U/D/TZ

Reg. No: 24BVD

VIT
Vellore Institute of Technology

Final Assessment Test – April 2026

Course: **BEVD205L - Scripting Languages and Verificatior**

Class NBR(s): **0895 / 0897** Slo **1+TA1**

Time: **Three Hours** ax. Marks: **100**

- **KEEPING MOBILE PHONE/ANY ELECTRONIC GADGETS, EVEN IN 'OFF' POSITION IS TREATED AS EXAM MALPRACTICE**
- **DON'T WRITE ANYTHING ON THE QUESTION PAPER**

COs	CO Statements
CO1	Handle files, directories and manage processes using PERL scripts.
CO2	Handle files, directories and manage processes using TCL scripts.
CO3	Develop scripts for Automation.
CO4	Understand the VLSI Verification Techniques.
CO5	Develop System Verilog Modules
CO6	Develop System Verilog constrained random test environment.

BL – Blooms Taxonomy Level (1 – Remember, 2 – Understand, 3 – Apply, 4 – Analyse, 5 – Evaluate, 6 – Create)

Answer ALL Questions
(10 X 10 = 100 Marks)

1. a) Write a Perl script that performs the following: **[5] CO1 BL3**

- Accepts a filename from the command line.
- Opens and reads the file.
- Counts the number of lines, words and characters
- Prints the counts in the following format:

Lines: <count>; Words: <count>; Characters: <count>

Example:

Command line : Perl stats.pl data.txt

Output: Lines: 5; Words: 32; Characters: 180

b) Predict the output of this Perl program: **[5]**

```
my $str = "dog cat dog bird cat dog";
my %count;
foreach my $w (split " ", $str) {
$count{$w}++;
if ($count{$w} == 2) {
$count{$w} += 1;} }
print $count{"dog"}, "\n";
print $count{"cat"}, "\n";
print scalar(keys %count), "\n";
```

2. Write a Perl script that lists all currently running processes on the system, filters only those processes whose CPU usage exceeds a threshold value provided as a command-line argument, and writes the filtered process details (PID, name, CPU%) to a log file with a timestamp. CO1 BL2

3.(a) A log file named run.log records simulation phase in the following format: CO2 BL3

BEGIN INIT
BEGIN TEST
END TEST
BEGIN CLEANUP
END CLEANUP
END INIT

Write a TCL script to:

- Read run.log line by line and extract the action and phase name from each entry.
- Maintain an associative array named phase_status that stores the latest action recorded for each phase.
- After processing the entire file, identify phases whos last action is BEGIN.
- Store these phase names in a list named unclosed.
- Display them as:

Unclosed Phases:
<phase1>
<phase2> ....

OR

3.(b) Write a TCL script that creates a list of 10 signal names, sorts them alphabetically, removes duplicate entries, and displays the final cleaned list along with its length. CO2 BL3

4. A verification engineer runs a test suite on a combinational circuit and obtains the following coverage results: line coverage = 90%, branch coverage = 75%, and toggle coverage = 60%. Explain what each metric means, identify which parts of the design may be untested based on these results, and suggest what kind of additional test cases should be written to improve coverage. CO3 BL3

5. Write a SystemVerilog program that performs the following: CO5 BL3

- Declare a dynamic array of integers arr.
- Allocate space for 10 elements using new[] and initialize them with random values between 0 and 20.
- Display all elements of the array.
- Find and display all elements greater than 10.
- Compute and display the sum of all even elements.

6. For the given stack memory model specification, develop a SystemVerilog task to perform the push operation. The stack has signals: clk, rst_n, push enable (push), pop enable (pop), 8-bit data input (din), 8-bit data output (dout), full and empty status flags. CO5 BL4

Push Operation: push and din should be driven at the same clock cycle. Data is written only when full is not asserted.

Pop Operation: pop should be asserted for one clock cycle. Design will respond with dout in the next clock cycle only when empty is not asserted.

7.(a) Write a parameterized class Stack with parameter type T = int. CO5 BL3

The class should contain:

- a queue items[$] of type T
- a function push(T item) to insert an element.
- a function pop() to remove and returns the last element

In a module:

- Create two stack objects:
  (i) one storing int (ii) one storing bit [3:0]
- Push two values into each stack.
- Pop one value from each stack and display the results.

OR

7.(b) i) Develop the class constructor with arguments taking default values for the EthFrame class given below and write the display method externally to the EthFrame class to display the values of the class properties. CO5 BL3

```
class EthFrame;
  bit [47:0] src_mac;
  bit [47:0] dst_mac;
  bit [15:0] ether_type;
  //Write your code
endclass
```

ii) Define a module and create an array of 4 objects for the EthFrame class. Initialize each object with different values and display the properties of each object using the display method.

8. Write a SystemVerilog class FrameConfig with the following random variables: CO6 BL3

- mode (1-bit) - length (8-bit)

Apply constraints such that:

- If mode == 0, then length must be less than 10.
- If mode == 1, then length must be greater than 100.

In an initial block:

- Create an object of the class.
- Randomize the object five times using randomize().
- Display mode and length after each randomization.

9. SystemVerilog verification environments operate at the transaction level rather than directly manipulating signals. Answer the following: **CO6 BL2**

i) What is a transaction in a verification environment?

ii) Why are transactions preferred over direct signal manipulation?

iii) Explain how a transaction moves through the generator, mailbox, driver, monitor, and scoreboard?

iv) How does a mailbox help in synchronization and decoupling between generator and driver?

v) How does transaction – level modeling allow the same test bench to be reused across different DUTs?

10. The SystemVerilog code for a 4-bit Ring Counter is given below. **CO6 BL3**

```
module RingCounter(
 input  logic clk,
 input  logic rst_n,
 input  logic load,
 input  logic [3:0] seed,
 output logic [3:0] count
);
 always @(posedge clk) begin
  if (!rst_n)
   count <= 4'b0001;
  else if (load)
   count <= seed;
  else
   count <= {count[0], count[3:1]};
 end
endmodule
```

Write a monitor class

- Samples the count output of the ring
- Captures atleast 8 consecutive output values.
- Displays the sample values.

Write a scoreboard class

- Receives values and checks if the DUT follow a valid ring counter rotation. Report errors if any.

⇔⇔⇔ U/D/TZ ⇔⇔⇔