SLOT: A1 + TA1

REG.NO.:

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRONICS ENGINEERING
CONTINUOUS ASSESSMENT TEST - II
WINTER SEMESTER 2025-2026

Programme Name & Branch : B.Tech VLSI Design
Course Code and Course Name : BEVD205L , **Scripting Languages and Verification**
Faculty Name(s) : Dr. J. Saikia, Dr. A. Tripathi
Class Number(s) : VL2025260500895, VL2025260500897
Date of Examination : 15/03/26
Exam Duration : 90 minutes **Maximum Marks:** 50

General instruction(s):

- Answer All Questions
- M - Max mark; CO – Course Outcome; BL – Blooms Taxonomy Level (1 – Remember, 2 – Understand, 3 – Apply, 4 – Analyse, 5 – Evaluate, 6 – Create)
- Course Outcomes:
  CO4: Understand the VLSI Verification Techniques
  CO5: Develop System Verilog Modules

Q. No	Question	M	CO	BL
1.	The Verilog module below implements a Moore FSM-based sequence detector that detects the overlapping bit pattern **"1011"** on a serial input stream. The module declaration is given below: ***module seq_detector_1011 (*** ***input clk, rst, din,*** ***output reg detected*** ***);*** Write a Verilog testbench to achieve maximum code coverage on this DUT. Your testbench must: **(a)** Apply a reset sequence and then drive a serial input stream that covers all state transitions of the FSM (S0 → S1 → S2 → S3 → S4/detect), including false-start paths (e.g., partial matches like "101" followed by a '0' that forces a state rollback). **(b)** Include at least three distinct test vectors: one that produces a detected output, one that applies a mid-stream reset to verify reset functionality, and one that deliberately applies a non-matching stream to exercise the default/else branches of the FSM. **(c)** Use **$monitor** or **$display** statements to log clk, rst, din, current state, and detected output at every clock edge. **(d)** State which simulator command-line option (e.g., for VCS, ModelSim) is used to enable code coverage collection.	10	4	3
2.	A design team is verifying a 16-bit ALU that supports the following operations: ADD, SUB, AND, OR, XOR, SHIFT_LEFT, SHIFT_RIGHT The team writes 30 directed tests, each verifying one operation using a few manually selected input values. Each test prints the output of the ALU to a log file, and engineers verify correctness by inspecting simulation waveforms. a) Identify two key weaknesses in the verification strategy used by the team	10	4	4

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRONICS ENGINEERING
CONTINUOUS ASSESSMENT TEST - II
WINTER SEMESTER 2025-2026

---

	that could allow a bug to escape detection. Explain your reasoning. b) Propose three improvements to the verification approach that could significantly increase the probability of detecting bugs.			
3.	Write a SystemVerilog function that: Accepts a dynamic array of 16-bit unsigned integers (representing ADC samples) as input. Returns a **packed struct** containing: the **peak** (maximum), **valley** (minimum), **range** (peak−valley), and **above_mid_count** (number of samples strictly greater than the midpoint of peak and valley). If the array is empty, return a zeroed struct and display **"No samples provided"**. If any sample contains unknown (X/Z) bits, skip it, track a separate **bad_count**, and display a warning with the count of bad samples after processing.	10	5	3

4.	Given the code:
	`module test;`
	`int q[$] = {1,2,3,4,5,6};`
	`initial begin`
	`foreach(q[i]) begin`
	`if(q[i] % 2 == 1)`
	`q.pop_front();`
	`end`
	`$display(q);`
	`end`
	`endmodule`
	attempting to display a queue that only has even elements.
	a) Go through each iteration and show the change in the queue.
	b) Identify any possible issue and make a change to the code to avoid it.
	c) What would be the final contents of the queue if the condition inside the loop were changed to:
	`if(q[i] % 2 == 0)`
	`q.pop_front();`
		10	2	3

5.	**(a)** Define a base class named Shape with: string name to store the shape name, A virtual function area() that returns a real value (default returns 0.0), A virtual function disp() that prints the shape name and its area using $display. **(b)** Derive two classes from Shape: (1) Circle: add a real radius field, Override area() to return π × radius², Override disp() to also print the radius. (2) Rectangle: add real width and real height fields, Override area() to return width × height, Override disp() to also print width and height. **(c)** Add a custom constructor to each derived class that initializes all fields (including calling **super.new()** with the shape name).	10	5	3

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRONICS ENGINEERING
CONTINUOUS ASSESSMENT TEST - II
WINTER SEMESTER 2025-2026

Key

1.

```
`timescale 1ns/1ps

module seq_detector_1011_tb;
   reg clk, rst, din;
   wire detected;
   // Instantiate DUT
   seq_detector_1011 dut (
      .clk(clk),
      .rst(rst),
      .din(din),
      .detected(detected)
   );
   // Clock generation
   always #5 clk = ~clk;
   // Task to apply serial stream
   task apply_stream(input [31:0] stream, input integer len);
      integer i;
      begin
         for (i = len-1; i >= 0; i = i - 1) begin
            din = stream[i];
            @(posedge clk);
         end
      end
   endtask
   // Monitor (assumes internal state is named 'state')
   initial begin
      $monitor("T=%0t | clk=%b rst=%b din=%b state=%0d detected=%b",
               $time, clk, rst, din, dut.state, detected);
               end
   initial begin
      clk = 0; rst = 1; din = 0;

      // --- Apply reset ---
      repeat(2) @(posedge clk);
      rst = 0;

      // ============================================================
      // TEST 1: Valid detection + overlapping + false start
      // Covers: S0→S1→S2→S3→S4 and rollback paths
```

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRONICS ENGINEERING
CONTINUOUS ASSESSMENT TEST - II
WINTER SEMESTER 2025-2026

```
    // Stream: 1 0 1 1 0 1 0 (contains "1011")
    // ============================================================
    apply_stream(7'b1011010, 7);

    // ============================================================
    // TEST 2: Mid-stream reset
    // ============================================================
    apply_stream(4'b1011, 4);   // partial detection                    [
    rst = 1;
    @(posedge clk);
    rst = 0;
    apply_stream(4'b1011, 4);   // verify restart works

    // ============================================================
    // TEST 3: Non-matching stream (default/else coverage)
    // ============================================================
      apply_stream(6'b111111, 6);

    // Finish
    #20 $finish;
  end
endmodule
```

Coverage:

```
vsim -coverage work.seq_detector_1011_tb
run -all
coverage report -all
```

2.

A) 1) Over-reliance on directed testing.
Directed tests cover a small test case exploration space.
2) Lack of automated testing.
Relies on manual wave inspection to determine correctness. Maybe missed by user error.

B) Improvements:

1) Test sequences. Rather than just one combination at a time.
SUB -> XOR
ADD -> SHIFT_LEFT
2) Use automated self-checking:.
3) Increase stimulus diversity with directed tests.

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF ELECTRONICS ENGINEERING
CONTINUOUS ASSESSMENT TEST - II
WINTER SEMESTER 2025-2026

3.

```
typedef struct packed {
logic [15:0] peak;
logic [15:0] valley;
logic [15:0] range;
int above_mid_count;
int bad_count;
} stats_t;

function automatic stats_t analyze_samples(input logic [15:0] samples[]);
stats_t s;
logic [15:0] mid;
int valid_count = 0;

// Initialize struct to zero
s = '{default:0};

// Empty array check
if (samples.size() == 0) begin
$display("No samples provided");
return s;
end

// First pass: find peak & valley, skip X/Z
foreach (samples[i]) begin
if ($isunknown(samples[i])) begin
s.bad_count++;
continue;
end

if (valid_count == 0) begin
s.peak = samples[i];
s.valley = samples[i];
end else begin
if (samples[i] > s.peak) s.peak = samples[i];
if (samples[i] < s.valley) s.valley = samples[i];
end
valid_count++;
end

// If no valid samples
```

```
if (valid_count == 0) begin
$display("No valid samples (all contained X/Z)");
return s;
end

// Compute range and midpoint
s.range = s.peak - s.valley;
mid = (s.peak + s.valley) >> 1;

// Second pass: count above midpoint
foreach (samples[i]) begin
if ($isunknown(samples[i])) continue;
if (samples[i] > mid) s.above_mid_count++;
end

// Warning for bad samples
if (s.bad_count > 0) begin
$display("Warning: %0d samples contained X/Z and were skipped", s.bad_count);
end

return s;
endfunction
```

4.

A) Iteration1: i=0, q[0] = 1, 1%2 =1 TRUE, popped 1. Queue now: {2,3,4,5,6}
Iteration2: i=1, q[1] = 3, 3%2= 1 TRUE, popped 2, Queue now: {3,4,5,6}
Iteration3: i=2, q[2] =5, 5%2 = 1 TRUE, popped 3, Queue now: {4,5,6}
No more iterations.

B) Pop_front shifts element left skipping one index.
Code Fix:

```
int temp[$];
foreach(q[i]) begin
        if(q[i] % 2 == 0)
                temp.push_back(q[i]);
        end
          $display(temp);
  end
```

C) Iteration1: i=0, q[0] = 1, 1%2 =0 FALSE, Queue now: {1, 2,3,4,5,6}
Iteration2: i=1, q[1] = 2, 2%2= 0 TRUE, popped 1, Queue now: {2,3,4,5,6}
Iteration3: i=2, q[2] =4, 4%2 = 0 TRUE, popped 2, Queue now: {3,4,5,6}

Iteration4: i=3, q[3] =6, 6%2 = 0 TRUE, popped 3, Queue now: {4,5,6}
No more iterations.
Queue: {4,5,6}

5.

```
class Shape;
string name;

function new(string name = "Shape");
this.name = name;
endfunction

// Virtual area function
virtual function real area();
return 0.0;
endfunction

// Virtual display function
virtual function void disp();
$display("Shape: %s | Area: %0.2f", name, area());
endfunction
endclass


// ------------------------------------------------
// Derived class: Circle
class Circle extends Shape;
real radius;

function new(string name, real radius);
super.new(name);
this.radius = radius;
endfunction

// Override area
function real area();
return 3.14159 * radius * radius;
endfunction

// Override display
function void disp();
```

```
$display("Shape: %s | Radius: %0.2f | Area: %0.2f",
name, radius, area());
endfunction
endclass


// ------------------------------------------------
// Derived class: Rectangle
class Rectangle extends Shape;
real width, height;

function new(string name, real width, real height);
super.new(name);
this.width = width;
this.height = height;
endfunction

// Override area
function real area();
return width * height;
endfunction

// Override display
function void disp();
$display("Shape: %s | W: %0.2f H: %0.2f | Area: %0.2f",
name, width, height, area());
endfunction
endclass
```