1.

1.1. Interface:

```
interface alsu_if(clk);
    input clk;
    logic rst;
    logic cin;
    logic red_op_A;
    logic red_op_B;
    logic bypass_A;
    logic bypass_B;
    logic direction;
    logic serial_in;
    logic [2:0] opcode;
    logic signed [2:0] A;
    logic signed [2:0] B;
    logic [15:0] leds;
    logic signed [5:0] out;
endinterface: alsu_if
```

1.2. Do file:

```
vlib work
vlog -f src_files.list
vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all
add wave /top/alsuif/*
run -all
```

1.3. Top file:

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

module top;
    // Clock generation
    bit clk;
    initial begin
        forever begin
            #1 clk = ~clk;
        end
    end

    // Instantiate the interface
    alsu_if alsuif(clk);

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

    // Run the test
    initial begin
        run_test("alsu_test");
    end
endmodule
```

1.4. Test file:

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

class alsu_test extends uvm_test;
    `uvm_component_utils(alsu_test)

    alsu_env env;

    function new(string name = "alsu_test", uvm_component parent = null);
        super.new(name, parent);
    endfunction

    function void build_phase(uvm_phase phase);
        super.build_phase(phase);
        env = alsu_env::type_id::create("env", this);
    endfunction

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        phase.raise_objection(this);
        #100;
        `uvm_info("TEST", "Inside the ALSU test", UVM_MEDIUM)
        phase.drop_objection(this);
    endtask
endclass: alsu_test
endpackage: alsu_test_pkg
```

1.5. Env. file:

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

class alsu_env extends uvm_env;
    `uvm_component_utils(alsu_env)

    function new(string name = "alsu_env", uvm_component parent = null);
        super.new(name, parent);
    endfunction

endclass: alsu_env
endpackage: alsu_env_pkg
```

1.6. Transcript:

```
# 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(24) @ 100: uvm_test_top [TEST] Inside the ALSU test
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.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]     1
# [TEST_DONE]     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
```

2.

2.1. Top file:

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

module top;
    // Clock generation
    bit clk;
    initial begin
        forever begin
            #1 clk = ~clk;
        end
    end

    // Instantiate the interface
    alsu_if alsuif(clk);

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

    // Run the test
    initial begin
        uvm_config_db #(virtual alsu_if)::set(null, "uvm_test_top", "ALSU_IF",
alsuif);
        run_test("alsu_test");
    end

endmodule
```

2.2. Test file:

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

class alsu_test extends uvm_test;
    `uvm_component_utils(alsu_test)

    alsu_env env;
    virtual alsu_if alsu_test_vif;

    function new(string name = "alsu_test", uvm_component parent = null);
        super.new(name, parent);
    endfunction

    function void build_phase(uvm_phase phase);
        super.build_phase(phase);

        if (!uvm_config_db #(virtual alsu_if)::get(this, "", "ALSU_IF",
alsu_test_vif)) begin
            `uvm_fatal("NO_VIF", "Virtual interface ALSU_IF not found in
config DB")
        end

        uvm_config_db #(virtual alsu_if)::set(this, "*", "VIF",
alsu_test_vif);

        env = alsu_env::type_id::create("env", this);
    endfunction

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        phase.raise_objection(this);
        #100;
        `uvm_info("TEST", "Inside the ALSU test", UVM_MEDIUM)
        phase.drop_objection(this);
    endtask
endclass: alsu_test

endpackage: alsu_test_pkg
```

2.3. Env. file:

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

class alsu_env extends uvm_env;
    `uvm_component_utils(alsu_env)

    alsu_driver driver;

    function new(string name = "alsu_env", uvm_component parent = null);
        super.new(name, parent);
    endfunction

    function void build_phase(uvm_phase phase);
        super.build_phase(phase);

        driver = alsu_driver::type_id::create("driver", this);
    endfunction
endclass: alsu_env

endpackage: alsu_env_pkg
```

2.4. Driver file:

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

class alsu_driver extends uvm_driver;
    `uvm_component_utils(alsu_driver)

    virtual alsu_if alsu_driver_vif;

    function new(string name = "alsu_driver", uvm_component parent = null);
        super.new(name, parent);
    endfunction

    function void build_phase(uvm_phase phase);
        super.build_phase(phase);

        if (!uvm_config_db #(virtual alsu_if)::get(this, "", "VIF",
alsu_driver_vif)) begin
            `uvm_fatal("NO_VIF", "Virtual interface VIF not found in config
DB")
        end
    endfunction

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        phase.raise_objection(this);

        @(negedge alsu_driver_vif.clk);
        alsu_driver_vif.rst = 1;
        @(negedge alsu_driver_vif.clk);
        alsu_driver_vif.rst = 0;

        forever begin
            @(negedge alsu_driver_vif.clk);
            alsu_driver_vif.opcode = $random;
            alsu_driver_vif.A = $random;
            alsu_driver_vif.B = $random;
            alsu_driver_vif.cin = $random;
            alsu_driver_vif.red_op_A = $random;
            alsu_driver_vif.red_op_B = $random;
            alsu_driver_vif.bypass_A = $random;
            alsu_driver_vif.bypass_B = $random;
            alsu_driver_vif.direction = $random;
            alsu_driver_vif.serial_in = $random;
        end
```

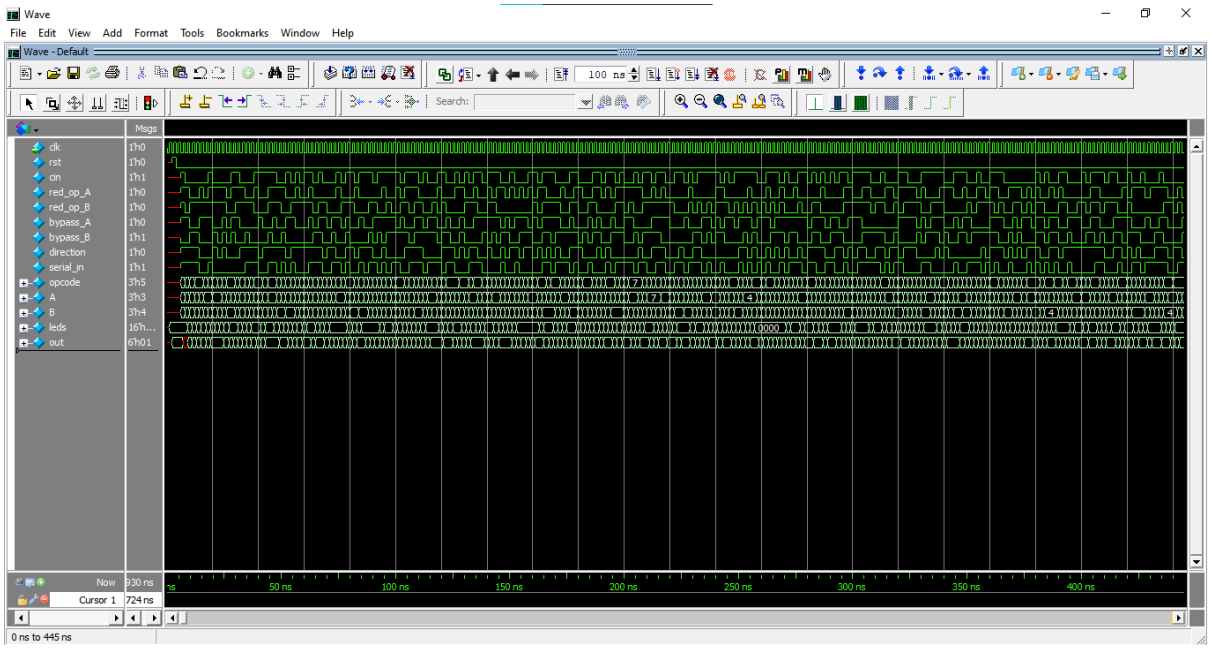
```
        phase.drop_objection(this);
    endtask
endclass: alsu_driver

endpackage: alsu_driver_pkg
```

2.5. Transcript:

```
# 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(32) @ 100: uvm_test_top [TEST] Inside the ALSU test
> quit -sim
```

2.6. Simulation Snippet:



3.

3.1. Test file:

```
package alsu_test_pkg;
import uvm_pkg::*;
import alsu_env_pkg::*;
import alsu_config_pkg::*;
`include "uvm_macros.svh"

class alsu_test extends uvm_test;
    `uvm_component_utils(alsu_test)

    alsu_env env;
    alsu_config_obj alsu_config_obj_test;

    function new(string name = "alsu_test", uvm_component parent = null);
        super.new(name, parent);
    endfunction

    function void build_phase(uvm_phase phase);
        super.build_phase(phase);

        env = alsu_env::type_id::create("env", this);
        alsu_config_obj_test =
alsu_config_obj::type_id::create("alsu_config_obj_test");

        if (!uvm_config_db #(virtual alsu_if)::get(this, "", "ALSU_IF",
alsu_config_obj_test.alsu_config_vif)) begin
            `uvm_fatal("NO_VIF", "Virtual interface ALSU_IF not found in
config DB")
        end

        uvm_config_db #(alsu_config_obj)::set(this, "*", "CFG",
alsu_config_obj_test);
    endfunction

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        phase.raise_objection(this);
        #100;
        `uvm_info("TEST", "Inside the ALSU test", UVM_MEDIUM)
        phase.drop_objection(this);
    endtask
endclass: alsu_test

endpackage: alsu_test_pkg
```

3.2. Driver file:

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

class alsu_driver extends uvm_driver;
    `uvm_component_utils(alsu_driver)

    virtual alsu_if alsu_driver_vif;
    alsu_config_obj alsu_config_obj_driver;

    function new(string name = "alsu_driver", uvm_component parent = null);
        super.new(name, parent);
    endfunction

    function void build_phase(uvm_phase phase);
        super.build_phase(phase);

        if (!uvm_config_db #(alsu_config_obj)::get(this, "", "CFG",
alsu_config_obj_driver)) begin
            `uvm_fatal("NO_CFG", "CFG object not found in config DB")
        end
    endfunction

    function void connect_phase(uvm_phase phase);
        super.connect_phase(phase);

        alsu_driver_vif = alsu_config_obj_driver.alsu_config_vif;
    endfunction

    task run_phase(uvm_phase phase);
        super.run_phase(phase);
        phase.raise_objection(this);

        @(negedge alsu_driver_vif.clk);
        alsu_driver_vif.rst = 1;
        @(negedge alsu_driver_vif.clk);
        alsu_driver_vif.rst = 0;

        forever begin
            @(negedge alsu_driver_vif.clk);
            alsu_driver_vif.opcode = $random;
            alsu_driver_vif.A = $random;
            alsu_driver_vif.B = $random;
            alsu_driver_vif.cin = $random;
            alsu_driver_vif.red_op_A = $random;
```

```
            alsu_driver_vif.red_op_B = $random;
            alsu_driver_vif.bypass_A = $random;
            alsu_driver_vif.bypass_B = $random;
            alsu_driver_vif.direction = $random;
            alsu_driver_vif.serial_in = $random;
        end

        phase.drop_objection(this);
    endtask
endclass: alsu_driver

endpackage: alsu_driver_pkg
```

3.3. Config. File:

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

class alsu_config_obj extends uvm_object;
    `uvm_object_utils(alsu_config_obj)

    virtual alsu_if alsu_config_vif;

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

endpackage: alsu_config_pkg
```

3.4. Transcript:

```
# 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 [TEST] Inside the ALSU test
> quit -sim
```

3.5. Simulation Snippet: