SLOT: A2 + TA2

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
Class Number(s) : VL2025260500896
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 module declaration of a 3-bit synchronous Gray code counter with active-high reset and enable is given below. When en=1 the counter advances in the Gray sequence; when en=0 it holds its value. *module gray_counter_3bit (* *input clk, rst, en,* *output reg [2:0] gray_out);* Write a task-based self-checking Verilog testbench for this module. Define a task check that takes **en** and **expected_gray** as inputs, drives the DUT for one clock cycle, and prints a PASS/FAIL message by comparing gray_out with expected_gray. From the initial block: - Call the task to verify the full Gray sequence 000 → 001 → 011 → 010 → 110 → 111 → 101 → 100 → 000. - Assert reset after the 4th count, verify the output returns to 000, and then resume counting from 000. - Apply en = 0 for three consecutive cycles and verify that gray_out remains unchanged. - Print HOLD-PASS / HOLD-FAIL accordingly.	10	4	3
2.	A combinational module has 8 input bits and 4 output bits. A verification team runs 50 simulation tests, each applying an input pattern and checking the output. Later they discover that 10 tests repeated input combinations already used earlier. All tests pass. a) Explain why passing these tests does not guarantee the correctness. b) Suggest two improvements to the verification strategy that would increase confidence in correctness without exhaustively testing all input combinations.	10	4	4
3.	Write a SystemVerilog module that models a vending machine controller with the following requirements: 1. Define an enumerated type *vend_state_t* with values: IDLE, COIN5,	10	5	3

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

	COIN10, COIN15, DISPENSE, CHANGE. 2. Define a struct type *vend_info_t* containing: a *vend_state_t* field for current state, an integer credit representing accumulated coins in rupees, and a logic dispense output flag. 3. The SystemVerilog module should: a. Accept coin inputs: coin5 and coin10 (one-hot; both zero = no coin). Item cost is fixed at 15 rupees. b. Cycle through states: accumulate credit → DISPENSE when credit >= 15 → CHANGE if credit > 15 → IDLE. c. Use the timer field as a 1-cycle hold in DISPENSE state before returning to IDLE. d. After reset, start in IDLE with credit = 0. 4. Display the current state and credit value at each clock edge using $display.			
4.	Write a SystemVerilog function named find_error that: - Accepts a dynamic array of integers (int) as input. - Returns the index of the first value greater than 100. - If no such value exists, return -1 and display a message indicating no error values detected. - If the array is empty, return -2 and display a message indicating no data available.	5	5	3
	b) A SystemVerilog module analyzes a sequence of device status codes. Write the SystemVerilog code to: - Declare and initialize a dynamic array of status codes. - Call the find_error function. - Display the index of the first detected error.	5		
5.	A verification environment needs to model network packets of different types. a) Create a base class packet with the following members: - integer variable id - integer variable size Include a virtual function info() that prints the packet id and size. b) Create two classes that inherit packet: - data_packet containing an additional variable payload - control_packet containing an additional variable command Override the info() function in both classes so that it prints all relevant variables. c) In a module: - Declare a queue of base-class handles named pkt_q. - Create one object of each derived class and assign values to their variables. - Insert the objects into the queue. - Use a loop to call info() for each element of the queue.	10	5	3

SLOT: A2 + TA2

REG.NO.:

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 tb_gray_counter_3bit;

reg clk, rst, en;
wire [2:0] gray_out;

// DUT Instantiation
gray_counter_3bit dut (
.clk(clk),
.rst(rst),
.en(en),
.gray_out(gray_out)
);

// Clock Generation
always #5 clk = ~clk;

// Task for checking output
task check;
input en_in;
input [2:0] expected_gray;
begin
en = en_in;
@(posedge clk); #1;

if (gray_out === expected_gray)
$display("PASS: en=%0d, expected=%b, got=%b", en, expected_gray, gray_out);
else
$display("FAIL: en=%0d, expected=%b, got=%b", en, expected_gray, gray_out);
end
endtask

// Test sequence
initial begin
clk = 0;
rst = 1;
en = 0;
```

```
// Apply reset
@(posedge clk);
rst = 0;

// Full Gray sequence
check(1, 3'b000);
check(1, 3'b001);
check(1, 3'b011);
check(1, 3'b010);

// Assert reset after 4th count
rst = 1;
@(posedge clk); #1;
rst = 0;

if (gray_out === 3'b000)
$display("RESET PASS: gray_out = %b", gray_out);
else
$display("RESET FAIL: gray_out = %b", gray_out);

// Resume counting from 000
check(1, 3'b001);
check(1, 3'b011);
check(1, 3'b010);
check(1, 3'b110);
check(1, 3'b111);
check(1, 3'b101);
check(1, 3'b100);
check(1, 3'b000);

// HOLD behavior (en = 0)
en = 0;
@(posedge clk); #1;
reg [2:0] hold_val;
hold_val = gray_out;

repeat (3) begin
@(posedge clk); #1;
if (gray_out === hold_val)
$display("HOLD PASS: value = %b", gray_out);
else
```

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

```
$display("HOLD FAIL: expected=%b, got=%b", hold_val, gray_out);
end

$finish;
end

endmodule
```

2. Key:
A)

Total input combinations = 2^8 =256.
Number of combinations actually tested- 256-40 =210.
Might not have tested some crucial corner cases.

B) Directed testing
- For appropriate corner case testing.

Test planning
- to avoid repeats and explore more space.

Increase the number of tests.
(Any 2)

3.

```
module vending_machine (
input logic clk,
input logic rst,
input logic coin5,
input logic coin10
);

// 1. Enumerated state type
typedef enum logic [2:0] {
IDLE,
COIN5,
COIN10,
COIN15,
DISPENSE,
CHANGE
} vend_state_t;

// 2. Struct definition
typedef struct {
vend_state_t state;
```

```
int credit;
logic dispense;
} vend_info_t;

vend_info_t curr, next;

// Timer for DISPENSE hold (1-cycle)
logic timer;

// ----------------------------
// State + Data Register
// ----------------------------
always_ff @(posedge clk or posedge rst) begin
if (rst) begin
curr.state <= IDLE;
curr.credit <= 0;
curr.dispense <= 0;
timer <= 0;
end
else begin
curr <= next;
if (curr.state == DISPENSE)
timer <= 1;
else
timer <= 0;
end
end

// ----------------------------
// Next-State Logic
// ----------------------------
always_comb begin
next = curr; // default hold

// default outputs
next.dispense = 0;

// Coin decoding
int coin_val;
coin_val = 0;
if (coin5) coin_val = 5;
if (coin10) coin_val = 10;
```

```
case (curr.state)

IDLE: begin
if (coin_val != 0) begin
next.credit = curr.credit + coin_val;

if (next.credit >= 15)
next.state = DISPENSE;
else if (next.credit == 5)
next.state = COIN5;
else if (next.credit == 10)
next.state = COIN10;
end
end

COIN5, COIN10: begin
if (coin_val != 0) begin
next.credit = curr.credit + coin_val;

if (next.credit >= 15)
next.state = DISPENSE;
else if (next.credit == 10)
next.state = COIN10;
end
end

DISPENSE: begin
next.dispense = 1;

// stay for 1 cycle, then decide next
if (timer) begin
if (curr.credit > 15)
next.state = CHANGE;
else begin
next.state = IDLE;
next.credit = 0;
end
end
end

CHANGE: begin
```

```
// return extra and reset
next.credit = 0;
next.state = IDLE;
end

default: next.state = IDLE;

endcase
end

// ----------------------------
// Display at every clock edge
// ----------------------------
always_ff @(posedge clk) begin
$display("Time=%0t | State=%s | Credit=%0d | Dispense=%0b",
$time, curr.state.name(), curr.credit, curr.dispense);
end

endmodule
```

4.Key:

A)

```
function int find_error(input int arr[]);
            if (arr.size() == 0) begin
            $display("No data available");
   return -2;
end

foreach (arr[i]) begin
   if (arr[i] > 100) begin
      return i;
   end
end

$display("No error values detected");
return -1;
endfunction
```

B)

```
module test;
int status_codes[];
int index;
```

```
initial begin

   // Declare and initialize dynamic array
   status_codes = new[5];
   status_codes = '{45, 72, 120, 88, 95};

   // Call the function
   index = find_error(status_codes);

   // Display result
   if (index >= 0)
      $display("First error detected at index %0d", index);
   else
      $display("Function returned value = %0d", index);

end
endmodule
```

5. Key:

```
class packet;

int id;
int size;

virtual function void info();
$display("Packet id=%0d size=%0d", id, size);
endfunction

endclass


class data_packet extends packet;

int payload;

function void info();
$display("DATA packet id=%0d size=%0d payload=%0d",
id, size, payload);
endfunction

endclass
```

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

```
class control_packet extends packet;

int command;

function void info();
$display("CONTROL packet id=%0d size=%0d command=%0d",
id, size, command);
endfunction

endclass


module test;

packet pkt_q[$];
data_packet dp;
control_packet cp;

initial begin

dp = new();
dp.id = 1;
dp.size = 64;
dp.payload = 100;

cp = new();
cp.id = 2;
cp.size = 32;
cp.command = 5;

pkt_q.push_back(dp);
pkt_q.push_back(cp);

foreach(pkt_q[i])
pkt_q[i].info();

end

endmodule
```