* 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. Do file:

vlib work

vlog -f src\_files.list

vsim -voptargs=+acc work.top -classdebug -uvmcontrol=all

add wave /top/alsuif/\*

run -all

* 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

        run\_test("alsu\_test");

    end

endmodule

* 1. 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. 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. 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

* 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

* 1. 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

* 1. 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

* 1. 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

* 1. 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

* 1. Simulation Snippet:

صورة تحتوي على عرض, نص, لقطة شاشة, برمجيات

قد يكون المحتوى المعد بواسطة الذكاء الاصطناعي غير صحيح.

* 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

* 1. 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

* 1. 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

* 1. 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

* 1. Simulation Snippet:

صورة تحتوي على لقطة شاشة, عرض, نص, برمجيات

قد يكون المحتوى المعد بواسطة الذكاء الاصطناعي غير صحيح.