

Assignment 6

Verification For ALSU Using UVM

alsu.do

```
1 vlib work
2 vlog -f alsu_files.list
3 vsim -voptargs=+acc work.alsu_top -classdebug -uvmcontrol=all
4 add wave /alsu_top/alsuif/*
5 run -all
```

```
1 module ALSU( alsu_if.DUT alsuif);
2
3     parameter INPUT_PRIORITY = "A";
4     parameter FULL_ADDER = "ON";
5
6     reg red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
7     reg [1:0] cin_reg; // should be a single bit
8     reg [2:0] opcode_reg;
9     reg signed [2:0] A_reg, B_reg;
10
11     wire invalid_red_op, invalid_opcode, invalid;
12
13     //Invalid handling
14     assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);
15     assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
16     assign invalid = invalid_red_op | invalid_opcode;
17
18     //Registering input signals
19     always @(posedge alsuif.clk or posedge alsuif.rst) begin
20         if(alsuif.rst) begin
21             cin_reg <= 0;
22             red_op_B_reg <= 0;
23             red_op_A_reg <= 0;
24             bypass_B_reg <= 0;
25             bypass_A_reg <= 0;
26             direction_reg <= 0;
27             serial_in_reg <= 0;
28             opcode_reg <= 0;
29             A_reg <= 0;
30             B_reg <= 0;
31         end else begin
32             cin_reg <= alsuif.cin;
33             red_op_B_reg <= alsuif.red_op_B;
34             red_op_A_reg <= alsuif.red_op_A;
35             bypass_B_reg <= alsuif.bypass_B;
36             bypass_A_reg <= alsuif.bypass_A;
37             direction_reg <= alsuif.direction;
38             serial_in_reg <= alsuif.serial_in;
39             opcode_reg <= alsuif.opcode;
40             A_reg <= alsuif.A;
41             B_reg <= alsuif.B;
42         end
43     end
44 end
```



```

91     alsuif.out <= A_reg ^ B_reg;
92     end
93     3'h2: begin // FULL_ADDER parameter must be taken into consediration
94         if (FULL_ADDER == "ON")
95             alsuif.out <= A_reg + B_reg + cin_reg ;
96         else
97             alsuif.out <= A_reg + B_reg ;
98         end
99         3'h3: alsuif.out <= A_reg * B_reg;
100         3'h4: begin
101             if (direction_reg)
102                 alsuif.out <= {alsuif.out[4:0], serial_in_reg};
103             else
104                 alsuif.out <= {serial_in_reg, alsuif.out[5:1]};
105             end
106             3'h5: begin
107                 if (direction_reg)
108                     alsuif.out <= {alsuif.out[4:0], alsuif.out[5]};
109                 else
110                     alsuif.out <= {alsuif.out[0], alsuif.out[5:1]};
111                 end
112                 // pragma coverage off
113                 default : alsuif.out <= 0 ;
114                 // pragma coverage on
115             endcase
116         end
117     end
118 end
119
120 endmodule

```

```

1
2 module ALSU_golden_model ( alsu_if.GOLDEN_MODEL alsuif);
3     parameter INPUT_PRIORITY = "A";
4     parameter FULL_ADDER = "ON";
5
6     reg red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
7     reg [1:0] cin_reg;
8     reg [2:0] opcode_reg;
9     reg signed [2:0] A_reg, B_reg;
10
11     wire invalid_red_op, invalid_opcode, invalid;
12
13     assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);
14     assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
15     assign invalid = invalid_red_op | invalid_opcode;
16
17     always @(posedge alsuif.clk or posedge alsuif.rst) begin
18         if(alsuif.rst) begin
19             cin_reg <= 0;
20             red_op_B_reg <= 0;
21             red_op_A_reg <= 0;
22             bypass_B_reg <= 0;
23             bypass_A_reg <= 0;
24             direction_reg <= 0;
25             serial_in_reg <= 0;
26             opcode_reg <= 0;
27             A_reg <= 0;
28             B_reg <= 0;
29         end else begin
30             cin_reg <= alsuif.cin;
31             red_op_B_reg <= alsuif.red_op_B;
32             red_op_A_reg <= alsuif.red_op_A;
33             bypass_B_reg <= alsuif.bypass_B;
34             bypass_A_reg <= alsuif.bypass_A;
35             direction_reg <= alsuif.direction;
36             serial_in_reg <= alsuif.serial_in;
37             opcode_reg <= alsuif.opcode;
38             A_reg <= alsuif.A;
39             B_reg <= alsuif.B;
40         end
41     end
42
43     always @(posedge alsuif.clk or posedge alsuif.rst) begin
44         if(alsuif.rst) begin
45             alsuif.leds_ex <= 0;
46         end else begin
47             if (invalid)
48                 alsuif.leds_ex <= ~alsuif.leds_ex;

```

```

49     else
50         alsuif.leds_ex <= 0;
51     end
52 end
53
54 always @(posedge alsuif.clk , posedge alsuif.rst) begin
55     if (alsuif.rst) begin
56         alsuif.out_ex <= 0;
57     end
58     else begin
59         if (bypass_A_reg && bypass_B_reg)
60             alsuif.out_ex <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
61         else if (bypass_A_reg)
62             alsuif.out_ex <= A_reg;
63         else if (bypass_B_reg)
64             alsuif.out_ex <= B_reg;
65         else if (invalid)
66             alsuif.out_ex <= 0;
67         else begin
68             case (opcode_reg)
69                 3'b000 : begin
70                     if (red_op_A_reg && red_op_B_reg)
71                         alsuif.out_ex <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;
72                     else if (red_op_A_reg)
73                         alsuif.out_ex <= |A_reg;
74                     else if (red_op_B_reg)
75                         alsuif.out_ex <= |B_reg;
76                     else
77                         alsuif.out_ex <= A_reg | B_reg;
78                     end
79                 3'b001 : begin
80                     if (red_op_A_reg && red_op_B_reg)
81                         alsuif.out_ex <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;
82                     else if (red_op_A_reg)
83                         alsuif.out_ex <= ^A_reg;
84                     else if (red_op_B_reg)
85                         alsuif.out_ex <= ^B_reg;
86                     else
87                         alsuif.out_ex <= A_reg ^ B_reg;
88                     end
89                 3'b010 : begin
90                     if (FULL_ADDER == "ON")
91                         alsuif.out_ex <= A_reg + B_reg + cin_reg;
92                     else
93

```

```

89         end
90         3'b010 : begin
91             if (FULL_ADDER == "ON")
92                 alsuif.out_ex <= A_reg + B_reg + cin_reg;
93             else
94                 alsuif.out_ex <= A_reg + B_reg;
95             end
96         3'b011 : alsuif.out_ex <= A_reg * B_reg;
97         3'b100 : begin
98             if (direction_reg)
99                 alsuif.out_ex <= {alsuif.out_ex[4:0],serial_in_reg};
100            else
101                alsuif.out_ex <= {serial_in_reg,alsuif.out_ex[5:1]};
102            end
103        3'b101 : begin
104            if (direction_reg)
105                alsuif.out_ex <= {alsuif.out_ex[4:0],alsuif.out_ex[5]};
106            else
107                alsuif.out_ex <= {alsuif.out_ex[0],alsuif.out_ex[5:1]};
108            end
109        endcase
110    end
111 end
112 end
113
114 endmodule
115

```

```

1  alsu_sva.sv
2  module ALSU_sva ( alsu_if,DUT alsuif );
3
4      parameter INPUT_PRIORITY = "A";
5      parameter FULL_ADDER = "ON";
6
7      property reset_check;
8          @(posedge alsuif.clk) (alsuif.rst) | => (alsuif.out == 0 && alsuif.leds == 0);
9      endproperty
10     assert property (reset_check) else $error(" error in rst ");
11     cover property (reset_check);
12
13     property leds_check;
14         @(posedge alsuif.clk) disable iff (alsuif.rst)
15         (ALSU.invalid) | => (alsuif.leds == $past(alsuif.leds));
16     endproperty
17     assert property (leds_check) else $error(" error in leds ");
18     cover property (leds_check);
19
20     property check1;
21         @(posedge alsuif.clk) disable iff (alsuif.rst)
22         ( !ALSU.invalid && ALSU.bypass_A_reg && ALSU.bypass_B_reg ) | =>
23         (alsuif.out == (( INPUT_PRIORITY == "A" ) ? $past(ALSU.A_reg) : $past(ALSU.B_reg)));
24     endproperty
25     assert property (check1) else $error("error in check1 (A priority)");
26     cover property (check1);
27
28     property check2;
29         @(posedge alsuif.clk) disable iff (alsuif.rst)
30         ( !ALSU.invalid && ALSU.bypass_A_reg && !ALSU.bypass_B_reg ) | =>
31         (alsuif.out == $past(ALSU.A_reg));
32     endproperty
33     assert property (check2) else $error("error in check2");
34     cover property (check2);
35
36     property check3;
37         @(posedge alsuif.clk) disable iff (alsuif.rst)
38         ( !ALSU.invalid && !ALSU.bypass_A_reg && ALSU.bypass_B_reg ) | =>
39         (alsuif.out == $past(ALSU.B_reg));
40     endproperty
41     assert property (check3) else $error("error in check3");
42     cover property (check3);
43

```

```

44 property check4;
45     @(posedge alsuif.clk) disable iff (alsuif.rst)
46     ( !ALSU.bypass_A_reg && !ALSU.bypass_B_reg && !ALSU.invalid && (ALSU.opcode_reg == 3'b000) && ALSU.red_op_A_reg && ALSU.red_op_B_reg ) |>
47     (alsuif.out == (( INPUT_PRIORITY == "A" ) ? $past(ALSU.A_reg) : $past(ALSU.B_reg)));
48 endproperty
49 assert property (check4) else $error("error in check4");
50 cover property (check4);
51
52 property check5;
53     @(posedge alsuif.clk) disable iff (alsuif.rst)
54     ( !ALSU.bypass_A_reg && !ALSU.bypass_B_reg && !ALSU.invalid && (ALSU.opcode_reg == 3'b000) && ALSU.red_op_A_reg && !ALSU.red_op_B_reg ) |>
55     (alsuif.out == $past(ALSU.A_reg));
56 endproperty
57 assert property (check5) else $error("error in check5");
58 cover property (check5);
59
60 property check6;
61     @(posedge alsuif.clk) disable iff (alsuif.rst)
62     ( !ALSU.bypass_A_reg && !ALSU.bypass_B_reg && !ALSU.invalid && (ALSU.opcode_reg == 3'b000) && !ALSU.red_op_A_reg && ALSU.red_op_B_reg ) |>
63     (alsuif.out == $past(ALSU.B_reg));
64 endproperty
65 assert property (check6) else $error("error in check6");
66 cover property (check6);
67
68 property check7;
69     @(posedge alsuif.clk) disable iff (alsuif.rst)
70     ( !ALSU.bypass_A_reg && !ALSU.bypass_B_reg && !ALSU.invalid && (ALSU.opcode_reg == 3'b000) && !ALSU.red_op_A_reg && !ALSU.red_op_B_reg ) |>
71     (alsuif.out == $past(ALSU.A_reg) | $past(ALSU.B_reg));
72 endproperty
73 assert property (check7) else $error("error in check7");
74 cover property (check7);
75
76 property check8;
77     @(posedge alsuif.clk) disable iff (alsuif.rst)
78     ( !ALSU.bypass_A_reg && !ALSU.bypass_B_reg && !ALSU.invalid && (ALSU.opcode_reg == 3'b001) && ALSU.red_op_A_reg && ALSU.red_op_B_reg ) |>
79     (alsuif.out == (( INPUT_PRIORITY == "A" ) ? ^$past(ALSU.A_reg) : ^$past(ALSU.B_reg)));
80 endproperty
81 assert property (check8) else $error("error in check8");
82 cover property (check8);
83
84 property check9;
85     @(posedge alsuif.clk) disable iff (alsuif.rst)
86     ( !ALSU.bypass_A_reg && !ALSU.bypass_B_reg && !ALSU.invalid && (ALSU.opcode_reg == 3'b001) && ALSU.red_op_A_reg && !ALSU.red_op_B_reg ) |>
87     (alsuif.out == ^$past(ALSU.A_reg));
88 endproperty
89 assert property (check9) else $error("error in check9");
90 cover property (check9);

```

```

92 property check10;
93     @(posedge alsuif.clk) disable iff (alsuif.rst)
94     ( !ALSU.bypass_A_reg && !ALSU.bypass_B_reg && !ALSU.invalid && (ALSU.opcode_reg == 3'b001) && !ALSU.red_op_A_reg && ALSU.red_op_B_reg ) |>
95     (alsuif.out == ^$past(ALSU.B_reg));
96 endproperty
97 assert property (check10) else $error("error in check10");
98 cover property (check10);
99
100 property check11;
101     @(posedge alsuif.clk) disable iff (alsuif.rst)
102     ( !ALSU.bypass_A_reg && !ALSU.bypass_B_reg && !ALSU.invalid && (ALSU.opcode_reg == 3'b001) && !ALSU.red_op_A_reg && !ALSU.red_op_B_reg ) |>
103     (alsuif.out == $past(ALSU.A_reg) ^ $past(ALSU.B_reg));
104 endproperty
105 assert property (check11) else $error("error in check11");
106 cover property (check11);
107
108 property check12;
109     @(posedge alsuif.clk) disable iff (alsuif.rst)
110     ( !ALSU.bypass_A_reg && !ALSU.bypass_B_reg && !ALSU.invalid && (ALSU.opcode_reg == 3'b010) ) |>
111     (alsuif.out == (( FULL_ADDER == "ON" ) ? $past(ALSU.A_reg)+$past(ALSU.B_reg)+$past(ALSU.cin_reg) : $past(ALSU.A_reg)+$past(ALSU.B_reg)));
112 endproperty
113 assert property (check12) else $error("error in check9");
114 cover property (check12);
115
116 property check13;
117     @(posedge alsuif.clk) disable iff (alsuif.rst)
118     ( !ALSU.bypass_A_reg && !ALSU.bypass_B_reg && !ALSU.invalid && (ALSU.opcode_reg == 3'b011) ) |>
119     (alsuif.out == $past(ALSU.A_reg) * $past(ALSU.B_reg));
120 endproperty
121 assert property (check13) else $error("error in check13");
122 cover property (check13);
123
124 property check14;
125     @(posedge alsuif.clk) disable iff (alsuif.rst)
126     ( !ALSU.bypass_A_reg && !ALSU.bypass_B_reg && !ALSU.invalid && (ALSU.opcode_reg == 3'b100) && ALSU.direction_reg ) |>
127     (alsuif.out == {$past(alsuif.out[4:0]), $past(ALSU.serial_in_reg)});
128 endproperty
129 assert property (check14) else $error("error in check14");
130 cover property (check14);
131
132 property check15;
133     @(posedge alsuif.clk) disable iff (alsuif.rst)
134     ( !ALSU.bypass_A_reg && !ALSU.bypass_B_reg && !ALSU.invalid && (ALSU.opcode_reg == 3'b100) && !ALSU.direction_reg ) |>
135     (alsuif.out == {$past(ALSU.serial_in_reg) , $past(alsuif.out[5:1])});
136 endproperty
137 assert property (check15) else $error("error in check15");
138 cover property (check15);

```

```

139
140 property check16;
141     @(posedge alsuif.clk) disable iff (alsuif.rst)
142     ( !ALSU.bypass_A_reg && !ALSU.bypass_B_reg && !ALSU.invalid && (ALSU.opcode_reg == 3'b101) && ALSU.direction_reg) |->
143     (alsuif.out == {$past(alsuif.out[4:0]), $past(alsuif.out[5])});
144 endproperty
145 assert property (check16) else $error("error in check16");
146 cover property (check16);
147
148 property check17;
149     @(posedge alsuif.clk) disable iff (alsuif.rst)
150     ( !ALSU.bypass_A_reg && !ALSU.bypass_B_reg && !ALSU.invalid && (ALSU.opcode_reg == 3'b101) && !ALSU.direction_reg) |->
151     (alsuif.out == {$past(alsuif.out[0]), $past(alsuif.out[5:1])});
152 endproperty
153 assert property (check17) else $error("error in check17");
154 cover property (check17);
155 endmodule
156
157

```

```

alsu_if.sv
1 interface alsu_if (clk);
2     parameter INPUT_PRIORITY = "A";
3     parameter FULL_ADDER = "ON";
4     input bit clk;
5     logic [1:0] cin ;
6     logic rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
7     logic [2:0] opcode;
8     logic signed [2:0] A, B;
9     logic [15:0] leds;
10    logic [15:0] leds_ex;
11    logic signed [5:0] out;
12    logic signed [5:0] out_ex;
13
14    modport DUT (
15        input clk, cin, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in,opcode,A,B,
16        output out,leds
17    );
18
19    modport GOLDEN_MODEL (
20        input clk, cin, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in,opcode,A,B,
21        output out_ex,leds_ex
22    );
23
24 endinterface

```



```

1 package alsu_seq_item_pkg;
2 import uvm_pkg::*;
3 `include "uvm_macros.svh"
4
5 typedef enum bit [2:0] { OR,XOR,ADD,MULT,SHIFT,ROTATE,invalid6,invalid7 } opcode_e;
6
7 localparam MAXPOS = 3, ZERO = 0, MAXNEG = -4;
8
9 class alsu_seq_item extends uvm_sequence_item;
10   `uvm_object_utils(alsu_seq_item);
11
12   rand logic rst;
13   rand logic signed [2:0] A,B;
14   rand logic red_op_A,red_op_B;
15   rand opcode_e opcode;
16   rand logic bypass_A, bypass_B;
17   rand logic direction,cin;
18   rand logic serial_in;
19   bit [2:0]walking_ones[] = '{3'b100,3'b010,3'b001};
20   rand logic [3:0] A_rem_values , B_rem_values ;
21   rand logic [3:0] A_max_values , B_max_values ;
22   rand bit[2:0]walking_ones_t , walking_ones_f;
23   rand opcode_e arr[6];
24   bit clk;
25   logic [15:0] leds;
26   logic [15:0] leds_ex;
27   logic [5:0] out;
28   logic [5:0] out_ex;
29
30   function new (string name = "alsu_seq_item");
31     super.new (name);
32   endfunction
33
34   function string convert2string();
35     return $sformatf("%s rst=%b a=%b b=%b opcode=%s",super.convert2string(),rst,A,B,opcode);
36   endfunction
37
38   function string convert2string_stimulus();
39     return $sformatf(" rst=%b a=%b b=%b opcode=%s",rst,A,B,opcode);
40   endfunction
41
42   constraint rst_c{
43     rst dist{1:= 2 ,0:= 98};
44   }
45

```

```

45
46 //label2
47 constraint opcode_c {
48     A_rem_values != MAXPOS || MAXNEG || ZERO ;
49     B_rem_values != MAXPOS || MAXNEG || ZERO ;
50     A_max_values == MAXPOS || MAXNEG || ZERO ;
51     B_max_values == MAXPOS || MAXNEG || ZERO ;
52     walking_ones_t inside {walking_ones};
53     !(walking_ones_f inside {walking_ones});
54
55     (opcode == ADD || opcode == MULT) -> A dist { A_max_values :/ 80};
56     (opcode == ADD || opcode == MULT) -> B dist { B_max_values :/ 80};
57
58     ((opcode == OR || opcode == XOR) && red_op_A) -> A dist {walking_ones_t:=80,walking_ones_f:=20};
59     ((opcode == OR || opcode == XOR) && red_op_A) -> B==0;
60
61     ((opcode == OR || opcode == XOR) && red_op_B) -> B dist {walking_ones_t:=80,walking_ones_f:=20};
62     ((opcode == OR || opcode == XOR) && red_op_B) -> A ==0;
63 }
64
65 constraint opcode_dist {
66     opcode dist { [invalid6:invalid7] := 20,[OR:ROTATE] := 80};
67 }
68
69 constraint bypass_dist {
70     bypass_A dist { 0:=98 , 1:=2};
71     bypass_B dist { 0:=98 , 1:=2};
72 }
73
74 constraint red_op_dist {
75     red_op_A dist { 0:=50 , 1:=50};
76     red_op_B dist { 0:=50 , 1:=50};
77 }
78
79 constraint arr_c {
80     foreach (arr[i]){
81         arr[i] == i;
82     }
83 }
84
85 endclass
86
87 endpackage

```

alsu_scoreboard.sv X

alsu_scoreboard.sv

```
1 package alsu_sco_pkg;
2 import alsu_seq_item_pkg::*;
3 import uvm_pkg::*;
4 `include "uvm_macros.svh"
5
6 class alsu_sco extends uvm_scoreboard;
7     `uvm_component_utils(alsu_sco)
8
9     uvm_analysis_export #(alsu_seq_item) sb_export;
10    uvm_tlm_analysis_fifo #(alsu_seq_item) sb_fifo;
11    alsu_seq_item seq_item_sb;
12    logic [5:0] dataout_ref;
13    int error_count = 0;
14    int correct_count = 0;
15    function new (string name = "alsu_sco" , uvm_component parent = null);
16        super.new (name,parent);
17    endfunction
18
19    function void build_phase (uvm_phase phase);
20        super.build_phase (phase);
21        sb_export = new("sb_export",this);
22        sb_fifo = new("sb_fifo",this);
23    endfunction
24
25    function void connect_phase(uvm_phase phase);
26        super.connect_phase(phase);
27        sb_export.connect(sb_fifo.analysis_export);
28    endfunction
29
30    task run_phase (uvm_phase phase);
31        super.run_phase(phase);
32        forever begin
33            sb_fifo.get(seq_item_sb);
34            if ((seq_item_sb.out != seq_item_sb.out_ex) && (seq_item_sb.leds != seq_item_sb.leds_ex)) begin
35                `uvm_error("run_phase",$sformatf("compartion failed while ref = %b",seq_item_sb.out_ex))
36                error_count++;
37            end
38            else
39                correct_count++;
40            end
41        endtask
42
43    function void report_phase (uvm_phase phase);
44        super.report_phase(phase);
45        `uvm_info("report_phase",$sformatf("corect = %d",correct_count) , UVM_MEDIUM)
46        `uvm_info("report_phase",$sformatf("error = %d",error_count) , UVM_MEDIUM)
47    endfunction
48 endclass
```

alsu_config.sv

```
1  package alsu_conf_pkg;
2  import uvm_pkg::*;
3  `include "uvm_macros.svh"
4
5  class alsu_config extends uvm_object;
6      `uvm_object_utils(alsu_config)
7
8      virtual alsu_if alsu_vif;
9
10     function new (string name = "alsu_config");
11         super.new(name);
12     endfunction
13
14
15 endclass
16 endpackage
```

```

1 package alsu_seq_pkg;
2 import uvm_pkg::*;
3 import alsu_seq_item_pkg::*;
4 `include "uvm_macros.svh"
5
6 class alsu_reset_seq extends uvm_sequence #(alsu_seq_item);
7     `uvm_object_utils(alsu_reset_seq)
8     alsu_seq_item seq_item;
9
10     function new(string name = "alsu_reset_seq");
11         super.new(name);
12     endfunction
13
14     task body;
15         seq_item = alsu_seq_item::type_id::create("seq_item");
16         start_item(seq_item);
17         seq_item.rst = 1;
18         finish_item (seq_item);
19     endtask
20 endclass
21
22 class alsu_main1_seq extends uvm_sequence #(alsu_seq_item);
23     `uvm_object_utils(alsu_main1_seq)
24     alsu_seq_item seq_item;
25
26     function new(string name = "alsu_main1_seq");
27         super.new(name);
28     endfunction
29
30     task body;
31         seq_item = alsu_seq_item::type_id::create("seq_item");
32         seq_item.arr_c.constraint_mode(0);
33         repeat (99999) begin
34             start_item(seq_item);
35             assert (seq_item.randomize());
36             finish_item (seq_item);
37         end
38     endtask
39 endclass
40
41 class alsu_main2_seq extends uvm_sequence #(alsu_seq_item);
42     `uvm_object_utils(alsu_main2_seq)
43     alsu_seq_item seq_item;
44
45     function new(string name = "alsu_main2_seq");
46         super.new(name);
47     endfunction
48

```

```

48
49     task body;
50         seq_item = alsu_seq_item::type_id::create("seq_item");
51         seq_item.constraint_mode(0);
52         seq_item.arr_c.constraint_mode(1);
53         repeat (999) begin
54             start_item(seq_item);
55             assert (seq_item.randomize());
56             finish_item (seq_item);
57         end
58     endtask
59 endclass
60
61 class alsu_main3_seq extends uvm_sequence #(alsu_seq_item);
62     `uvm_object_utils(alsu_main3_seq)
63     alsu_seq_item seq_item;
64
65     function new(string name = "alsu_main3_seq");
66         super.new(name);
67     endfunction
68
69     task body;
70         seq_item = alsu_seq_item::type_id::create("seq_item");
71         seq_item.constraint_mode(0);
72         seq_item.arr_c.constraint_mode(1);
73         for (int i=0 ; i<6 ; i = i+1) begin
74             start_item(seq_item);
75             assert (seq_item.randomize());
76             seq_item.opcode = seq_item.arr[i];
77             finish_item (seq_item);
78         end
79     endtask
80 endclass
81 endpackage

```

```

alsu_sequencer.sv x
alsu_sequencer.sv
1  package alsu_sequencer_pkg;
2  import uvm_pkg::*;
3  import alsu_seq_item_pkg::*;
4  `include "uvm_macros.svh"
5
6  class alsu_sequencer extends uvm_sequencer #(alsu_seq_item);
7      `uvm_component_utils(alsu_sequencer);
8
9  function new (string name = "alsu_sequencer" , uvm_component parent = null);
10     super.new (name,parent);
11 endfunction
12 endclass
13
14 endpackage

```

alsu_driver.sv

```
1 package alsu_driver_pkg;
2 import uvm_pkg::*;
3 import alsu_seq_item_pkg::*;
4 `include "uvm_macros.svh"
5
6 class alsu_driver extends uvm_driver#(alsu_seq_item);
7     `uvm_component_utils (alsu_driver)
8
9     virtual alsu_if alsu_vif;
10    alsu_seq_item stim_seq_item;
11
12    function new (string name = "alsu_driver" , uvm_component parent = null);
13        super.new (name,parent);
14    endfunction
15
16    task run_phase (uvm_phase phase);
17        super.run_phase(phase);
18        forever begin
19            stim_seq_item = alsu_seq_item::type_id::create ("stim_seq_item");
20            seq_item_port.get_next_item(stim_seq_item);
21            alsu_vif.cin = stim_seq_item.cin;
22            alsu_vif.rst = stim_seq_item.rst;
23            alsu_vif.red_op_A = stim_seq_item.red_op_A;
24            alsu_vif.red_op_B = stim_seq_item.red_op_B;
25            alsu_vif.bypass_A = stim_seq_item.bypass_A;
26            alsu_vif.bypass_B = stim_seq_item.bypass_B;
27            alsu_vif.direction = stim_seq_item.direction;
28            alsu_vif.serial_in = stim_seq_item.serial_in;
29            alsu_vif.opcode = stim_seq_item.opcode;
30            alsu_vif.A = stim_seq_item.A;
31            alsu_vif.B = stim_seq_item.B;
32            @(negedge alsu_vif.clk);
33            seq_item_port.item_done();
34            `uvm_info("run_phase" , stim_seq_item.convert2string_stimulus(),UVM_HIGH)
35        end
36    endtask
37
38
39 endclass
40
41
42 endpackage
```

```

1  package alsu_monitor_pkg;
2  import uvm_pkg::*;
3  import alsu_seq_item_pkg::*;
4  `include "uvm_macros.svh"
5
6  class alsu_monitor extends uvm_monitor;
7  `uvm_component_utils (alsu_monitor)
8
9  virtual alsu_if alsu_vif;
10 alsu_seq_item rsp_seq_item;
11 uvm_analysis_port #(alsu_seq_item) mon_ap;
12
13 function new (string name = "alsu_monitor" , uvm_component parent = null);
14     super.new (name,parent);
15 endfunction
16
17 function void build_phase (uvm_phase phase);
18     super.build_phase(phase);
19     mon_ap = new("mon_ap",this);
20 endfunction
21
22 task run_phase (uvm_phase phase);
23     super.run_phase(phase);
24     forever begin
25         rsp_seq_item = alsu_seq_item::type_id::create ("rsp_seq_item");
26         @(negedge alsu_vif.clk);
27         rsp_seq_item.cin = alsu_vif.cin;
28         rsp_seq_item.rst = alsu_vif.rst;
29         rsp_seq_item.red_op_A = alsu_vif.red_op_A;
30         rsp_seq_item.red_op_B = alsu_vif.red_op_B;
31         rsp_seq_item.bypass_A = alsu_vif.bypass_A;
32         rsp_seq_item.bypass_B = alsu_vif.bypass_B;
33         rsp_seq_item.direction = alsu_vif.direction;
34         rsp_seq_item.serial_in = alsu_vif.serial_in;
35         rsp_seq_item.opcode = opcode_e'(alsu_vif.opcode);
36         rsp_seq_item.A = alsu_vif.A;
37         rsp_seq_item.B = alsu_vif.B;
38         mon_ap.write(rsp_seq_item);
39         `uvm_info("run_phase" , rsp_seq_item.convert2string_stimulus(),UVM_HIGH)
40     end
41 endtask
42 endclass
43 endpackage

```



```

alsu_agent.sv
1  package alsu_agent_pkg;
2  import alsu_driver_pkg::*;
3  import alsu_monitor_pkg::*;
4  import alsu_sequencer_pkg::*;
5  import alsu_seq_item_pkg::*;
6  import alsu_conf_pkg::*;
7  import uvm_pkg::*;
8  `include "uvm_macros.svh"
9
10 class alsu_agent extends uvm_agent;
11     `uvm_component_utils(alsu_agent)
12
13     alsu_sequencer sqr;
14     alsu_driver drv;
15     alsu_monitor mon;
16     alsu_config alsu_cfg;
17     uvm_analysis_port #(alsu_seq_item) agt_ap;
18
19     function new (string name = "alsu_agent" , uvm_component parent = null);
20         super.new (name,parent);
21     endfunction
22
23     function void build_phase (uvm_phase phase);
24         super.build_phase (phase);
25         if (!uvm_config_db #(alsu_config)::get(this,"","CFG",alsu_cfg))
26             `uvm_fatal ("build_phase","test - unable to get the configuration");
27
28         sqr = alsu_sequencer::type_id::create ("sqr",this);
29         drv = alsu_driver::type_id::create ("drv",this);
30         mon = alsu_monitor::type_id::create ("mon",this);
31         agt_ap = new("agt_ap",this);
32     endfunction
33
34     function void connect_phase(uvm_phase phase);
35         drv.alsu_vif = alsu_cfg.alsu_vif;
36         mon.alsu_vif = alsu_cfg.alsu_vif;
37         drv.seq_item_port.connect(sqr.seq_item_export);
38         mon.mon_ap.connect(agt_ap);
39     endfunction
40 endclass
41
42 endpackage

```

```

1
2  package alsu_env_pkg;
3  import alsu_agent_pkg::*;
4  import alsu_sco_pkg::*;
5  import alsu_coverage_pkg::*;
6  import uvm_pkg::*;
7  `include "uvm_macros.svh"
8
9  class alsu_env extends uvm_env;
10     `uvm_component_utils (alsu_env)
11
12     alsu_agent agt;
13     alsu_sco sb;
14     alsu_coverage cov;
15
16     function new (string name = "alsu_env" , uvm_component parent = null);
17         super.new (name,parent);
18     endfunction
19
20     function void build_phase (uvm_phase phase);
21         super.build_phase (phase);
22         agt = alsu_agent::type_id::create ("agt",this);
23         sb = alsu_sco::type_id::create ("sb",this);
24         cov = alsu_coverage::type_id::create ("cov",this);
25     endfunction
26
27     function void connect_phase (uvm_phase phase);
28         agt.agt_ap.connect(sb.sb_export);
29         agt.agt_ap.connect(cov.cov_export);
30     endfunction
31
32 endclass
33 endpackage

```

```

1
2 package alsu_test_pkg;
3 import alsu_env_pkg::*;
4 import alsu_conf_pkg::*;
5 import alsu_seq_pkg::*;
6 import uvm_pkg::*;
7 `include "uvm_macros.svh"
8
9 class alsu_test extends uvm_test;
10
11     `uvm_component_utils (alsu_test)
12
13     alsu_env env;
14     alsu_config alsu_cfg;
15     virtual alsu_if alsu_vif;
16     alsu_main1_seq main1_seq;
17     alsu_main2_seq main2_seq;
18     alsu_main3_seq main3_seq;
19     alsu_reset_seq reset_seq;
20
21     function new (string name = "alsu_env" , uvm_component parent = null);
22         super.new (name,parent);
23     endfunction
24
25     function void build_phase (uvm_phase phase);
26         super.build_phase(phase);
27         env = alsu_env::type_id::create ("env",this);
28         alsu_cfg = alsu_config::type_id::create ("alsu_cfg");
29         main1_seq = alsu_main1_seq::type_id::create("main1_seq");
30         main2_seq = alsu_main2_seq::type_id::create("main2_seq");
31         main3_seq = alsu_main3_seq::type_id::create("main3_seq");
32         reset_seq = alsu_reset_seq::type_id::create("reset_seq");
33
34         if (!uvm_config_db#(virtual alsu_if)::get(this,"", "alsu_IF",alsu_cfg.alsu_vif))
35             `uvm_fatal ("build_phase","test - unable to get the virtual interface");
36
37         uvm_config_db#(alsu_config)::set(this,"*", "CFG",alsu_cfg);
38
39     endfunction
40
41     task run_phase (uvm_phase phase);
42         super.run_phase(phase);
43         phase.raise_objection(this);
44

```

```

44
45     //reset
46     `uvm_info ("run_phase" , "reset asserted" , UVM_LOW)
47     reset_seq.start(env.agt.sqr);
48     `uvm_info ("run_phase" , "reset deasserted" , UVM_LOW)
49
50     //main1
51     `uvm_info ("run_phase" , "stimulus generation started 1" , UVM_LOW)
52     main1_seq.start(env.agt.sqr);
53     `uvm_info ("run_phase" , "stimulus generation ended 1" , UVM_LOW)
54
55     //main2
56     `uvm_info ("run_phase" , "stimulus generation started 2" , UVM_LOW)
57     main2_seq.start(env.agt.sqr);
58     `uvm_info ("run_phase" , "stimulus generation ended 2" , UVM_LOW)
59
60     //main3
61     `uvm_info ("run_phase" , "stimulus generation started 3" , UVM_LOW)
62     main3_seq.start(env.agt.sqr);
63     `uvm_info ("run_phase" , "stimulus generation ended 3" , UVM_LOW)
64
65     //reset
66     `uvm_info ("run_phase" , "reset asserted" , UVM_LOW)
67     reset_seq.start(env.agt.sqr);
68     `uvm_info ("run_phase" , "reset deasserted" , UVM_LOW)
69
70     phase.drop_objection(this);
71 endtask
72
73 endclass: alsu_test
74 endpackage

```

```

1
2 import uvm_pkg::*;
3 `include "uvm_macros.svh"
4 import alsu_test_pkg::*;
5
6 module alsu_top();
7
8     bit clk;
9
10    initial begin
11        forever begin
12            #1 clk = ~clk;
13        end
14    end
15
16    alsu_if alsuif (clk);
17    ALSU dut (alsuif);
18    ALSU_golden_model dut1 (alsuif);
19    bind ALSU ALSU_sva assertion (alsuif);
20
21    initial begin
22        uvm_config_db#(virtual alsu_if)::set(null,"uvm_test_top","alsu_IF",alsuif);
23        run_test("alsu_test");
24    end
25
26 endmodule

```

```

# Time: 201980 ns Iteration: 2 Region: /uvm_pkg::uvm_task_phase::execute
# UVM_INFO alsu_test.sv(58) @ 201980: uvm_test_top [run_phase] stimulus generation ended 2
# UVM_INFO alsu_test.sv(61) @ 201980: uvm_test_top [run_phase] stimulus generation started 3
# UVM_INFO alsu_test.sv(63) @ 202010: uvm_test_top [run_phase] stimulus generation ended 3
# UVM_INFO alsu_test.sv(66) @ 202010: uvm_test_top [run_phase] reset asserted
# UVM_INFO alsu_test.sv(68) @ 202012: uvm_test_top [run_phase] reset deasserted
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1267) @ 202012: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
# UVM_INFO alsu_scoreboard.sv(48) @ 202012: uvm_test_top.env.sb [report_phase] correct = 101006
# UVM_INFO alsu_scoreboard.sv(49) @ 202012: uvm_test_top.env.sb [report_phase] error = 0
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO : 16
# UVM_WARNING : 0
# UVM_ERROR : 0
# UVM_FATAL : 0
# ** Report counts by id
# [Questa UVM] 2
# [RNTST] 1
# [TEST_DONE] 1
# [report_phase] 2
# [run_phase] 10
# ** Note: $finish : C:/questasim64_2021.1/win64/./verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
# Time: 202012 ns Iteration: 61 Instance: /alsu_top
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_2021.1/win64/./verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430

```

Name	Class Type	Coverage	Goal	% of Goal	Status	Included	Merge_instances
/alsu_coverage_pk...		100.00%					
TYPE cvr_gp		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	auto(0)
CVP cvr_g...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
CVP cvr_g...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
CVP cvr_g...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
CVP cvr_g...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
CVP cvr_g...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
CVP cvr_g...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
CVP cvr_g...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
CVP cvr_g...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
CVP cvr_g...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
CROSS cvr...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
CROSS cvr...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
CROSS cvr...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
CROSS cvr...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
CROSS cvr...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
CROSS cvr...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	
INST Vals...		100.00%	100	100.00...	<div></div>	<input checked="" type="checkbox"/>	

Assertions													+	⌵	⌶
Name	Assertion Type	Language	Enable	Failure Count	Pass Count	Active Count	Memory	Peak Memory	Peak Memory Time	Cumulative Threads	ATV	Assertion Expression	Includ...		
/uvm_pkg:/uvm_re... Immediate	SVA	on	0	0	0	-	-	-	-	-	off	assert (\$cast(seq,o))	✓		
/uvm_pkg:/uvm_re... Immediate	SVA	on	0	0	0	-	-	-	-	-	off	assert (\$cast(seq,o))	✗		
/alsu_seq_pkg:/als... Immediate	SVA	on	0	1	-	-	-	-	-	-	off	assert (randomize(...))	✓		
/alsu_seq_pkg:/als... Immediate	SVA	on	0	1	-	-	-	-	-	-	off	assert (randomize(...))	✓		
/alsu_seq_pkg:/als... Immediate	SVA	on	0	1	-	-	-	-	-	-	off	assert (randomize(...))	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) (alsuf...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0B	0 ns	0	off	assert (@(posedge alsuf.clk) disab...	✓		
+ /alsu_top/dut/asse... Concurrent	SVA	on	0	1	-	0B	0B	0							