each, first coverpoint contain bins for ZERO MAXPOS and MAXNEG and a default, second coverpoint contains bins for (0 > MAXPOS) (MAXPOS > MAXNEG) and (MAXNEG > MAXPOS)	check_result task checks for the correct output.
3					

4. Do file:

```
vlib work
vlog 2_adder_pkg.sv 2_adder.v 2_adder_tb.sv +cover -covercells
vsim -voptargs=+acc work.adder_tb -cover
add wave *
coverage save adder_tb.ucdb -onexit
run -all
```

5. Questasim snippet:



```
# errors: 0, success: 201
# ** Note: $stop : 2_adder_tb.sv(42)
# Time: 402 ns Iteration: 1 Instance: /adder_tb
# Break in Module adder_tb at 2_adder_tb.sv line 42
```


6. statement coverage:

```
28  Statement Coverage:
29      Enabled Coverage          Bins    Hits    Misses  Coverage
30      -----
31      Statements                3       3       0    100.00%
32
33      =====Statement Details=====
34
35  Statement Coverage for instance /adder_tb/dut --
36
37      Line      Item              Count    Source
38      ----      -
39  File 2_adder.v
40      1
41      module adder (
42      2
43          input clk,
44      3
45          input reset,
46      4
47          input signed [3:0] A, // Input data A in 2's complement
48      5
49          input signed [3:0] B, // Input data B in 2's complement
50      6
51          output reg signed [4:0] C // Adder output in 2's complement
52      7
53          );
54      8
55
56      9
57          // Register output C
58      10
59          1
60          220      always @(posedge clk or posedge reset) begin
61      11
62          if (reset)
63      12
64          41      C <= 5'b0;
65      13
66          else
67      14
68          179      C <= A + B;
```

7. Branch coverage:

```
7 Branch Coverage:
8   Enabled Coverage          Bins      Hits      Misses  Coverage
9   -----
10  Branches                  2        2        0    100.00%
11
12  =====Branch Details=====
13
14  Branch Coverage for instance /adder_tb/dut
15
16      Line      Item              Count      Source
17      ----      -
18  File 2_adder.v
19  -----IF Branch-----
20      11              220      Count coming in to IF
21      11          1          41          if (reset)
22
23      13          1          179          else
24
25  Branch totals: 2 hits of 2 branches = 100.00%
```


8. Toggle coverage:

```

69 Toggle Coverage:
70 Enabled Coverage      Bins      Hits      Misses    Coverage
71 -----
72 Toggles                30       30         0     100.00%
73
74 =====Toggle Details=====
75
76 Toggle Coverage for instance /adder_tb/dut --
77
78 |   |   |   |   |   |   |   |   |   | Node      1H->0L      0L->1H  "Coverage"
79 |---|---|---|---|---|---|---|-----|
80 |   |   |   |   |   |   |   |   | A[0-3]        1          1     100.00
81 |   |   |   |   |   |   |   |   | B[0-3]        1          1     100.00
82 |   |   |   |   |   |   |   |   | C[4-0]        1          1     100.00
83 |   |   |   |   |   |   |   |   | clk            1          1     100.00
84 |   |   |   |   |   |   |   |   | reset           1          1     100.00
85
86 Total Node Count      =          15
87 Toggled Node Count    =          15
88 Untoggled Node Count  =           0
89
90 Toggle Coverage      =      100.00% (30 of 30 bins)

```


9. Functional coverage :

340	✓	Covergroup Coverage:							
341	✓	Covergroups	2	na	na	100.00%			
342	✓	Coverpoints/Crosses	4	na	na	na			
343		Covergroup Bins	12	12	0	100.00%			
344									
345		Covergroup			Metric	Goal	Bins	Status	
346									
347	✓								
348	✓	TYPE /adder_pkg/rand_input/cg_A			100.00%	100	-	Covered	
349		covered/total bins:			6	6	-		
350		missing/total bins:			0	6	-		
351		% Hit:			100.00%	100	-		
352	✓	Coverpoint a1			100.00%	100	-	Covered	
353		covered/total bins:			3	3	-		
354		missing/total bins:			0	3	-		
355		% Hit:			100.00%	100	-		
356	✓	Coverpoint a2			100.00%	100	-	Covered	
357		covered/total bins:			3	3	-		
358		missing/total bins:			0	3	-		
359		% Hit:			100.00%	100	-		
360	✓	Covergroup instance \adder_pkg::rand_input::cg_A			100.00%	100	-	Covered	
361		covered/total bins:			6	6	-		
362		missing/total bins:			0	6	-		
363		% Hit:			100.00%	100	-		
364		% Hit:			100.00%	100	-		
365	✓	Coverpoint a1			100.00%	100	-	Covered	
366		covered/total bins:			3	3	-		
367		missing/total bins:			0	3	-		
368		% Hit:			100.00%	100	-		
369		bin zero			40	1	-	Covered	
370		bin maxpos			38	1	-	Covered	
371		bin maxneg			46	1	-	Covered	
372		default bin range			56		-	Occurred	
373	✓	Coverpoint a2			100.00%	100	-	Covered	
374		covered/total bins:			3	3	-		
375		missing/total bins:			0	3	-		
376		% Hit:			100.00%	100	-		
377		bin data0_max			11	1	-	Covered	
378		bin data_max_min			14	1	-	Covered	
379		bin data_min_max			10	1	-	Covered	
380	✓	TYPE /adder_pkg/rand_input/cg_B			100.00%	100	-	Covered	
381		covered/total bins:			6	6	-		
382		missing/total bins:			0	6	-		
383		% Hit:			100.00%	100	-		
384	✓	Coverpoint b1			100.00%	100	-	Covered	
385		covered/total bins:			3	3	-		
386		missing/total bins:			0	3	-		
387		% Hit:			100.00%	100	-		
388	✓	Coverpoint b2			100.00%	100	-	Covered	
389		covered/total bins:			3	3	-		
390		missing/total bins:			0	3	-		
391		% Hit:			100.00%	100	-		
392	✓	Covergroup instance \adder_pkg::rand_input::cg_B			100.00%	100	-	Covered	
393		covered/total bins:			6	6	-		
394		missing/total bins:			0	6	-		
395		% Hit:			100.00%	100	-		
396		% Hit:			100.00%	100	-		
397	✓	Coverpoint b1			100.00%	100	-	Covered	
398		covered/total bins:			3	3	-		
399		missing/total bins:			0	3	-		
400		% Hit:			100.00%	100	-		
401		bin zero			50	1	-	Covered	
402		bin maxpos			25	1	-	Covered	
403		bin maxneg			32	1	-	Covered	
404		default bin range			73		-	Occurred	
405	✓	Coverpoint b2			100.00%	100	-	Covered	
406		covered/total bins:			3	3	-		
407		missing/total bins:			0	3	-		
408		% Hit:			100.00%	100	-		
409		bin data0_max			7	1	-	Covered	
410		bin data_max_min			2	1	-	Covered	
411		bin data_min_max			5	1	-	Covered	

Question 3:

1. RTL Design :

```
8  module FSM_010(clk, rst, x, y, users_count);
9      parameter IDLE = 2'b00;
10     parameter ZERO = 2'b01;
11     parameter ONE = 2'b10;
12     parameter STORE = 2'b11;
13
14     input clk, rst, x;
15     output y;
16     output reg [9:0] users_count;
17
18     reg [1:0] cs, ns;
19
20     always @(*) begin
21         case (cs)
22             IDLE:
23                 if (x)
24                     ns = IDLE;
25                 else
26                     ns = ZERO;
27             ZERO:
28                 if (x)
29                     ns = ONE;
30                 else
31                     ns = ZERO;
32             ONE:
33                 if (x)
34                     ns = IDLE;
35                 else
36                     ns = STORE;
37             STORE:
38                 if (x)
39                     ns = IDLE;
40                 else
41                     ns = ZERO;
42             default: ns = IDLE;
43         endcase
44     end
45
```

1. RTL Design :

```
46  always @(posedge clk or posedge rst) begin
47      if(rst) begin
48          cs <= IDLE;
49      end
50      else begin
51          cs <= ns;
52      end
53  end
54
55  always @(posedge clk or posedge rst) begin
56      if(rst) begin
57          users_count <= 0;
58      end
59      else begin
60          if (cs == STORE)
61              users_count <= users_count + 1;
62      end
63  end
64
65  assign y = (cs == STORE)? 1:0;
66
67  endmodule
```


2. Testbench code:

```
1  import FSM_pkg::*;
2
3  module FSM_010_tb();
4      logic clk;
5      logic rst;
6      logic x;
7      logic y;
8      logic [9:0]users_count;
9
10     logic y_exp;
11     logic [9:0]users_count_exp;
12
13     integer correct_count, error_count;
14
15     fsm_transaction my_input;
16
17     FSM_010 dut(.);
18     FSM_010_golden golden(clk, rst, x, y_exp, users_count_exp);
19
20     initial begin
21         clk = 0;
22         forever
23             #1 clk = ~clk;
24     end
25
26
27     initial begin
28         correct_count = 0;
29         error_count = 0;
30         x = 0;
31
32         //reset_test
33         rst = 1;
34         check_result();
35
36         //randomize test
37         my_input = new(clk);
38         repeat(10000) begin
39             assert(my_input.randomize());
40             rst = my_input.rst;
41             x = my_input.x;
42             check_result();
43         end
44
45         $display("*** ERROR count: %0d, CORRECT count: %0d", error_count, correct_count);
46         $stop;
47     end
48
49     task check_result();
50         @(negedge clk);
51         if(y != y_exp || users_count != users_count_exp) begin
52             $display("*** ERROR! at time %0t, y = %0d, Expected = %0d***",
53                 $time, y, y_exp);
54             error_count++;
55         end
56         else
57             correct_count++;
58     endtask
59
60
61 endmodule
62
```

3. Package:

```
2  package FSM_pkg;
3  class fsm_transaction;
4      rand logic x;
5      rand logic rst;
6
7      constraint c1 {
8          rst dist {1 := 1, 0 := 90};
9          x dist {0 := 67, 1:= (100 - 67)};
10     }
11
12     covergroup cg_x (ref logic clk) @(posedge clk);
13         coverpoint x {
14             bins tr = (0 =>1);
15         }
16     endgroup
17
18     function new(ref logic clk);
19         cg_x = new(clk);
20     endfunction
21 endclass
22 endpackage
```


4. Verification Plan:

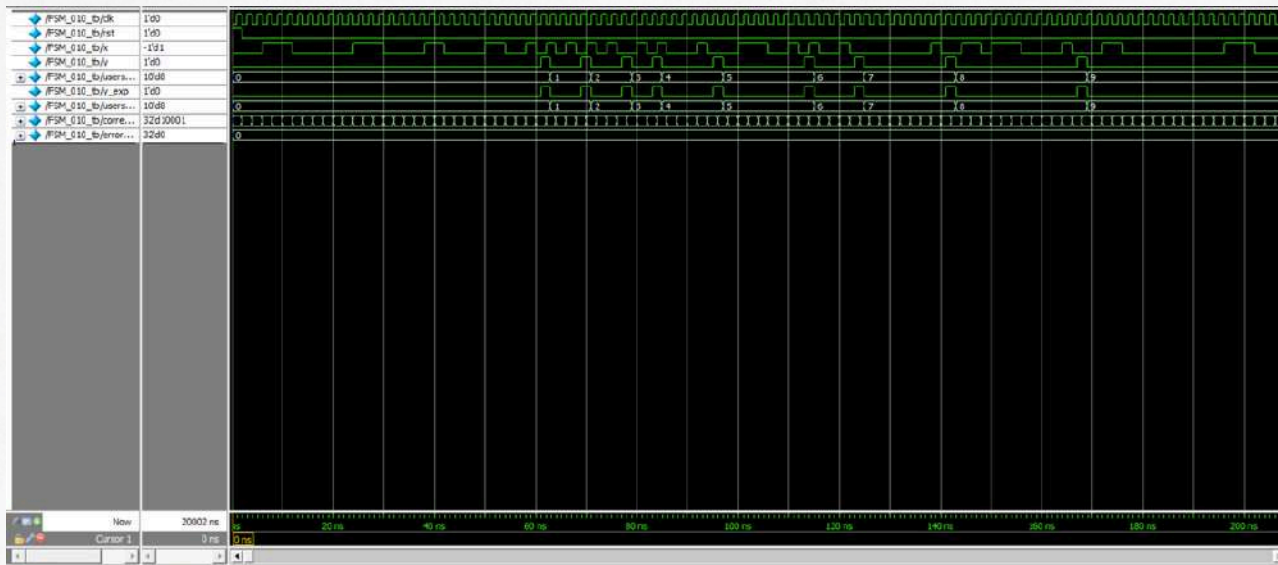
1	Label	Design Requirement Description	Stimulus Generation	Functional Coverage	Functionality Check
2	RESET test	When the reset is asserted, outputs should be low.	Directed at the start and the end of the simulation, and called in randomization step.	-	check_result() task checks that output is 0 during reset.
3	RANDOMIZE test	Output of the dut design is equal to the output of the golden model	Randomized test using rand_stimulus class with constraints 90% deactive reset, 67% x = 1.	one Cover group with one coerpint on x has one bin of transition 0 => 1	check_result() verifies correct output using golden model.

5. Do file:

```
vlib work
vlog 3_golden_model.sv 3_FSM_pkg.sv 3_FSM_010.v 3_FSM_tb.sv +cover -covercells
vsim -voptargs=+acc work.FSM_010_tb -cover
add wave *
coverage save 3_FSM_tb.ucdb -onexit -du work.FSM_010
coverage exclude -du FSM_010 -togglenode {users_count[6]}
coverage exclude -du FSM_010 -togglenode {users_count[7]}
coverage exclude -du FSM_010 -togglenode {users_count[8]}
coverage exclude -du FSM_010 -togglenode {users_count[9]}

run -all
```


6. Questasim wave:



```
# *** ERROR count: 0, CORRECT count: 10001
# ** Note: $stop      : 3_FSM_tb.sv(46)
#   Time: 20002 ns   Iteration: 1   Instance: /FSM_010_tb
# Break in Module FSM_010_tb at 3_FSM_tb.sv line 46
```


7. Statement coverage:

180	Statement Coverage				
181	Enabled Coverage	Bins	Hits	Misses	Coverage
182	-----	----	----	-----	-----
183	Statements	17	17	0	100.00%
184					
185	=====Statement Details=====				
186					
187	Statement Coverage for instance /\FSM_010_tb#dut --				
188					
189	Line	Item	Count	Source	
190	----	----	-----	-----	
191	File 3_FSM_010.v				
192	8			module FSM_010(clk, rst, x, y, users_count);	
193					
194	9			parameter IDLE = 2'b00;	
195					
196	10			parameter ZERO = 2'b01;	
197					
198	11			parameter ONE = 2'b10;	
199					
200	12			parameter STORE = 2'b11;	
201					
202	13				
203					
204	14			input clk, rst, x;	
205					
206	15			output y;	
207					
208	16			output reg [9:0] users_count;	
209					
210	17				
211					
212	18			reg [1:0] cs, ns;	
213					
214	19				
215					
216	20	1	10245	always @(*) begin	
217					
218	21			case (cs)	
219					
220	22			IDLE:	
221					
222	23			if (x)	
223					
224	24	1	1014	ns = IDLE;	
225					
226	25			else	
227					
228	26	1	1073	ns = ZERO;	
229					
230	27			ZERO:	
231					
232	28			if (x)	
233					
234	29	1	1785	ns = ONE;	
235					
236	30			else	
237					
238	31	1	1821	ns = ZERO;	
239					
240	32			ONE:	
241					
242	33			if (x)	
243					
244	34	1	1767	ns = IDLE;	
245					
246	35			else	
247					
248	36	1	1195	ns = STORE;	
249					
250	37			STORE:	
251					
252	38			if (x)	
253					
254	39	1	404	ns = IDLE;	
255					
256	40			else	
257					

7. Statement coverage:

180	Statement Coverage				
181	Enabled Coverage	Bins	Hits	Misses	Coverage
182	-----	----	----	-----	-----
183	Statements	17	17	0	100.00%
184					
185	=====Statement Details=====				
186					
187	Statement Coverage for instance /\FSM_010_tb#dut --				
188					
189	Line	Item	Count	Source	
190	----	----	-----	-----	
191	File 3_FSM_010.v				
192	8			module FSM_010(clk, rst, x, y, users_count);	
193					
194	9			parameter IDLE = 2'b00;	
195					
196	10			parameter ZERO = 2'b01;	
197					
198	11			parameter ONE = 2'b10;	
199					
200	12			parameter STORE = 2'b11;	
201					
202	13				
203					
204	14			input clk, rst, x;	
205					
206	15			output y;	
207					
208	16			output reg [9:0] users_count;	
209					
210	17				
211					
212	18			reg [1:0] cs, ns;	
213					
214	19				
215					
216	20	1	10245	always @(*) begin	
217					
218	21			case (cs)	
219					
220	22			IDLE:	
221					
222	23			if (x)	
223					
224	24	1	1014	ns = IDLE;	
225					
226	25			else	
227					
228	26	1	1073	ns = ZERO;	
229					
230	27			ZERO:	
231					
232	28			if (x)	
233					
234	29	1	1785	ns = ONE;	
235					
236	30			else	
237					
238	31	1	1821	ns = ZERO;	
239					
240	32			ONE:	
241					
242	33			if (x)	
243					
244	34	1	1767	ns = IDLE;	
245					
246	35			else	
247					
248	36	1	1195	ns = STORE;	
249					
250	37			STORE:	
251					
252	38			if (x)	
253					
254	39	1	404	ns = IDLE;	
255					
256	40			else	
257					

7. Statement coverage:

```
256 40 else
257
258 41 1 1185 ns = ZERO;
259
260 42 1 1 default: ns = IDLE;
261
262 43 endcase
263
264 44 end
265
266 45
267
268 46 1 7505 always @(posedge clk or posedge rst) begin
269
270 47 if(rst) begin
271
272 48 1 200 cs <= IDLE;
273
274 49 end
275
276 50 else begin
277
278 51 1 7305 cs <= ns;
279
280 52 end
281
282 53 end
283
284 54
285
286 55 1 5981 always @(posedge clk or posedge rst) begin
287
288 56 if(rst) begin
289
290 57 1 200 users_count <= 0;
291
292 58 end
293
294 59 else begin
295
296 60 if (cs == STORE)
297
298 61 1 1168 users_count <= users_count + 1;
299
300 62 end
301
302 63 end
303
304 64
305
306 65 1 5828 assign y = (cs == STORE)? 1:0;
307
```

8. Branch coverage:

```
7 Branch Coverage:
8   Enabled Coverage      Bins    Hits    Misses  Coverage
9   -----
10  Branches              21      21      0    100.00%
11
12 =====Branch Details=====
13
14 Branch Coverage for instance /\FSM_010_tb#dut
15
16   Line      Item      Count    Source
17   ----      -
18   File 3_FSM_010.v
19   -----CASE Branch-----
20   21              10245    Count coming in to CASE
21   22              2087    IDLE:
22
23   27              3606    ZERO:
24
25   32              2962    ONE:
26
27   37              1589    STORE:
28
29   42              1        default: ns = IDLE;
30
31 Branch totals: 5 hits of 5 branches = 100.00%
32
33 -----IF Branch-----
34   23              2087    Count coming in to IF
35   23              1014    if (x)
36
37   25              1073    else
38
39 Branch totals: 2 hits of 2 branches = 100.00%
40
41 -----IF Branch-----
42   28              3606    Count coming in to IF
43   28              1785    if (x)
44
45   30              1821    else
46
47 Branch totals: 2 hits of 2 branches = 100.00%
48
49 -----IF Branch-----
50   33              2962    Count coming in to IF
51   33              1767    if (x)
52
53   35              1195    else
54
55 Branch totals: 2 hits of 2 branches = 100.00%
56
57 -----IF Branch-----
58   38              1589    Count coming in to IF
59   38              404    if (x)
60
61   40              1185    else
62
63 Branch totals: 2 hits of 2 branches = 100.00%
64
65 -----IF Branch-----
66   47              7505    Count coming in to IF
67   47              200    if(rst) begin
68
69   50              7305    else begin
70
71 Branch totals: 2 hits of 2 branches = 100.00%
72
73 -----IF Branch-----
74   56              5981    Count coming in to IF
75   56              200    if(rst) begin
76
77   59              5781    else begin
78
79 Branch totals: 2 hits of 2 branches = 100.00%
80
```


8. Branch coverage:

81	✓	-----IF Branch-----			
82		60		5781	Count coming in to IF
83	✓	60	1	1168	if (cs == STORE)
84					
85				4613	All False Count
86		Branch totals: 2 hits of 2 branches = 100.00%			
87					
88	✓	-----IF Branch-----			
89		65		5827	Count coming in to IF
90		65	1	1185	assign y = (cs == STORE)? 1:0;
91					
92		65	2	4642	assign y = (cs == STORE)? 1:0;
93					
94		Branch totals: 2 hits of 2 branches = 100.00%			
95					

9. FSM coverage:

```
139
140 =====FSM Details=====
141
142 FSM Coverage for instance /\FSM_010_tb#dut --
143
144 FSM_ID: cs
145   Current State Object : cs
146   -----
147   State Value MapInfo :
148   -----
149   Line          State Name          Value
150   ----          -
151   22            IDLE                0
152   27            ZERO                1
153   32            ONE                 2
154   37            STORE               3
155   Covered States :
156   -----
157   State          Hit_count
158   ----          -
159   IDLE           1517
160   ZERO           3036
161   ONE            1767
162   STORE          1185
163   Covered Transitions :
164   -----
165   Line          Trans_ID          Hit_count          Transition
166   ----          -
167   26            0                 1053              IDLE -> ZERO
168   29            1                 1767              ZERO -> ONE
169   48            2                  54               ZERO -> IDLE
170   36            3                 1185              ONE -> STORE
171   34            4                  582              ONE -> IDLE
172   41            5                 768               STORE -> ZERO
173   39            6                 417               STORE -> IDLE
174
175
176   Summary          Bins          Hits          Misses          Coverage
177   -----
178   FSM States          4              4              0          100.00%
179   FSM Transitions     7              7              0          100.00%
```


10. Toggle coverage:

```
309 Toggle Coverage:
310   Enabled Coverage      Bins    Hits    Misses  Coverage
311   -----
312   Toggles                28      28        0  100.00%
313
314   =====Toggle Details=====
315
316 Toggle Coverage for instance /\FSM_010_tb#dut --
317
318   Node      1H->0L    0L->1H    "Coverage"
319   -----
320   clk                1        1    100.00
321   cs[1-0]            1        1    100.00
322   ns[1-0]            1        1    100.00
323   rst                1        1    100.00
324   users_count[5-0]  1        1    100.00
325   x                  1        1    100.00
326   y                  1        1    100.00
327
328 Total Node Count      =      14
329 Toggled Node Count    =      14
330 Untoggled Node Count  =        0
331
332 Toggle Coverage       =    100.00% (28 of 28 bins)
333
334
335 Total Coverage By Instance (filtered view): 100.00%
336
```

10. Functional coverage:

920	COVERGROUP COVERAGE:				
921	-----				
922	Covergroup	Metric	Goal	Bins	Status
923	-----				
925	TYPE /FSM_pkg/fsm_transaction/cg_x	100.00%	100	-	Covered
926	covered/total bins:	1	1	-	
927	missing/total bins:	0	1	-	
928	% Hit:	100.00%	100	-	
929	Coverpoint x	100.00%	100	-	Covered
930	covered/total bins:	1	1	-	
931	missing/total bins:	0	1	-	
932	% Hit:	100.00%	100	-	
933	Covergroup instance \FSM_pkg::fsm_transaction::cg_x				
934		100.00%	100	-	Covered
935	covered/total bins:	1	1	-	
936	missing/total bins:	0	1	-	
937	% Hit:	100.00%	100	-	
938	Coverpoint x	100.00%	100	-	Covered
939	covered/total bins:	1	1	-	
940	missing/total bins:	0	1	-	
941	% Hit:	100.00%	100	-	
942	bin tr	2209	1	-	Covered
943					
944	TOTAL COVERGROUP COVERAGE: 100.00% COVERGROUP TYPES: 1				

Question 4:

1. Design:

```
module mem();  
    logic [23:0] mem [bit[19:0]];  
  
    initial begin  
        mem[20'h00000] = 24'hA50400; // reset  
        mem[20'h00400] = 24'h123456; // instruction1  
        mem[20'h00401] = 24'h789ABC; // instruction2  
        mem[20'hFFFFFF] = 24'h0F1E2D; //isr  
  
        $display("Number of elements in memory = %0d", mem.num());  
        foreach(mem[i]) begin  
            $display("mem[%0h] = %0h", i, mem[i]);  
        end  
    end  
endmodule
```

2. Result:

```
# Number of elements in memory = 4  
# mem[0] = a50400  
# mem[400] = 123456  
# mem[401] = 789abc  
# mem[ffffff] = fle2d
```

Question 5:

1.Design:

```
1  module struct_e();
2
3      typedef bit [6:0] ubyte_t;
4
5      typedef struct {
6          ubyte_t header;
7          ubyte_t cmd;
8          ubyte_t data;
9          ubyte_t crc;
10     } packet;
11
12     packet my_packet;
13
14     initial begin
15         my_packet.header = 7'h5A;
16         $display("my_packet.header = %h", my_packet.header);
17         $display("my_packet.header = %p", my_packet);
18     end
19 endmodule
```

2.Result:

```
VSIM 23> run -all
# my_packet.header = 5a
# my_packet.header = '{header:90, cmd:0, data:0, crc:0}
```