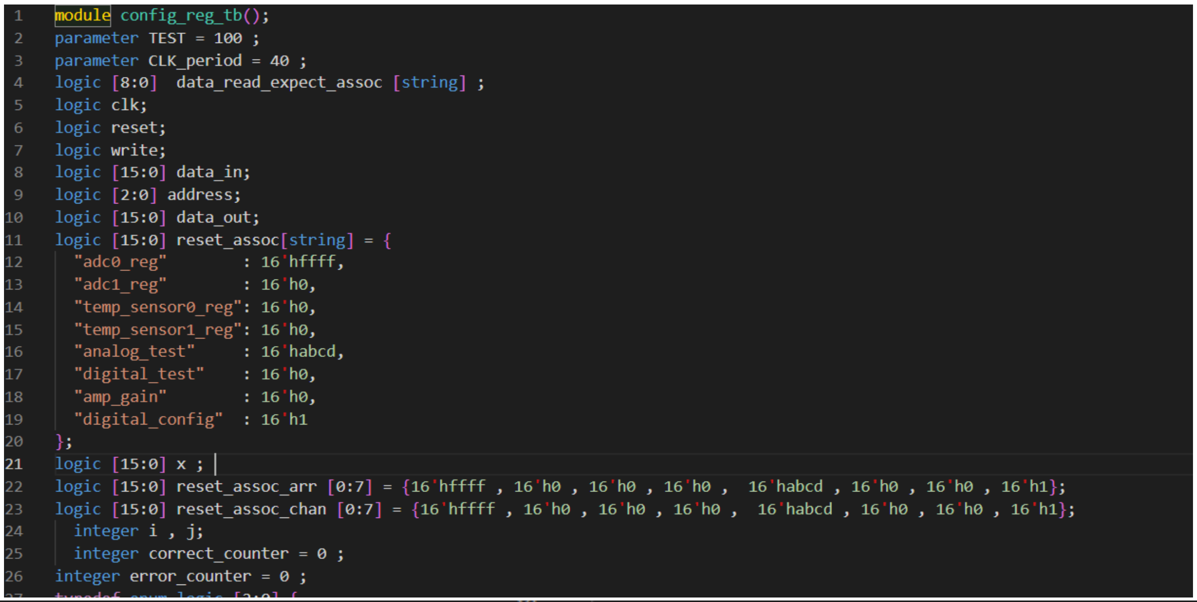
Assignment4_extra

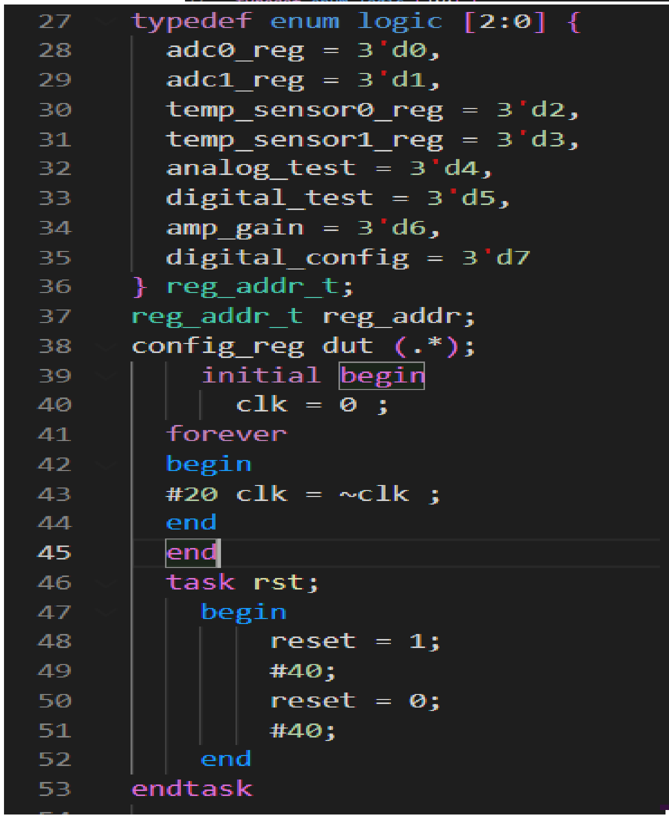
By: Mohamed Sayed Mohamed Soliman

Question 1

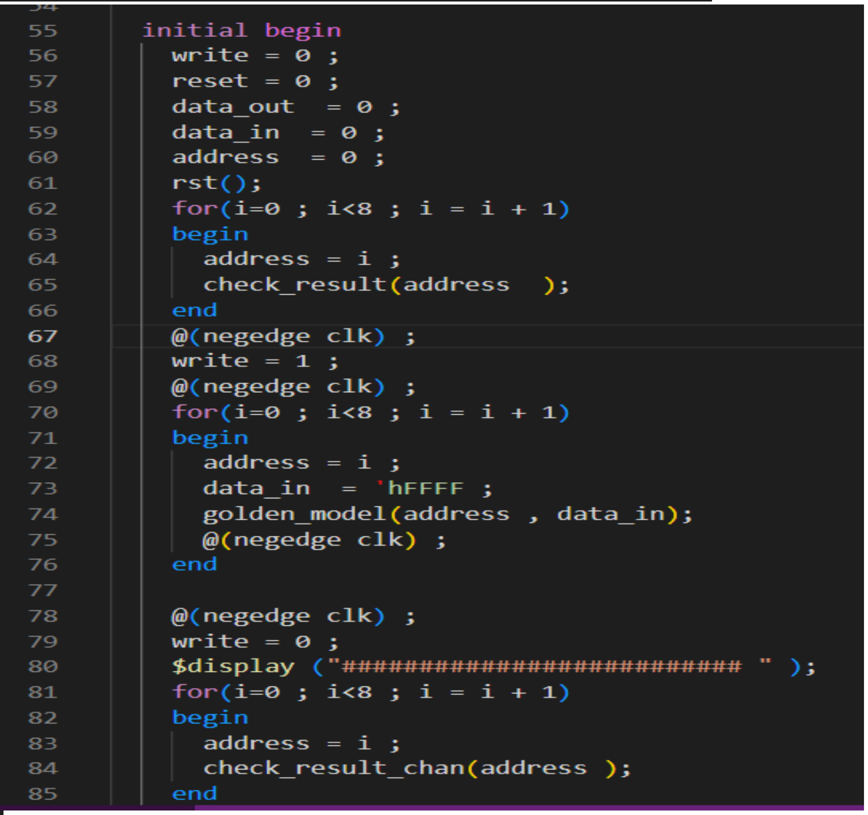
1. ALSU Testbench



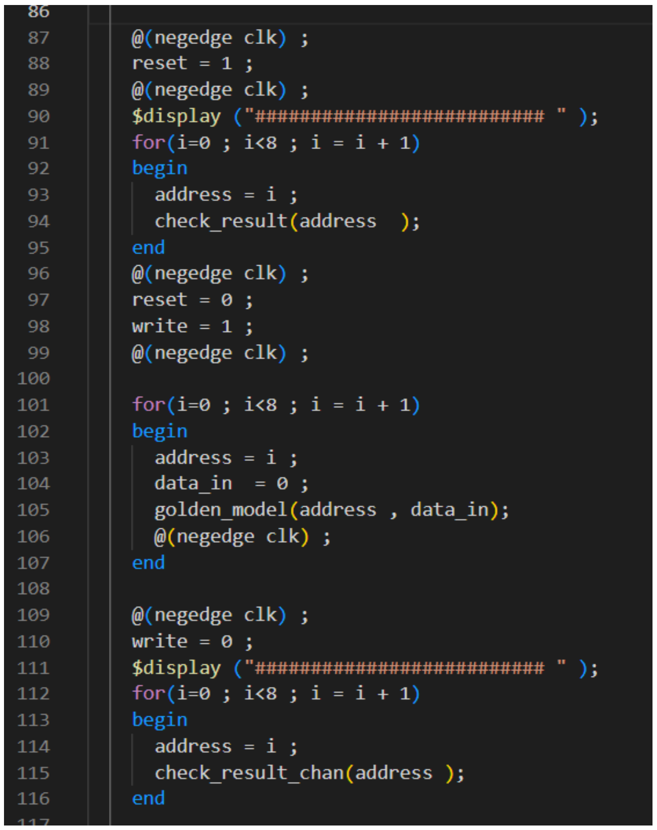
```
module config_reg_tb();
parameter TEST = 100 ;
parameter CLK_period = 40 ;
logic [8:0]  data_read_expect_assoc [string] ;
logic clk;
logic reset;
logic write;
logic [15:0] data_in;
logic [2:0] address;
logic [15:0] data_out;
logic [15:0] reset_assoc[string] = {
  "adc0_reg"        : 16'hffff,
  "adc1_reg"        : 16'h0,
  "temp_sensor0_reg": 16'h0,
  "temp_sensor1_reg": 16'h0,
  "analog_test"     : 16'habcd,
  "digital_test"    : 16'h0,
  "amp_gain"        : 16'h0,
  "digital_config"  : 16'h1
};
logic [15:0] x ;
logic [15:0] reset_assoc_arr [0:7] = {16'hffff , 16'h0 , 16'h0 , 16'h0 ,  16'habcd , 16'h0 , 16'h0 , 16'h1};
logic [15:0] reset_assoc_chan [0:7] = {16'hffff , 16'h0 , 16'h0 , 16'h0 ,  16'habcd , 16'h0 , 16'h0 , 16'h1};
  integer i , j;
  integer correct_counter = 0 ;
integer error_counter = 0 ;
```



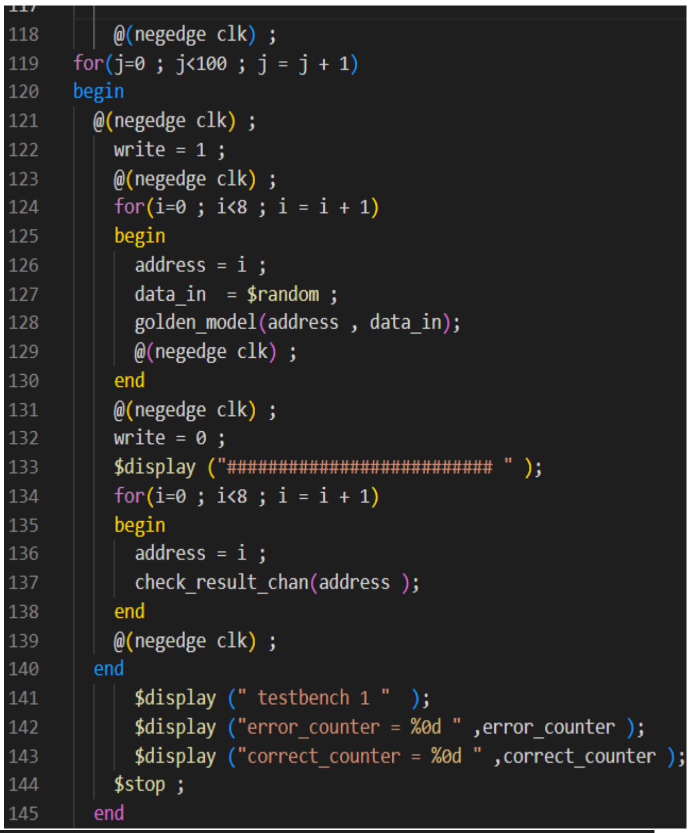
```
typedef enum logic [2:0] {
  adc0_reg = 3'd0,
  adc1_reg = 3'd1,
  temp_sensor0_reg = 3'd2,
  temp_sensor1_reg = 3'd3,
  analog_test = 3'd4,
  digital_test = 3'd5,
  amp_gain = 3'd6,
  digital_config = 3'd7
} reg_addr_t;
reg_addr_t reg_addr;
config_reg dut (.*);
  initial begin
    clk = 0 ;
  forever
  begin
  #20 clk = ~clk ;
  end
  end
  task rst;
    begin
      reset = 1;
      #40;
      reset = 0;
      #40;
    end
endtask
```



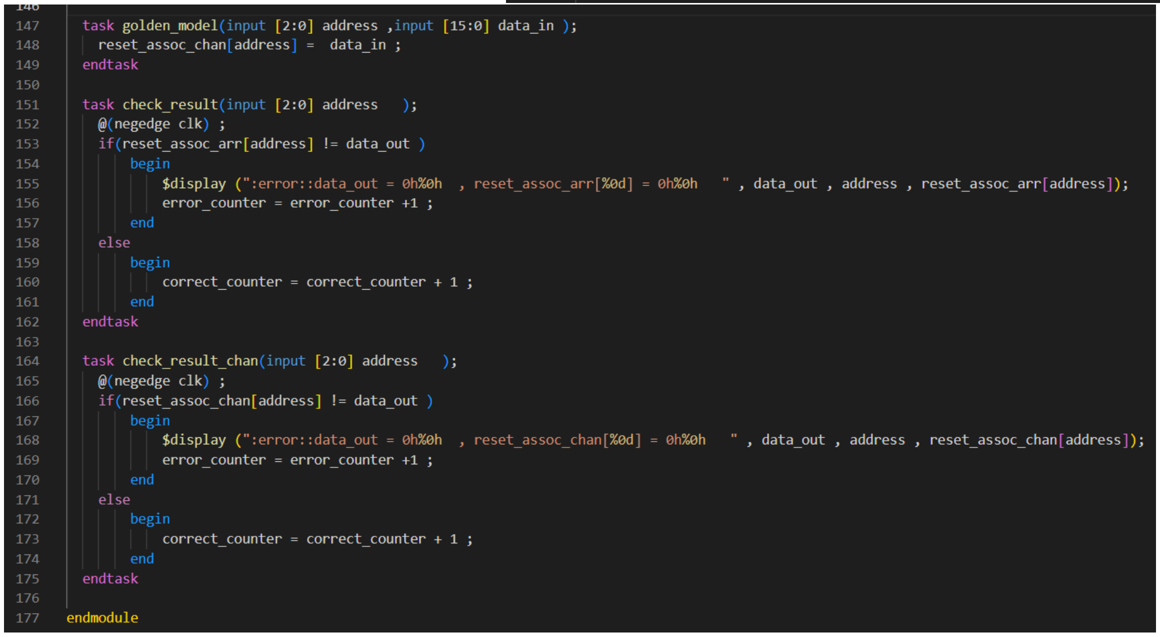
```
initial begin
  write = 0 ;
  reset = 0 ;
  data_out  = 0 ;
  data_in  = 0 ;
  address  = 0 ;
  rst();
  for(i=0 ; i<8 ; i = i + 1)
  begin
    address = i ;
    check_result(address   );
  end
  @(negedge clk) ;
  write = 1 ;
  @(negedge clk) ;
  for(i=0 ; i<8 ; i = i + 1)
  begin
    address = i ;
    data_in  = 'hFFFF ;
    golden_model(address , data_in);
    @(negedge clk) ;
  end

  @(negedge clk) ;
  write = 0 ;
  $display ("########################## " );
  for(i=0 ; i<8 ; i = i + 1)
  begin
    address = i ;
    check_result_chan(address );
  end
```



```verilog

    @(negedge clk) ;
    reset = 1 ;
    @(negedge clk) ;
    $display ("########################## " );
    for(i=0 ; i<8 ; i = i + 1)
    begin
      address = i ;
      check_result(address  );
    end
    @(negedge clk) ;
    reset = 0 ;
    write = 1 ;
    @(negedge clk) ;

    for(i=0 ; i<8 ; i = i + 1)
    begin
      address = i ;
      data_in   = 0 ;
      golden_model(address , data_in);
      @(negedge clk) ;
    end

    @(negedge clk) ;
    write = 0 ;
    $display ("########################## " );
    for(i=0 ; i<8 ; i = i + 1)
    begin
      address = i ;
      check_result_chan(address );
    end
```

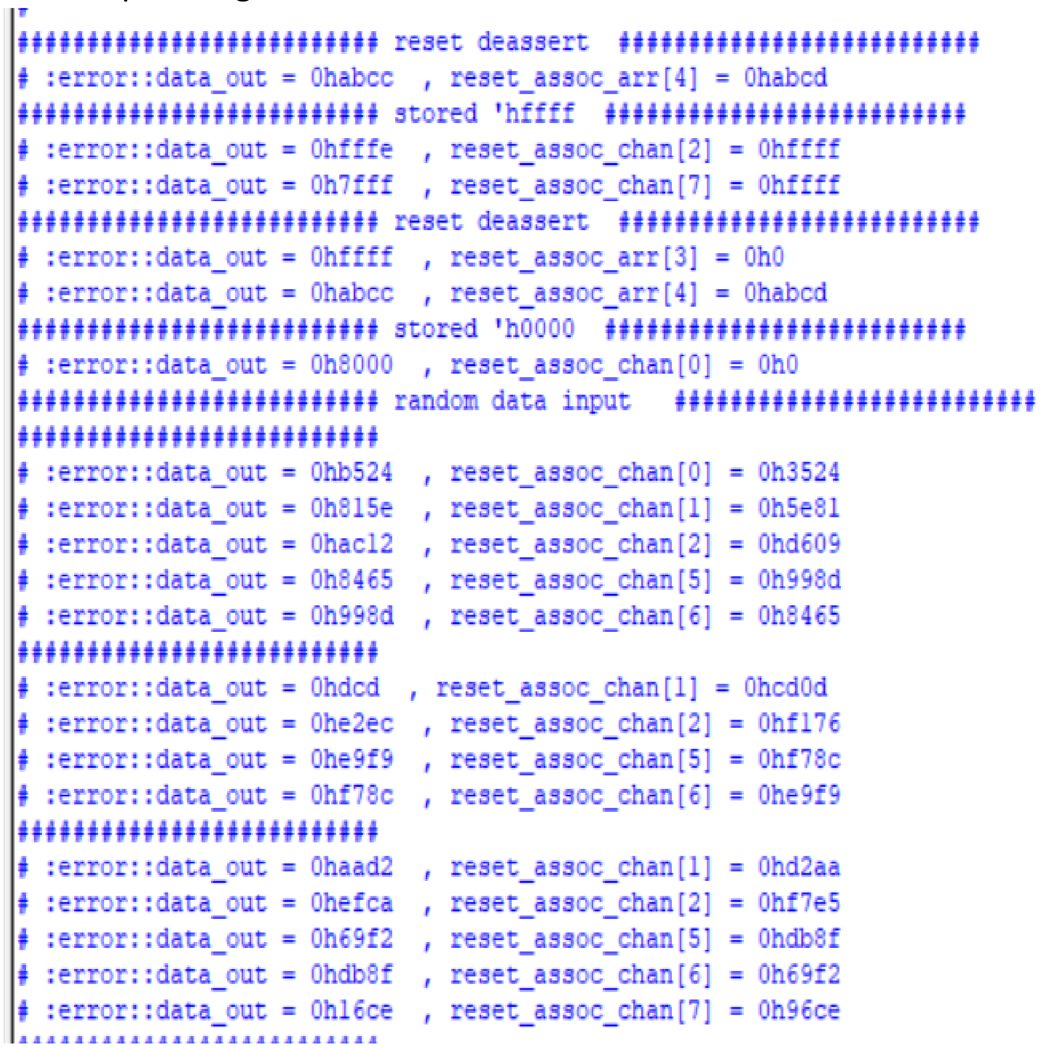


```verilog
      @(negedge clk) ;
    for(j=0 ; j<100 ; j = j + 1)
    begin
      @(negedge clk) ;
        write = 1 ;
        @(negedge clk) ;
        for(i=0 ; i<8 ; i = i + 1)
        begin
          address = i ;
          data_in  = $random ;
          golden_model(address , data_in);
          @(negedge clk) ;
        end
        @(negedge clk) ;
        write = 0 ;
        $display ("########################## " );
        for(i=0 ; i<8 ; i = i + 1)
        begin
          address = i ;
          check_result_chan(address );
        end
        @(negedge clk) ;
      end
          $display (" testbench 1 "  );
          $display ("error_counter = %0d " ,error_counter );
          $display ("correct_counter = %0d " ,correct_counter );
        $stop ;
      end
```



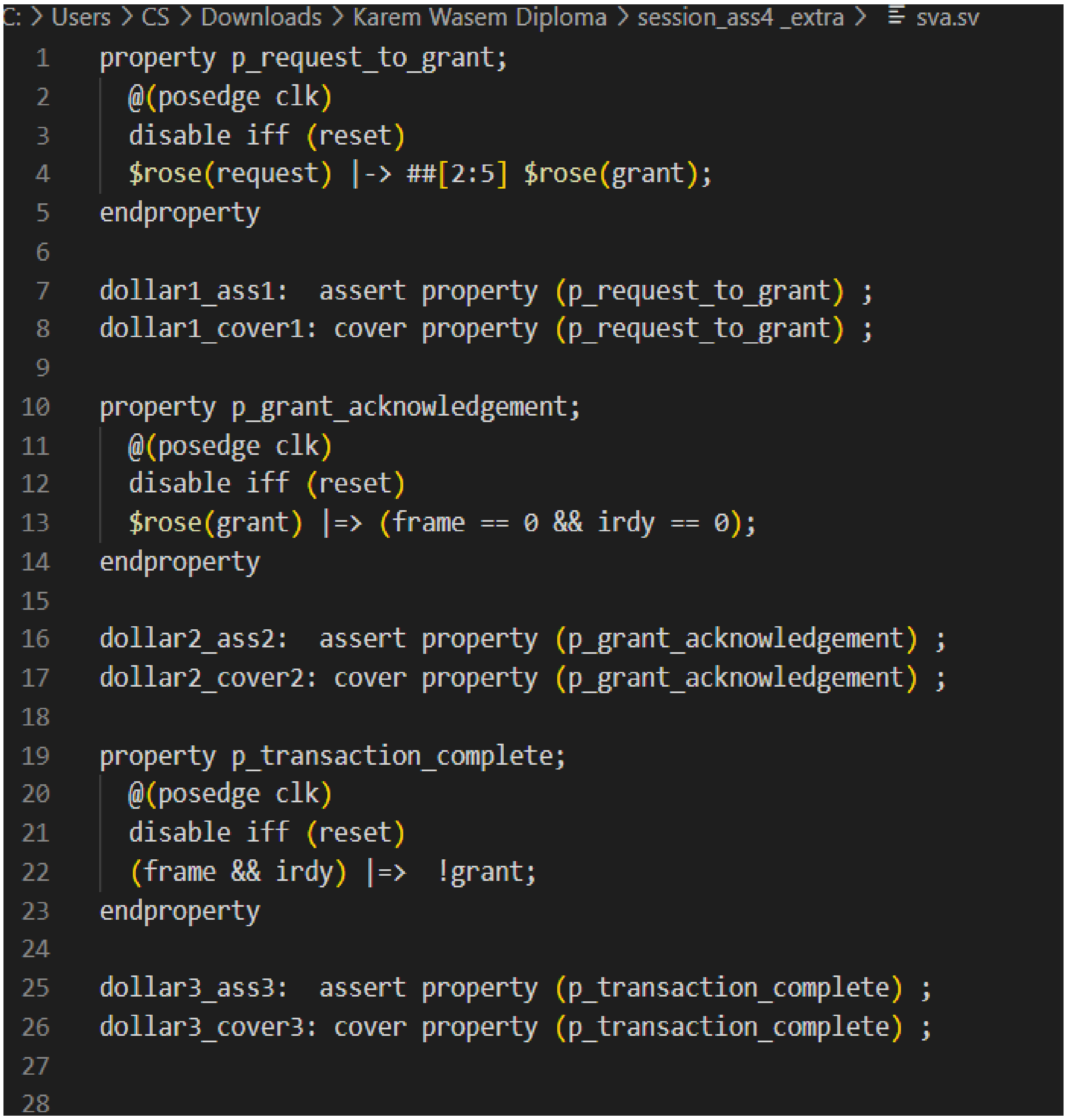
```verilog

  task golden_model(input [2:0] address ,input [15:0] data_in );
    reset_assoc_chan[address] =  data_in ;
  endtask

  task check_result(input [2:0] address   );
    @(negedge clk) ;
    if(reset_assoc_arr[address] != data_out )
        begin
          $display (":error::data_out = 0h%0h  , reset_assoc_arr[%0d] = 0h%0h   " , data_out , address , reset_assoc_arr[address]);
          error_counter = error_counter +1 ;
        end
    else
        begin
          correct_counter = correct_counter + 1 ;
        end
  endtask

  task check_result_chan(input [2:0] address   );
    @(negedge clk) ;
    if(reset_assoc_chan[address] != data_out )
        begin
          $display (":error::data_out = 0h%0h  , reset_assoc_chan[%0d] = 0h%0h   " , data_out , address , reset_assoc_chan[address]);
          error_counter = error_counter +1 ;
        end
    else
        begin
          correct_counter = correct_counter + 1 ;
        end
  endtask

endmodule
```

2. Transcript & bugs



```
#
######################### reset deassert  ##########################
# :error::data_out = 0habcc  , reset_assoc_arr[4] = 0habcd
######################### stored 'hffff  ##########################
# :error::data_out = 0hfffe  , reset_assoc_chan[2] = 0hffff
# :error::data_out = 0h7fff  , reset_assoc_chan[7] = 0hffff
######################### reset deassert  ##########################
# :error::data_out = 0hffff  , reset_assoc_arr[3] = 0h0
# :error::data_out = 0habcc  , reset_assoc_arr[4] = 0habcd
######################### stored 'h0000  ##########################
# :error::data_out = 0h8000  , reset_assoc_chan[0] = 0h0
######################### random data input   ##########################
##########################
# :error::data_out = 0hb524  , reset_assoc_chan[0] = 0h3524
# :error::data_out = 0h815e  , reset_assoc_chan[1] = 0h5e81
# :error::data_out = 0hac12  , reset_assoc_chan[2] = 0hd609
# :error::data_out = 0h8465  , reset_assoc_chan[5] = 0h998d
# :error::data_out = 0h998d  , reset_assoc_chan[6] = 0h8465
##########################
# :error::data_out = 0hdcd  , reset_assoc_chan[1] = 0hcd0d
# :error::data_out = 0he2ec  , reset_assoc_chan[2] = 0hf176
# :error::data_out = 0he9f9  , reset_assoc_chan[5] = 0hf78c
# :error::data_out = 0hf78c  , reset_assoc_chan[6] = 0he9f9
##########################
# :error::data_out = 0haad2  , reset_assoc_chan[1] = 0hd2aa
# :error::data_out = 0hefca  , reset_assoc_chan[2] = 0hf7e5
# :error::data_out = 0h69f2  , reset_assoc_chan[5] = 0hdb8f
# :error::data_out = 0hdb8f  , reset_assoc_chan[6] = 0h69f2
# :error::data_out = 0h16ce  , reset_assoc_chan[7] = 0h96ce
##########################
```

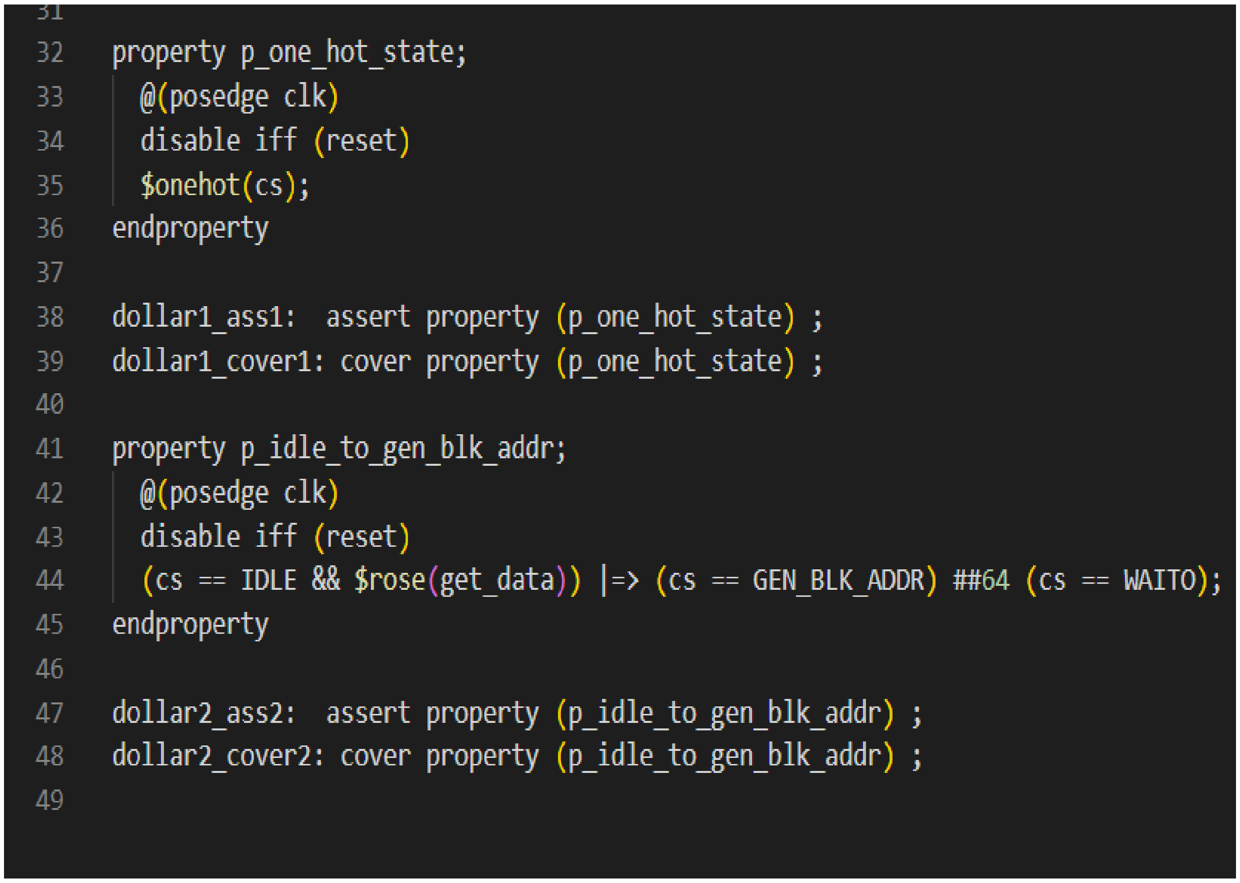
Question 2



C: > Users > CS > Downloads > Karem Wasem Diploma > session_ass4 _extra > sva.sv

```
property p_request_to_grant;
  @(posedge clk)
  disable iff (reset)
  $rose(request) |-> ##[2:5] $rose(grant);
endproperty

dollar1_ass1:  assert property (p_request_to_grant) ;
dollar1_cover1: cover property (p_request_to_grant) ;

property p_grant_acknowledgement;
  @(posedge clk)
  disable iff (reset)
  $rose(grant) |=> (frame == 0 && irdy == 0);
endproperty

dollar2_ass2:  assert property (p_grant_acknowledgement) ;
dollar2_cover2: cover property (p_grant_acknowledgement) ;

property p_transaction_complete;
  @(posedge clk)
  disable iff (reset)
  (frame && irdy) |=>  !grant;
endproperty

dollar3_ass3:  assert property (p_transaction_complete) ;
dollar3_cover3: cover property (p_transaction_complete) ;


```

Question 3



```
property p_one_hot_state;
  @(posedge clk)
  disable iff (reset)
  $onehot(cs);
endproperty

dollar1_ass1:  assert property (p_one_hot_state) ;
dollar1_cover1: cover property (p_one_hot_state) ;

property p_idle_to_gen_blk_addr;
  @(posedge clk)
  disable iff (reset)
  (cs == IDLE && $rose(get_data)) |=> (cs == GEN_BLK_ADDR) ##64 (cs == WAIT0);
endproperty

dollar2_ass2:  assert property (p_idle_to_gen_blk_addr) ;
dollar2_cover2: cover property (p_idle_to_gen_blk_addr) ;
```