يوسف أحمد محمد ابراهيم

Assignment 5

Part 1:

1. Top:

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

module top();

    bit clk;
    always #1 clk = ~clk;

    alsu_if alsuif(clk);
    ALSU dut(alsuif);

    initial begin
        uvm_config_db#(virtual alsu_if)::set(null, "uvm_test_top", "ALSU_VIF", alsuif);
        run_test("alsu_test");

    end
endmodule
```

2. Test:

```
package alsu_test_pkg;
    import alsu_env_pkg::*;
    import uvm_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);
            if(!uvm_config_db#(virtual alsu_if)::get(this, "", "ALSU_VIF", cfg.alsu_config_vif))
                `uvm_fatal("ALSU_TEST", "Virtual interface not exist");

        endfunction

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

            phase.raise_objection(this);
            #100 `uvm_info("run_phase", "Inside the ALSU test", UVM_MEDIUM);
            phase.drop_objection(this);
        endtask

    endclass
endpackage
```

3. Env:

```
package alsu_env_pkg;
    import alsu_driver_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

        function void build_phase(uvm_phase phase);
            super.build_phase(phase);
        endfunction
    endclass
endpackage
```

4. Questasim:

```
# 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(27) @ 100: uvm_test_top [run_phase] 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_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
# 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:

1. alsu_top:

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

module top();

    bit clk;
    always #1 clk = ~clk;

    alsu_if alsuif(clk);
    ALSU dut(alsuif);

    initial begin
        uvm_config_db#(virtual alsu_if)::set(null, "uvm_test_top", "ALSU_VIF", alsuif);
        run_test("alsu_test");

    end
endmodule
```

2. Test:

```
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_VIF", alsu_test_vif))
                `uvm_fatal("ALSU_TEST", "Virtual interface not exist");
            uvm_config_db#(virtual alsu_if)::set(this, "*", "ALSU_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("run_phase", "Inside the ALSU test", UVM_MEDIUM);
            phase.drop_objection(this);
        endtask

    endclass
endpackage
```

3. env:

```
package alsu_env_pkg;
    import alsu_driver_pkg::*;
    import uvm_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
endpackage
```

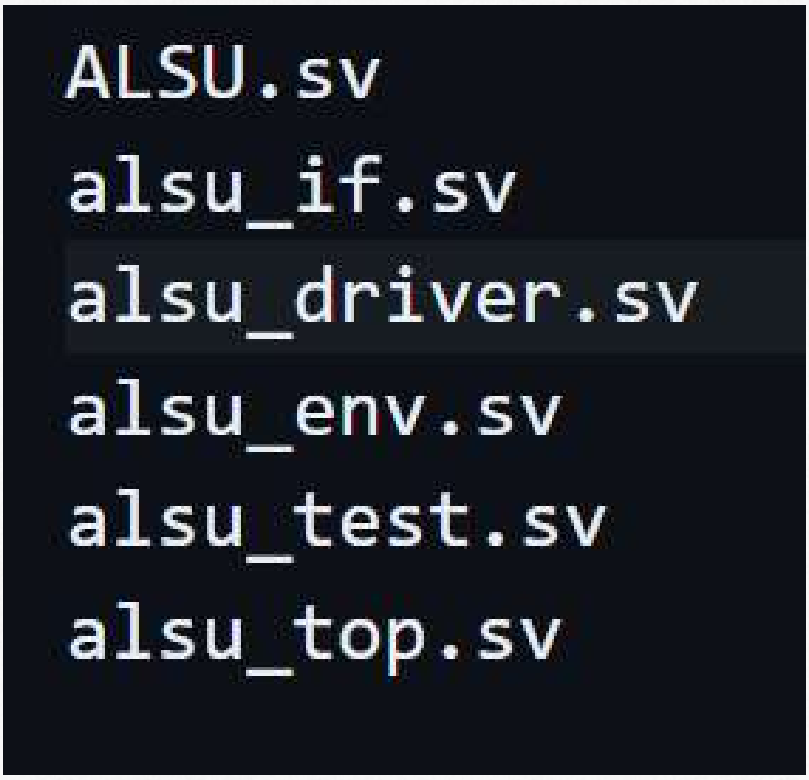
4. Driver:

```
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);
            uvm_config_db#(virtual alsu_if)::get(this, "", "ALSU_VIF", alsu_driver_vif);
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            alsu_driver_vif.rst = 1;
            @(negedge alsu_driver_vif.clk);
            alsu_driver_vif.rst = 0;
            forever begin
                alsu_driver_vif.A = $random;
                alsu_driver_vif.B = $random;
                alsu_driver_vif.cin = $random;
                alsu_driver_vif.serial_in = $random;
                alsu_driver_vif.red_op_A = $random;
                alsu_driver_vif.red_op_B = $random;
                alsu_driver_vif.opcode = $random;
                alsu_driver_vif.bypass_A = $random;
                alsu_driver_vif.bypass_B = $random;
                alsu_driver_vif.direction = $random;
                @(negedge alsu_driver_vif.clk);
            end
        endtask

    endclass
endpackage
```

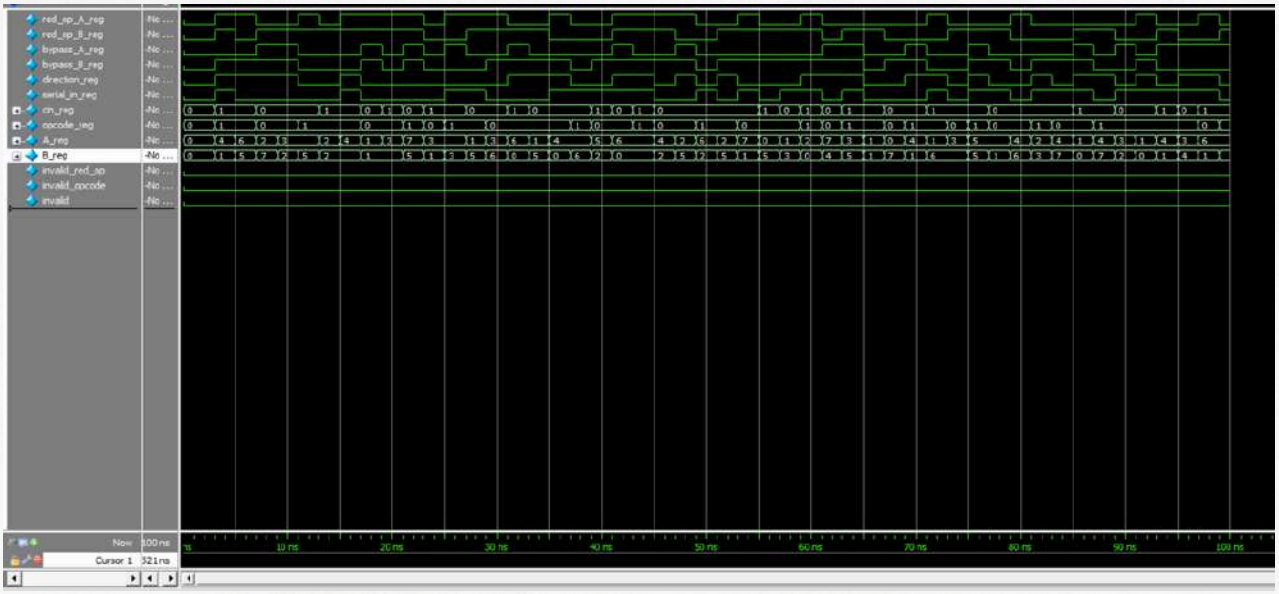
5. interface:

```
interface alsu_if(input clk);

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

endinterface
```

6.Src file:


ALSU.sv
alsu_if.sv
alsu_driver.sv
alsu_env.sv
alsu_test.sv
alsu_top.sv

7. Questasim:




```
# 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(27) @ 100: uvm_test_top [run_phase] 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_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
# 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 3:

1. Top:

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

module top();

    bit clk;
    always #1 clk = ~clk;

    alsu_if alsuif(clk);
    ALSU dut(alsuif);

    initial begin
        uvm_config_db#(virtual alsu_if)::set(null, "uvm_test_top", "ALSU_VIF", alsuif);
        run_test("alsu_test");

    end
endmodule
```

2. Interface:

```
interface alsu_if(input clk);

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

endinterface
```

3. Test component:

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

    class alsu_test extends uvm_test;
        `uvm_component_utils(alsu_test);
        alsu_env env;
        alsu_config cfg;
        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);
            cfg = alsu_config::type_id::create("cfg", this);
            env = alsu_env::type_id::create("env", this);
            if(!uvm_config_db#(virtual alsu_if)::get(this, "", "ALSU_VIF", cfg.alsu_config_vif))
                `uvm_fatal("ALSU_TEST", "Virtual interface not exist");
            uvm_config_db#(alsu_config)::set(this, "*", "CFG", cfg);
        endfunction

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

            phase.raise_objection(this);
            #100 `uvm_info("run_phase", "Inside the ALSU test", UVM_MEDIUM);
            phase.drop_objection(this);
        endtask

    endclass
endpackage
```

4. Environment:

```
package alsu_env_pkg;
    import alsu_driver_pkg::*;
    import uvm_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
endpackage
```

5. Driver:

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

    class alsu_driver extends uvm_driver;
        `uvm_component_utils(alsu_driver);
        virtual alsu_if vif;
        alsu_config alsu_cfg_drv;

        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);
            uvm_config_db#(alsu_config)::get(this, "", "CFG", alsu_cfg_drv);
        endfunction

        function void connect_phase(uvm_phase phase);
            super.connect_phase(phase);
            vif = alsu_cfg_drv.alsu_config_vif;
        endfunction

        task run_phase(uvm_phase phase);
            super.run_phase(phase);
            vif.rst = 1;
            @(negedge vif.clk);
            vif.rst = 0;
            forever begin
                vif.A = $random;
                vif.B = $random;
                vif.cin = $random;
                vif.serial_in = $random;
                vif.red_op_A = $random;
                vif.red_op_B = $random;
                vif.opcode = $random;
                vif.bypass_A = $random;
                vif.bypass_B = $random;
                vif.direction = $random;
                @(negedge vif.clk);
            end
        endtask

    endclass
endpackage
```

6. Config:

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

    class alsu_config extends uvm_object;
        `uvm_object_utils(alsu_config);
        virtual alsu_if alsu_config_vif;
        function new(string name = "alsu_config");
            super.new(name);
        endfunction
    endclass
endpackage
```

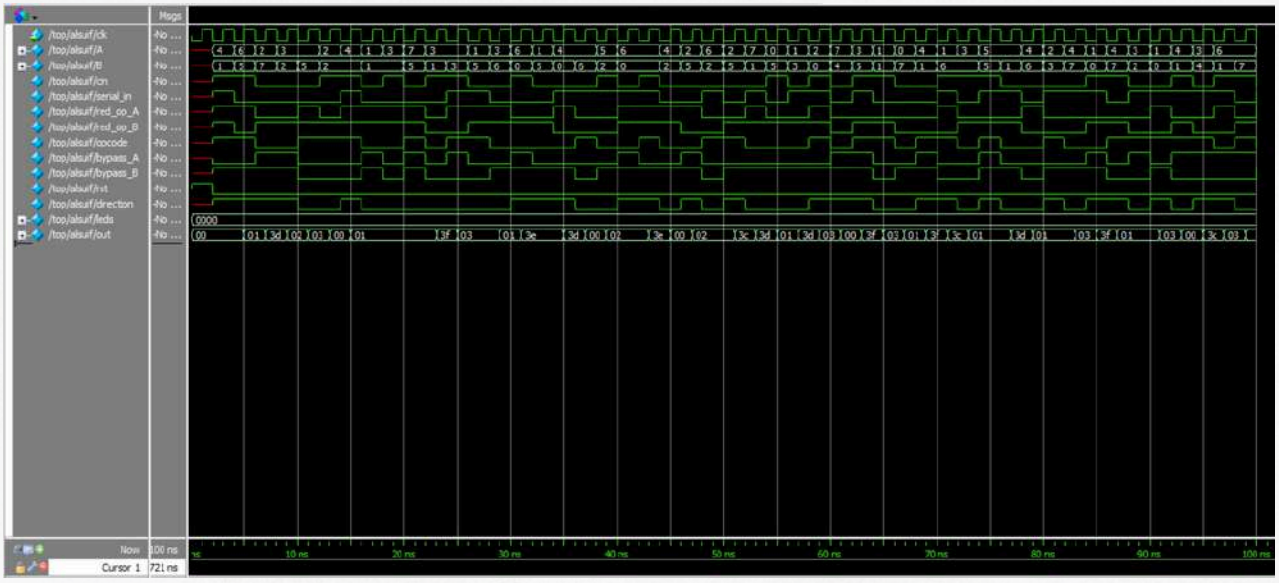
7. source files:

```
ALSU.sv
alsu_if.sv
alsu_config.sv
alsu_driver.sv
alsu_env.sv
alsu_test.sv
alsu_top.sv
```

8. Do file:

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

9. Questasim:




```
# ** 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
# 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
```