

Digital Verification using SV and UVM Assignment-5

Name: Fares Khalaf Sultan

➤ Part#1:

❖ UVM Environment:

```
top_module_tb.sv
import alsu_test_pkg::*;
import uvm_pkg::*;
`include "uvm_macros.svh"

module top_tb();

    bit clk;

    initial begin
        clk = 0;
        forever begin
            #1 clk = ~clk;
        end
    end

    ALSU_if alsu_if(clk);

    ALSU DUT (
        .A(alsu_if.A), .B(alsu_if.B), .cin(alsu_if.cin), .serial_in(alsu_if.serial_in),
        .red_op_A(alsu_if.red_op_A), .red_op_B(alsu_if.red_op_B), .opcode(alsu_if.opcode),
        .bypass_A(alsu_if.bypass_A), .bypass_B(alsu_if.bypass_B), .clk(clk), .rst(alsu_if.rst),
        .direction(alsu_if.direction), .leds(alsu_if.leds), .out(alsu_if.out)
    );

    initial begin
        run_test("alsu_test");
    end

endmodule
```

```
ALSU_if.sv
interface ALSU_if(input bit clk);

    parameter INPUT_PRIORITY = "A";
    parameter FULL_ADDER = "ON";
    logic cin, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
    logic [2:0] opcode;
    logic signed [2:0] A, B;
    logic [15:0] leds;
    logic signed [5:0] out;

endinterface
```

```

1  `ifndef ALSU_TEST_PKG_SV
2  `define ALSU_TEST_PKG_SV
3
4  package alsu_test_pkg;
5      import uvm_pkg::*;
6      import alsu_env_pkg::*;
7      `include "uvm_macros.svh"
8
9      class alsu_test extends uvm_test;
10         `uvm_component_utils(alsu_test)
11
12         alsu_env env0;
13
14         function new(string name = "alsu_test", uvm_component parent = null);
15             super.new(name, parent);
16         endfunction
17
18         function void build_phase(uvm_phase phase);
19             super.build_phase(phase);
20
21             env0 = alsu_env::type_id::create("env0", this);
22         endfunction
23
24         task run_phase(uvm_phase phase);
25             super.run_phase(phase);
26
27             phase.raise_objection(this);
28
29             #100; `uvm_info("run_phase", "Welcome to UVM - env0 created", UVM_MEDIUM)
30
31             phase.drop_objection(this);
32         endtask
33     endclass
34
35
36 endpackage
37 `endif

```

```

1  `ifndef ALSU_ENV_PKG_SV
2  `define ALSU_ENV_PKG_SV
3
4  package alsu_env_pkg;
5      import uvm_pkg::*;
6      `include "uvm_macros.svh"
7
8      class alsu_env extends uvm_env;
9         `uvm_component_utils(alsu_env)
10
11         function new(string name = "alsu_env", uvm_component parent = null);
12             super.new(name, parent);
13         endfunction
14     endclass
15
16 endpackage
17
18
19 `endif

```

❖ Do file:

```
# run.do
1  vlib work
2  vlog -f src_files.list
3  vsim -voptargs=+acc work.top_tb -classdebug -uvmcontrol=all
4  add wave /top_tb/alsu_if/*
5  run -all
```

❖ Results:

```
# -----
# UVM-1.1d
# (C) 2007-2013 Mentor Graphics Corporation
# (C) 2007-2013 Cadence Design Systems, Inc.
# (C) 2006-2013 Synopsys, Inc.
# (C) 2011-2013 Cypress Semiconductor Corp.
# -----
#
# ***** IMPORTANT RELEASE NOTES *****
#
# You are using a version of the UVM library that has been compiled
# with 'UVM_NO_DEPRECATED' undefined.
# See http://www.eda.org/svdb/view.php?id=3313 for more details.
#
# You are using a version of the UVM library that has been compiled
# with 'UVM_OBJECT_MUST_HAVE_CONSTRUCTOR' undefined.
# See http://www.eda.org/svdb/view.php?id=3770 for more details.
#
# (Specify +UVM_NO_RELNOTES to turn off this notice)
#
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(277) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(278) @ 0: reporter [Questa UVM] questa_uvm::init(all)
# UVM_INFO @ 0: reporter [RNTST] Running test alsu_test...
# UVM_INFO alsu_test.sv(31) @ 100: uvm_test_top [run_phase] Welcome to UVM - env0 created
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_object.svh(1267) @ 100: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO : 5
# UVM_WARNING : 0
# UVM_ERROR : 0
# UVM_FATAL : 0
#
# ** Report counts by id
# [Questa UVM] 4
# [RNTST] 1
# [TEST_DONE] 1
# [run_phase] 1
#
# ** Note: $finish : C:/questasim64_2021.1/win64/./verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
# Time: 100 ns Iteration: 54 Instance: /top_tb
# 1
# 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
```

➤ Part#2: Driving the interface:

❖ UVM Environment:

```
1 import alsu_test_pkg::*;
2 import uvm_pkg::*;
3 `include "uvm_macros.svh"
4
5 module top_tb();
6
7     bit clk;
8
9     initial begin
10         clk = 0;
11         forever begin
12             #1 clk = ~clk;
13         end
14     end
15
16     ALSU_if alsu_if(clk);
17
18     ALSU DUT (
19         .A(alsu_if.A), .B(alsu_if.B), .cin(alsu_if.cin), .serial_in(alsu_if.serial_in),
20         .red_op_A(alsu_if.red_op_A), .red_op_B(alsu_if.red_op_B), .opcode(alsu_if.opcode),
21         .bypass_A(alsu_if.bypass_A), .bypass_B(alsu_if.bypass_B), .clk(clk), .rst(alsu_if.rst),
22         .direction(alsu_if.direction), .leds(alsu_if.leds), .out(alsu_if.out)
23     );
24
25     initial begin
26         uvm_config_db#(virtual ALSU_if)::set(null,"uvm_test_top","ALSU_IF",alsu_if);
27         run_test("alsu_test");
28     end
29 endmodule
```

```
ALSU_if.sv
interface ALSU_if(input bit clk);

    parameter INPUT_PRIORITY = "A";
    parameter FULL_ADDER = "ON";
    logic cin, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
    logic [2:0] opcode;
    logic signed [2:0] A, B;
    logic [15:0] leds;
    logic signed [5:0] out;

endinterface
```

```

4 package alsu_test_pkg;
5     import uvm_pkg::*;
6     import alsu_env_pkg::*;
7     `include "uvm_macros.svh"
8
9     class alsu_test extends uvm_test;
10         `uvm_component_utils(alsu_test)
11
12         alsu_env env0;
13         virtual ALSU_if alsu_test_vif;
14
15         function new(string name = "alsu_test", 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
22             if(!uvm_config_db #(virtual ALSU_if)::get(this, "", "ALSU_IF", alsu_test_vif))
23                 `uvm_fatal("build_phase", "Test - unable to get the virtual interface")
24             uvm_config_db #(virtual ALSU_if)::set(this, "*", "CFG", alsu_test_vif);
25
26             env0 = alsu_env::type_id::create("env0", this);
27         endfunction
28
29         task run_phase(uvm_phase phase);
30             super.run_phase(phase);
31             phase.raise_objection(this);
32
33             #100; `uvm_info("run_phase", "Inside the ALSU test", UVM_MEDIUM)
34
35             phase.drop_objection(this);
36         endtask
37     endclass
38
39 endpackage

```

```

3
4 package alsu_env_pkg;
5     import uvm_pkg::*;
6     import alsu_driver_pkg::*;
7     `include "uvm_macros.svh"
8
9     class alsu_env extends uvm_env;
10         `uvm_component_utils(alsu_env)
11         alsu_driver driver;
12
13         function new(string name = "alsu_env", 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
20             driver = alsu_driver::type_id::create("driver", this);
21         endfunction
22     endclass
23
24 endpackage
25

```

```

3
4 package alsu_driver_pkg;
5 import uvm_pkg::*;
6 `include "uvm_macros.svh"
7
8 class alsu_driver extends uvm_driver;
9     `uvm_component_utils(alsu_driver)
10
11     virtual ALSU_if alsu_driver_vif;
12
13     function new(string name = "alsu_driver", 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
20         if(!uvm_config_db #(virtual ALSU_if)::get(this, "", "CFG", alsu_driver_vif))
21             `uvm_fatal("build_phase", "Driver - unable to get the virtual interface")
22
23     endfunction
24

```

```

24
25     task run_phase(uvm_phase phase);
26         super.run_phase(phase);
27
28         phase.raise_objection(this);
29
30         alsu_driver_vif.rst = 1;
31         @(negedge alsu_driver_vif.clk);
32         alsu_driver_vif.rst = 0;
33
34         repeat(10) begin
35             alsu_driver_vif.cin = $random();
36             alsu_driver_vif.red_op_A = $random();
37             alsu_driver_vif.red_op_B = $random();
38             alsu_driver_vif.bypass_A = $random();
39             alsu_driver_vif.bypass_B = $random();
40             alsu_driver_vif.direction = $random();
41             alsu_driver_vif.serial_in = $random();
42             alsu_driver_vif.opcode = $random();
43             alsu_driver_vif.A = $random();
44             alsu_driver_vif.B = $random();
45             @(negedge alsu_driver_vif.clk);
46         end
47
48         phase.drop_objection(this);
49     endtask
50 endclass
51
52 endpackage

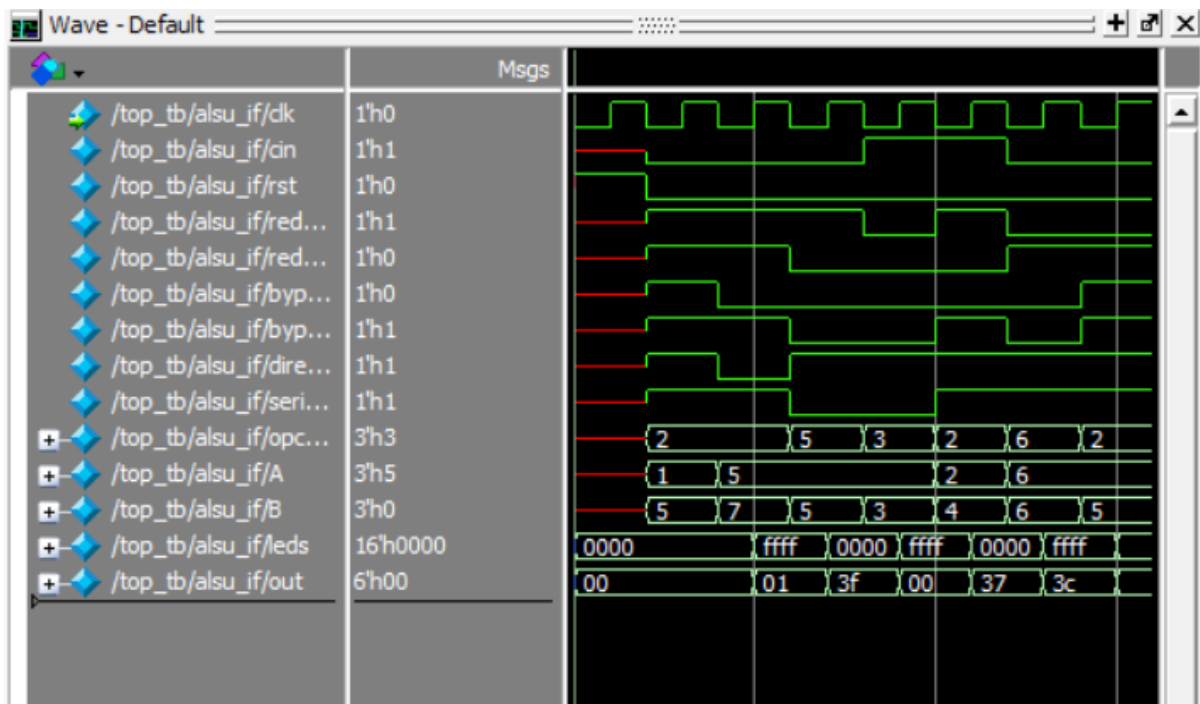
```

❖ Results:

```

#
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(277) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(278) @ 0: reporter [Questa UVM] questa_uvm::init(all)
# UVM_INFO @ 0: reporter [RNTST] Running test alsu_test...
# UVM_INFO alsu_test.sv(34) @ 100: uvm_test_top [run_phase] Inside the ALSU test
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_object.svh(1267) @ 100: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
#
--- UVM Report Summary ---
#
** Report counts by severity
# UVM_INFO : 5
# UVM_WARNING : 0
# UVM_ERROR : 0
# UVM_FATAL : 0
#
** Report counts by id
# [Questa UVM] 2
# [RNTST] 1
# [TEST_DONE] 1
# [run_phase] 1
#
** Note: $finish : C:/questasim64_2021.1/win64/./verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
# Time: 100 ns Iteration: 54 Instance: /top_tb
#
# 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

```



➤ Part#3: config object:

```
1 import alsu_test_pkg::*;
2 import uvm_pkg::*;
3 `include "uvm_macros.svh"
4
5 module top_tb();
6
7     bit clk;
8
9     initial begin
10         clk = 0;
11         forever begin
12             #1 clk = ~clk;
13         end
14     end
15
16     ALSU_if alsu_if(clk);
17
18     ALSU DUT (
19         .A(alsu_if.A), .B(alsu_if.B), .cin(alsu_if.cin), .serial_in(alsu_if.serial_in),
20         .red_op_A(alsu_if.red_op_A), .red_op_B(alsu_if.red_op_B), .opcode(alsu_if.opcode),
21         .bypass_A(alsu_if.bypass_A), .bypass_B(alsu_if.bypass_B), .clk(clk), .rst(alsu_if.rst),
22         .direction(alsu_if.direction), .leds(alsu_if.leds), .out(alsu_if.out)
23     );
24
25     initial begin
26         uvm_config_db#(virtual ALSU_if)::set(null,"uvm_test_top","ALSU_IF",alsu_if);
27         run_test("alsu_test");
28     end
29 endmodule
```

ALSU_if.sv

```
interface ALSU_if(input bit clk);

    parameter INPUT_PRIORITY = "A";
    parameter FULL_ADDER = "ON";
    logic cin, rst, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
    logic [2:0] opcode;
    logic signed [2:0] A, B;
    logic [15:0] leds;
    logic signed [5:0] out;

endinterface
```

- Test:

```
alsu_test.sv
4 package alsu_test_pkg;
5   import uvm_pkg::*;
6   import alsu_env_pkg::*;
7   import config_obj_pkg::*;
8   `include "uvm_macros.svh"
9
10  class alsu_test extends uvm_test;
11    `uvm_component_utils(alsu_test)
12
13    alsu_env env0;
14    alsu_config alsu_config_obj_test;
15
16    function new(string name = "alsu_test", 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      alsu_config_obj_test = alsu_config::type_id::create("alsu_config_obj_test");
23
24      if(!uvm_config_db #(virtual ALSU_if)::get(this, "", "ALSU_IF", alsu_config_obj_test.alsu_config_if))
25        `uvm_fatal("build_phase", "Test - unable to get the virtual interface")
26      uvm_config_db #(alsu_config)::set(this, "*", "CFG", alsu_config_obj_test);
27
28      env0 = alsu_env::type_id::create("env0", this);
29    endfunction
30
31    task run_phase(uvm_phase phase);
32      super.run_phase(phase);
33      phase.raise_objection(this);
34
35      #100; `uvm_info("run_phase", "Inside the ALSU test", UVM_MEDIUM)
36
37      phase.drop_objection(this);
38    endtask
39  endclass
40 endpackage
```

- Config object:

```
package config_obj_pkg;
  import uvm_pkg::*;
  `include "uvm_macros.svh"

  class alsu_config extends uvm_object;
    `uvm_object_utils(alsu_config)

    virtual ALSU_if alsu_config_if;

    function new(string name = "alsu_config_obj");
      super.new(name);
    endfunction
  endclass

endpackage
```

- Env:

```
4 package alsu_env_pkg;
5     import uvm_pkg::*;
6     import alsu_driver_pkg::*;
7     `include "uvm_macros.svh"
8
9     class alsu_env extends uvm_env;
10         `uvm_component_utils(alsu_env)
11
12         alsu_driver driver;
13
14         function new(string name = "alsu_env", uvm_component parent = null);
15             super.new(name, parent);
16         endfunction
17
18         function void build_phase(uvm_phase phase);
19             super.build_phase(phase);
20
21             driver = alsu_driver::type_id::create("driver", this);
22         endfunction
23     endclass
24
25 endpackage
```

- Driver:

```
4 package alsu_driver_pkg;
5     import uvm_pkg::*;
6     import config_obj_pkg::*;
7
8     `include "uvm_macros.svh"
9
10    class alsu_driver extends uvm_driver;
11        `uvm_component_utils(alsu_driver)
12
13        virtual ALSU_if alsu_driver_vif;
14        alsu_config alsu_config_obj_driver;
15
16        function new(string name = "alsu_driver", 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
23            if(!uvm_config_db #(alsu_config)::get(this, "", "CFG", alsu_config_obj_driver))
24                `uvm_fatal("build_phase", "Driver - unable to get the virtual interface")
25
26        endfunction
27
28        function void connect_phase(uvm_phase phase);
29            super.connect_phase(phase);
30
31            alsu_driver_vif = alsu_config_obj_driver.alsu_config_if;
32
33        endfunction
34    endclass
```

```

35     task run_phase(uvm_phase phase);
36         super.run_phase(phase);
37
38         phase.raise_objection(this);
39
40         alsu_driver_vif.rst = 1;
41         @(negedge alsu_driver_vif.clk);
42         alsu_driver_vif.rst = 0;
43
44         repeat(10) begin
45             alsu_driver_vif.cin = $random();
46             alsu_driver_vif.red_op_A = $random();
47             alsu_driver_vif.red_op_B = $random();
48             alsu_driver_vif.bypass_A = $random();
49             alsu_driver_vif.bypass_B = $random();
50             alsu_driver_vif.direction = $random();
51             alsu_driver_vif.serial_in = $random();
52             alsu_driver_vif.opcode = $random();
53             alsu_driver_vif.A = $random();
54             alsu_driver_vif.B = $random();
55             @(negedge alsu_driver_vif.clk);
56         end
57
58         phase.drop_objection(this);
59     endtask
60 endclass
61 endpackage

```

❖ Result:

```
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(277) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.3
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(278) @ 0: reporter [Questa UVM]  questa_uvm::init(all)
# UVM_INFO @ 0: reporter [RNTST] Running test alsu_test...
# UVM_INFO alsu_test.sv(36) @ 100: uvm_test_top [run_phase] Inside the ALSU test
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_object.svh(1267) @ 100: reporter [TEST_DONE] 'run' phase is ready to proceed to the 'extract' phase
#
# --- UVM Report Summary ---
#
# ** Report counts by severity
# UVM_INFO : 5
# UVM_WARNING : 0
# UVM_ERROR : 0
# UVM_FATAL : 0
# ** Report counts by id
# [Questa UVM] 2
# [RNTST] 1
# [TEST_DONE] 1
# [run_phase] 1
# ** Note: $finish : C:/questasim64_2021.1/win64/./verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
# Time: 100 ns Iteration: 54 Instance: /top_tb
# 1
# 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
```

Waveform:

