UVM

Assignment5

By: Mohamed Sayed Mohamed Soliman

Part #1

1- CODE Design

```
3   module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction, leds, out);
4   parameter INPUT_PRIORITY = "A";
5   parameter FULL_ADDER = "ON";
6   input clk, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
7   input [2:0] opcode;
8   input signed [2:0] A, B;    // first bug --> we must put it signed
9   output reg [15:0] leds;
10  output reg signed [5:0] out;  // second bug --> we must put it signed
11  reg cin_reg, red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
12  reg [2:0] opcode_reg;
13  reg signed  [2:0]  A_reg, B_reg;
14  wire invalid_red_op, invalid_opcode, invalid;
15  //Invalid handling
16  assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);
17  assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
18  assign invalid = invalid_red_op | invalid_opcode;
19  //Registering input signals
20  always @(posedge clk or posedge rst) begin
21    if(rst) begin
22      cin_reg <= 0;
23      red_op_B_reg <= 0;
24      red_op_A_reg <= 0;
25      bypass_B_reg <= 0;
26      bypass_A_reg <= 0;
27      direction_reg <= 0;
28      serial_in_reg <= 0;
29      opcode_reg <= 0;
30      A_reg <= 0;
31      B_reg <= 0;
32        end else begin
33            cin_reg <= cin;
34            red_op_B_reg <= red_op_B;
35            red_op_A_reg <= red_op_A;
36            bypass_B_reg <= bypass_B;
37            bypass_A_reg <= bypass_A;
38            direction_reg <= direction;
39            serial_in_reg <= serial_in;
40            opcode_reg <= opcode;
41            A_reg <= A;
42            B_reg <= B;
43        end
44    end
45    //leds output blinking
46    always @(posedge clk or posedge rst) begin
47      if(rst) begin
48          leds <= 0;
49      end else begin
50          if (invalid)
51            leds <= ~leds;
52          else
53            leds <= 0;
54      end
55    end
```

```
always @(posedge clk or posedge rst) begin
  if(rst) begin
    out <= 0;
  end
  else begin
    if (invalid)
      out <= 0;
    else if (bypass_A_reg && bypass_B_reg)
      out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
    else if (bypass_A_reg)
      out <= A_reg;
    else if (bypass_B_reg)
      out <= B_reg;
    else begin
      case (opcode_reg) //  third bug is to used the opcode_reg  not the opcode
        3'h0: begin
          if (red_op_A_reg && red_op_B_reg)
            out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;  // third bug is to replace AND with OR
          else if (red_op_A_reg)
            out <= |A_reg;
          else if (red_op_B_reg)
            out <= |B_reg;
          else
            out <= A_reg | B_reg;
        end
        3'h1: begin
          if (red_op_A_reg && red_op_B_reg)
            out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;  // fourth bug is to replace OR with XOR
          else if (red_op_A_reg)
            out <= ^A_reg;
          else if (red_op_B_reg)
            out <= ^B_reg;
          else
            out <= A_reg ^ B_reg;
        end
        3'h2:begin
              if(FULL_ADDER == "ON")  // fifth bug to add Cin operation in case of full adde
                  out <= A_reg + B_reg + cin_reg ;
              else
                  out <= A_reg + B_reg ;
              end
        3'h3: out <= A_reg * B_reg;
        3'h4: begin
          if (direction_reg)
            out <= {out[4:0], serial_in_reg};
          else
            out <= {serial_in_reg, out[5:1]};
        end
        3'h5: begin
          if (direction_reg)
            out <= {out[4:0], out[5]};
          else
            out <= {out[0], out[5:1]};
        end
       default : out <= 0 ;
        endcase
    end
  end
end

endmodule
```

2- Interface

```
interface ALSU_if (clk);
  input clk ;
  logic  rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
  logic [2:0] opcode;
  logic signed [2:0] A, B;    // first bug --> we must put it signed
  logic  [15:0] leds;
  logic  signed [5:0] out;
endinterface
```

3- top module

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

module ALSU_top();
bit clk ;

initial
  begin
    forever
    #1 clk = ~clk ;
  end
  ALSU_if ALSUif (clk) ;
  ALSU DUT (clk , ALSUif.rst , ALSUif.cin , ALSUif.red_op_A , ALSUif.red_op_B , ALSUif.bypass_A , ALSUif.bypass_B , ALSUif.direction , ALSUif.serial_in , ALSUif.opcode ,

  initial
    begin
        run_test("ALSU_test");
    end
endmodule
```

4- alsu_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 );
   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:run_phase

  endclass
endpackage
```

5- alsu_env

```
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

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

6- Transcript

```
#
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(215) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.2
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(217) @ 0: reporter [Questa UVM]  questa_uvm::init(+struct)
# UVM_INFO @ 0: reporter [RNTST] Running test ALSU_test...
# UVM_INFO C:/Users/CS/Downloads/Karem Wasem Diploma/session5_Assignment/labl_ass5/ALSU_test.sv(21) @ 100: uvm_test_top [run_phase] Inside the ALSU tes
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1268) @ 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_10.4c/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 100 ns  Iteration: 54  Instance: /ALSU_top
# 1
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_10.4c/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```

Part #2

1- CODE Design

```
3   module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction, leds, out);
4   parameter INPUT_PRIORITY = "A";
5   parameter FULL_ADDER = "ON";
6   input clk, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
7   input [2:0] opcode;
8   input signed [2:0] A, B;    // first bug --> we must put it signed
9   output reg [15:0] leds;
10  output reg signed [5:0] out;  // second bug --> we must put it signed
11  reg cin_reg, red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
12  reg [2:0] opcode_reg;
13  reg signed  [2:0]  A_reg, B_reg;
14  wire invalid_red_op, invalid_opcode, invalid;
15  //Invalid handling
16  assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);
17  assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
18  assign invalid = invalid_red_op | invalid_opcode;
19  //Registering input signals
20  always @(posedge clk or posedge rst) begin
21    if(rst) begin
22      cin_reg <= 0;
23      red_op_B_reg <= 0;
24      red_op_A_reg <= 0;
25      bypass_B_reg <= 0;
26      bypass_A_reg <= 0;
27      direction_reg <= 0;
28      serial_in_reg <= 0;
29      opcode_reg <= 0;
30      A_reg <= 0;
31      B_reg <= 0;
32        end else begin
33            cin_reg <= cin;
34            red_op_B_reg <= red_op_B;
35            red_op_A_reg <= red_op_A;
36            bypass_B_reg <= bypass_B;
37            bypass_A_reg <= bypass_A;
38            direction_reg <= direction;
39            serial_in_reg <= serial_in;
40            opcode_reg <= opcode;
41            A_reg <= A;
42            B_reg <= B;
43        end
44    end
45    //leds output blinking
46    always @(posedge clk or posedge rst) begin
47      if(rst) begin
48          leds <= 0;
49      end else begin
50          if (invalid)
51            leds <= ~leds;
52          else
53            leds <= 0;
54      end
55    end
```

```
always @(posedge clk or posedge rst) begin
  if(rst) begin
    out <= 0;
  end
  else begin
    if (invalid)
      out <= 0;
    else if (bypass_A_reg && bypass_B_reg)
      out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
    else if (bypass_A_reg)
      out <= A_reg;
    else if (bypass_B_reg)
      out <= B_reg;
    else begin
      case (opcode_reg) //  third bug is to used the opcode_reg  not the opcode
        3'h0: begin
          if (red_op_A_reg && red_op_B_reg)
            out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;  // third bug is to replace AND with OR
          else if (red_op_A_reg)
            out <= |A_reg;
          else if (red_op_B_reg)
            out <= |B_reg;
          else
            out <= A_reg | B_reg;
        end
        3'h1: begin
          if (red_op_A_reg && red_op_B_reg)
            out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;  // fourth bug is to replace OR with XOR
          else if (red_op_A_reg)
            out <= ^A_reg;
          else if (red_op_B_reg)
            out <= ^B_reg;
          else
            out <= A_reg ^ B_reg;
        end
        3'h2:begin
              if(FULL_ADDER == "ON")  // fifth bug to add Cin operation in case of full adde
                  out <= A_reg + B_reg + cin_reg ;
              else
                  out <= A_reg + B_reg ;
              end
        3'h3: out <= A_reg * B_reg;
        3'h4: begin
          if (direction_reg)
            out <= {out[4:0], serial_in_reg};
          else
            out <= {serial_in_reg, out[5:1]};
        end
        3'h5: begin
          if (direction_reg)
            out <= {out[4:0], out[5]};
          else
            out <= {out[0], out[5:1]};
        end
       default : out <= 0 ;
        endcase
    end
  end
end

endmodule
```

2- Interface

```
interface ALSU_if (clk);
  input clk ;
  logic  rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
  logic [2:0] opcode;
  logic signed [2:0] A, B;    // first bug --> we must put it signed
  logic  [15:0] leds;
  logic  signed [5:0] out;
endinterface
```

3- top module

```
import ALSU_test_pkg::*;
import uvm_pkg::* ;
`include "uvm_macros.svh"
module ALSU_top();
bit clk ;

initial
  begin
    forever
    #1 clk = ~clk ;
  end
  ALSU_if ALSUif (clk) ;
  ALSU DUT (clk , ALSUif.rst , ALSUif.cin , ALSUif.red_op_A , ALSUif.red_op_B , ALSUif.bypass_A , ALSUif.bypass_B , ALSUif.direction , ALSUif.serial_in , ALSUif.opcode ,

  initial
    begin
        uvm_config_db#(virtual ALSU_if)::set(null , "uvm_test_top" , "ALSUif" , ALSUif );
        run_test("ALSU_test");
    end
endmodule
```

4- alsu_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 ;
    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);
      env = ALSU_env::type_id::create("env",this );
      if(!uvm_config_db#(virtual ALSU_if)::get (this , "" , "ALSUif" , alsu_test_vif ))
        `uvm_fatal("run_phase" , "test - unable to get the virtual interface ") ;
      uvm_config_db#(virtual ALSU_if)::set (this , "*" , "CFG" , alsu_test_vif ) ;
    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:run_phase

  endclass
endpackage
```

5- alsu_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  ALSUdriver ;
      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);
         ALSUdriver = ALSU_driver::type_id::create("ALSUdriver",this );
      endfunction
   endclass
endpackage
```

6- alsu_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);
            if(!uvm_config_db#(virtual ALSU_if)::get (this , "" , "CFG" , alsu_driver_vif ) )
              `uvm_fatal("build_phase" , "driver - unable to get the virtual interface ") ;
        endfunction
        function void connect_phase(uvm_phase phase) ;
            super.build_phase(phase);
        endfunction
        task run_phase(uvm_phase phase) ;
            super.run_phase(phase);
            alsu_driver_vif.rst = 1 ;
            alsu_driver_vif.red_op_A = 0 ;
            alsu_driver_vif.red_op_B = 0 ;
            alsu_driver_vif.bypass_A = 0 ;
            alsu_driver_vif.bypass_B = 0 ;
            alsu_driver_vif.direction = 0 ;
            alsu_driver_vif.serial_in = 0 ;
            alsu_driver_vif.opcode = 0 ;
            alsu_driver_vif.A = 0 ;
            alsu_driver_vif.B = 0 ;
            alsu_driver_vif.cin = 0 ;
            @(negedge alsu_driver_vif.clk ) ;
            alsu_driver_vif.rst = 0 ;

                forever
                begin
                    @(negedge alsu_driver_vif.clk ) ;
                    alsu_driver_vif.serial_in = $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 ;
                    alsu_driver_vif.opcode = $random ;
                    alsu_driver_vif.A = $random ;
                    alsu_driver_vif.B = $random ;
                    alsu_driver_vif.cin = $random ;
                end
            endtask
        endclass
    endpackage
```

7- Transcript

```
#
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(215) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.2
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(217) @ 0: reporter [Questa UVM]  questa_uvm::init(+struct)
# UVM_INFO @ 0: reporter [RNTST] Running test ALSU_test...
# UVM_WARNING @ 0: uvm_test_top.env.ALSUdriver [UVM_DEPRECATED] build()/build_phase() has been called explicitly, outside of the phasing system. This usage of build is deprecated and may lead to unexpected behavio
# UVM_INFO C:/Users/CS/Downloads/Karem Wasem Diploma/session5_Assignment/lab2_ass5/ALSU_test.sv(22) @ 100: uvm_test_top [run_phase] Inside the ALSU test
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1268) @ 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 :    1
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [UVM_DEPRECATED]     1
# [run_phase]     1
# ** Note: $finish    : C:/questasim64_10.4c/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 100 ns  Iteration: 54  Instance: /ALSU_top
# 1
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_10.4c/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```

Part #3

1- CODE Design

```
3   module ALSU(A, B, cin, serial_in, red_op_A, red_op_B, opcode, bypass_A, bypass_B, clk, rst, direction, leds, out);
4   parameter INPUT_PRIORITY = "A";
5   parameter FULL_ADDER = "ON";
6   input clk, rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
7   input [2:0] opcode;
8   input signed [2:0] A, B;    // first bug --> we must put it signed
9   output reg [15:0] leds;
10  output reg signed [5:0] out;  // second bug --> we must put it signed
11  reg cin_reg, red_op_A_reg, red_op_B_reg, bypass_A_reg, bypass_B_reg, direction_reg, serial_in_reg;
12  reg [2:0] opcode_reg;
13  reg signed  [2:0]  A_reg, B_reg;
14  wire invalid_red_op, invalid_opcode, invalid;
15  //Invalid handling
16  assign invalid_red_op = (red_op_A_reg | red_op_B_reg) & (opcode_reg[1] | opcode_reg[2]);
17  assign invalid_opcode = opcode_reg[1] & opcode_reg[2];
18  assign invalid = invalid_red_op | invalid_opcode;
19  //Registering input signals
20  always @(posedge clk or posedge rst) begin
21    if(rst) begin
22      cin_reg <= 0;
23      red_op_B_reg <= 0;
24      red_op_A_reg <= 0;
25      bypass_B_reg <= 0;
26      bypass_A_reg <= 0;
27      direction_reg <= 0;
28      serial_in_reg <= 0;
29      opcode_reg <= 0;
30      A_reg <= 0;
31      B_reg <= 0;
32        end else begin
33            cin_reg <= cin;
34            red_op_B_reg <= red_op_B;
35            red_op_A_reg <= red_op_A;
36            bypass_B_reg <= bypass_B;
37            bypass_A_reg <= bypass_A;
38            direction_reg <= direction;
39            serial_in_reg <= serial_in;
40            opcode_reg <= opcode;
41            A_reg <= A;
42            B_reg <= B;
43        end
44    end
45    //leds output blinking
46    always @(posedge clk or posedge rst) begin
47      if(rst) begin
48          leds <= 0;
49      end else begin
50          if (invalid)
51            leds <= ~leds;
52          else
53            leds <= 0;
54      end
55    end
```

```
always @(posedge clk or posedge rst) begin
  if(rst) begin
    out <= 0;
  end
  else begin
    if (invalid)
      out <= 0;
    else if (bypass_A_reg && bypass_B_reg)
      out <= (INPUT_PRIORITY == "A")? A_reg: B_reg;
    else if (bypass_A_reg)
      out <= A_reg;
    else if (bypass_B_reg)
      out <= B_reg;
    else begin
      case (opcode_reg) //  third bug is to used the opcode_reg  not the opcode
        3'h0: begin
          if (red_op_A_reg && red_op_B_reg)
            out <= (INPUT_PRIORITY == "A")? |A_reg: |B_reg;  // third bug is to replace AND with OR
          else if (red_op_A_reg)
            out <= |A_reg;
          else if (red_op_B_reg)
            out <= |B_reg;
          else
            out <= A_reg | B_reg;
        end
        3'h1: begin
          if (red_op_A_reg && red_op_B_reg)
            out <= (INPUT_PRIORITY == "A")? ^A_reg: ^B_reg;  // fourth bug is to replace OR with XOR
          else if (red_op_A_reg)
            out <= ^A_reg;
          else if (red_op_B_reg)
            out <= ^B_reg;
          else
            out <= A_reg ^ B_reg;
        end
        3'h2:begin
              if(FULL_ADDER == "ON")  // fifth bug to add Cin operation in case of full adde
                  out <= A_reg + B_reg + cin_reg ;
              else
                  out <= A_reg + B_reg ;
              end
        3'h3: out <= A_reg * B_reg;
        3'h4: begin
          if (direction_reg)
            out <= {out[4:0], serial_in_reg};
          else
            out <= {serial_in_reg, out[5:1]};
        end
        3'h5: begin
          if (direction_reg)
            out <= {out[4:0], out[5]};
          else
            out <= {out[0], out[5:1]};
        end
       default : out <= 0 ;
        endcase
    end
  end
end

endmodule
```

2- Interface

```
interface ALSU_if (clk);
  input clk ;
  logic  rst, cin, red_op_A, red_op_B, bypass_A, bypass_B, direction, serial_in;
  logic [2:0] opcode;
  logic signed [2:0] A, B;    // first bug --> we must put it signed
  logic  [15:0] leds;
  logic  signed [5:0] out;
endinterface
```

3- top module

```
import ALSU_test_pkg::*;
import uvm_pkg::* ;
`include "uvm_macros.svh"
module ALSU_top();
bit clk ;

initial
  begin
    forever
    #1 clk = ~clk ;
  end
  ALSU_if ALSUif (clk) ;
  ALSU DUT (clk , ALSUif.rst , ALSUif.cin , ALSUif.red_op_A , ALSUif.red_op_B , ALSUif.bypass_A , ALSUif.bypass_B , ALSUif.direction , ALSUif.serial_in , ALSUif.opcode ,

  initial
    begin
        uvm_config_db#(virtual ALSU_if)::set(null , "uvm_test_top" , "ALSUif" , ALSUif );
        run_test("ALSU_test");
    end
endmodule
```

4- alsu_test

```
> Users > CS > Downloads > Karem Wasem Diploma > session5_Assignment > lab3_ass5 > = ALSU_test.sv
1   package ALSU_test_pkg ;
2     import ALSU_env_pkg::*;
3     import ALSU_config_pkg::*;
4     import uvm_pkg::* ;
5     `include "uvm_macros.svh"
6     class ALSU_test extends uvm_test;
7       `uvm_component_utils(ALSU_test)
8       ALSU_env  env ;
9       ALSU_config  alsu_config_obj_test ;
10      function new(string name = "ALSU_test" , uvm_component parent = null );
11        super.new(name ,parent ) ;
12      endfunction
13      function void build_phase(uvm_phase phase) ;
14       super.build_phase(phase);
15       env = ALSU_env::type_id::create("env",this );
16       alsu_config_obj_test = ALSU_config::type_id::create("alsu_config_obj_test");
17       if(!uvm_config_db#(virtual ALSU_if)::get (this , "" , "ALSUif" , alsu_config_obj_test.alsu_config_vif ))
18         `uvm_fatal("run_phase" , "test - unable to get the virtual interface ") ;
19
20       uvm_config_db#(ALSU_config)::set (this , "*" , "CFG" , alsu_config_obj_test ) ;
21
22     endfunction
23
24     task run_phase(uvm_phase phase) ;
25      super.run_phase(phase);
26      phase.raise_objection(this);
27      #100 ; `uvm_info("run_phase", "Inside the ALSU test" , UVM_MEDIUM)
28      phase.drop_objection(this);
29     endtask:run_phase
30
31    endclass
32  endpackage
```

5- alsu_env

```
1   package ALSU_env_pkg ;
2
3     import ALSU_driver_pkg::*;
4      import uvm_pkg::* ;
5      `include "uvm_macros.svh"
6
7
8      class ALSU_env extends uvm_env;
9         `uvm_component_utils(ALSU_env)
10
11        ALSU_driver  ALSUdriver ;
12        function new(string name = "ALSU_env" , uvm_component parent = null );
13           super.new(name ,parent ) ;
14        endfunction
15
16        function void build_phase(uvm_phase phase) ;
17           super.build_phase(phase);
18           ALSUdriver = ALSU_driver::type_id::create("ALSUdriver",this );
19        endfunction
20     endclass
21  endpackage
```

6- alsu_driver

```
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  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)::get (this , "" , "CFG" , alsu_config_obj_driver ) )
                `uvm_fatal("build_phase" , "driver - unable to get the virtual interface ") ;
        endfunction
        function void connect_phase(uvm_phase phase) ;
            super.build_phase(phase);
            alsu_driver_vif = alsu_config_obj_driver.alsu_config_vif ;
        endfunction

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

7- alsu_config_obj

```
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
```

8- Transcript

```
#       (specify +UVM_NO_RELNOTES to turn off this notice)
#
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(215) @ 0: reporter [Questa UVM] QUESTA_UVM-1.2.2
# UVM_INFO verilog_src/questa_uvm_pkg-1.2/src/questa_uvm_pkg.sv(217) @ 0: reporter [Questa UVM]  questa_uvm::init(+struct)
# UVM_INFO @ 0: reporter [RNTST] Running test ALSU_test...
# UVM_WARNING @ 0: uvm_test_top.env.ALSUdriver [UVM_DEPRECATED] build()/build_phase() has been called explicitly, outside of the phasing system. This usage of build is deprecated and may lead to unexpected behavior.
# UVM_INFO C:/Users/CS/Downloads/Karem Wasem Diploma/session5_Assignment/lab3_ass5/ALSU_test.sv(32) @ 100: uvm_test_top [run_phase] Inside the ALSU test
# UVM_INFO verilog_src/uvm-1.1d/src/base/uvm_objection.svh(1268) @ 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 :    1
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [Questa UVM]     2
# [RNTST]     1
# [TEST_DONE]     1
# [UVM_DEPRECATED]     1
# [run_phase]     1
# ** Note: $finish    : C:/questasim64_10.4c/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh(430)
#    Time: 100 ns  Iteration: 54  Instance: /ALSU_top
# 1
# Break in Task uvm_pkg/uvm_root::run_test at C:/questasim64_10.4c/win64/../verilog_src/uvm-1.1d/src/base/uvm_root.svh line 430
```