

Assignment4_extra

By: Mohamed Sayed Mohamed Soliman

Question 1

1. ALSU Testbench

```
1 module config_reg_tb();
2 parameter TEST = 100 ;
3 parameter CLK_period = 40 ;
4 logic [8:0] data_read_expect_assoc [string] ;
5 logic clk;
6 logic reset;
7 logic write;
8 logic [15:0] data_in;
9 logic [2:0] address;
10 logic [15:0] data_out;
11 logic [15:0] reset_assoc[string] = {
12     "adc0_reg"      : 16'hffff,
13     "adc1_reg"      : 16'h0,
14     "temp_sensor0_reg": 16'h0,
15     "temp_sensor1_reg": 16'h0,
16     "analog_test"   : 16'habcd,
17     "digital_test"  : 16'h0,
18     "amp_gain"       : 16'h0,
19     "digital_config" : 16'h1
20 };
21 logic [15:0] x ; |
22 logic [15:0] reset_assoc_arr [0:7] = {16'hffff, 16'h0, 16'h0, 16'h0, 16'habcd, 16'h0, 16'h0, 16'h1};
23 logic [15:0] reset_assoc_chan [0:7] = {16'hffff, 16'h0, 16'h0, 16'h0, 16'habcd, 16'h0, 16'h0, 16'h1};
24 integer i , j;
25 integer correct_counter = 0 ;
26 integer error_counter = 0 ;
27 typedef enum logic [2:0] {
```

```
28     adc0_reg = 3'd0,
29     adc1_reg = 3'd1,
30     temp_sensor0_reg = 3'd2,
31     temp_sensor1_reg = 3'd3,
32     analog_test = 3'd4,
33     digital_test = 3'd5,
34     amp_gain = 3'd6,
35     digital_config = 3'd7
36 } reg_addr_t;
37 reg_addr_t reg_addr;
38 config_reg dut (.*) ;
39     initial begin
40         clk = 0 ;
41     forever
42     begin
43         #20 clk = ~clk ;
44     end
45 end
46 task rst;
47     begin
48         reset = 1;
49         #40;
50         reset = 0;
51         #40;
52     end
53 endtask
```

```
54
55 initial begin
56     write = 0 ;
57     reset = 0 ;
58     data_out = 0 ;
59     data_in = 0 ;
60     address = 0 ;
61     rst();
62     for(i=0 ; i<8 ; i = i + 1)
63     begin
64         address = i ;
65         check_result(address );
66     end
67 @(negedge clk) ;
68 write = 1 ;
69 @(negedge clk) ;
70 for(i=0 ; i<8 ; i = i + 1)
71 begin
72     address = i ;
73     data_in = 'hFFFF ;
74     golden_model(address , data_in);
75     @(negedge clk) ;
76 end
77
78 @(negedge clk) ;
79 write = 0 ;
80 $display ("##### " );
81 for(i=0 ; i<8 ; i = i + 1)
82 begin
83     address = i ;
84     check_result_chan(address );
85 end
```

```

86
87 @(negedge clk) ;
88 reset = 1 ;
89 @(negedge clk) ;
90 $display ("##### " );
91 for(i=0 ; i<8 ; i = i + 1)
92 begin
93     address = i ;
94     check_result(address ) ;
95 end
96 @(negedge clk) ;
97 reset = 0 ;
98 write = 1 ;
99 @(negedge clk) ;
100
101 for(i=0 ; i<8 ; i = i + 1)
102 begin
103     address = i ;
104     data_in = 0 ;
105     golden_model(address , data_in);
106     @(negedge clk) ;
107 end
108
109 @(negedge clk) ;
110 write = 0 ;
111 $display ("##### " );
112 for(i=0 ; i<8 ; i = i + 1)
113 begin
114     address = i ;
115     check_result_chan(address ) ;
116 end
117

```

```

117
118 @(negedge clk) ;
119 for(j=0 ; j<100 ; j = j + 1)
120 begin
121     @(negedge clk) ;
122     write = 1 ;
123     @(negedge clk) ;
124     for(i=0 ; i<8 ; i = i + 1)
125     begin
126         address = i ;
127         data_in = $random ;
128         golden_model(address , data_in);
129         @(negedge clk) ;
130     end
131     @(negedge clk) ;
132     write = 0 ;
133     $display ("##### " );
134     for(i=0 ; i<8 ; i = i + 1)
135     begin
136         address = i ;
137         check_result_chan(address ) ;
138     end
139     @(negedge clk) ;
140 end
141 $display (" testbench 1 " );
142 $display ("error_counter = %0d " ,error_counter );
143 $display ("correct_counter = %0d " ,correct_counter );
144 $stop ;
145 end

```

```

146
147 task golden_model(input [2:0] address ,input [15:0] data_in );
148     reset_assoc_chan[address] = data_in ;
149 endtask
150
151 task check_result(input [2:0] address );
152     @(negedge clk) ;
153     if(reset_assoc_arr[address] != data_out )
154     begin
155         $display ("error::data_out = 0h%0h , reset_assoc_arr[%0d] = 0h%0h " , data_out , address , reset_assoc_arr[address]);
156         error_counter = error_counter +1 ;
157     end
158     else
159     begin
160         correct_counter = correct_counter + 1 ;
161     end
162 endtask
163
164 task check_result_chan(input [2:0] address );
165     @(negedge clk) ;
166     if(reset_assoc_chan[address] != data_out )
167     begin
168         $display ("error::data_out = 0h%0h , reset_assoc_chan[%0d] = 0h%0h " , data_out , address , reset_assoc_chan[address]);
169         error_counter = error_counter +1 ;
170     end
171     else
172     begin
173         correct_counter = correct_counter + 1 ;
174     end
175 endtask
176
177 endmodule

```

2. Transcript & bugs

```
##### reset deassert #####
# :error::data_out = 0habcc , reset_assoc_arr[4] = 0habcd
##### stored 'hfff' #####
# :error::data_out = 0hfffe , reset_assoc_chan[2] = 0hffff
# :error::data_out = 0h7fff , reset_assoc_chan[7] = 0hffff
##### reset deassert #####
# :error::data_out = 0hffff , reset_assoc_arr[3] = 0h0
# :error::data_out = 0habcc , reset_assoc_arr[4] = 0habcd
##### stored 'h0000' #####
# :error::data_out = 0h8000 , reset_assoc_chan[0] = 0h0
##### random data input #####
#####
# :error::data_out = 0hb524 , reset_assoc_chan[0] = 0h3524
# :error::data_out = 0h815e , reset_assoc_chan[1] = 0h5e81
# :error::data_out = 0hac12 , reset_assoc_chan[2] = 0hd609
# :error::data_out = 0h8465 , reset_assoc_chan[5] = 0h998d
# :error::data_out = 0h998d , reset_assoc_chan[6] = 0h8465
#####
# :error::data_out = 0hdcd , reset_assoc_chan[1] = 0hcd0d
# :error::data_out = 0he2ec , reset_assoc_chan[2] = 0hf176
# :error::data_out = 0he9f9 , reset_assoc_chan[5] = 0hf78c
# :error::data_out = 0hf78c , reset_assoc_chan[6] = 0he9f9
#####
# :error::data_out = 0haad2 , reset_assoc_chan[1] = 0hd2aa
# :error::data_out = 0hefca , reset_assoc_chan[2] = 0hf7e5
# :error::data_out = 0h69f2 , reset_assoc_chan[5] = 0hdb8f
# :error::data_out = 0hdb8f , reset_assoc_chan[6] = 0h69f2
# :error::data_out = 0hl6ce , reset_assoc_chan[7] = 0h96ce
#####
```

Question 2

```
C: > Users > CS > Downloads > Kareem Wasem Diploma > session_ass4_extra > sva.sv
1  property p_request_to_grant;
2      @(posedge clk)
3      disable iff (reset)
4      $rose(request) |-> ##[2:5] $rose(grant);
5  endproperty
6
7  dollar1_ass1: assert property (p_request_to_grant) ;
8  dollar1_cover1: cover property (p_request_to_grant) ;
9
10 property p_grant_acknowledgement;
11     @(posedge clk)
12     disable iff (reset)
13     $rose(grant) |=> (frame == 0 && irdy == 0);
14 endproperty
15
16 dollar2_ass2: assert property (p_grant_acknowledgement) ;
17 dollar2_cover2: cover property (p_grant_acknowledgement) ;
18
19 property p_transaction_complete;
20     @(posedge clk)
21     disable iff (reset)
22     (frame && irdy) |=> !grant;
23 endproperty
24
25 dollar3_ass3: assert property (p_transaction_complete) ;
26 dollar3_cover3: cover property (p_transaction_complete) ;
27
28
```

Question 3

```
31
32 property p_one_hot_state;
33     @(posedge clk)
34     disable iff (reset)
35     $onehot(cs);
36 endproperty
37
38 dollar1_ass1: assert property (p_one_hot_state) ;
39 dollar1_cover1: cover property (p_one_hot_state) ;
40
41 property p_idle_to_gen_blk_addr;
42     @(posedge clk)
43     disable iff (reset)
44     (cs == IDLE && $rose(get_data)) | => (cs == GEN_BLK_ADDR) ##64 (cs == WAIT0);
45 endproperty
46
47 dollar2_ass2: assert property (p_idle_to_gen_blk_addr) ;
48 dollar2_cover2: cover property (p_idle_to_gen_blk_addr) ;
49
```