Lab Experiment- TCL Scripting

Objective:

This lab manual provides hands-on experience with TCL scripting, focusing on concepts relevant to digital design. You'll work through examples that demonstrate key TCL features without requiring specific EDA tools.

Prerequisites:

- Basic understanding of programming concepts
- Access to a TCL interpreter (e.g., tclsh)
- Text editor (e.g., nano, vim, or gedit)

Lab 1: Basic TCL Operations and Digital Design Calculations

Write a TCL script to perform basic operations and calculations common in digital design.

Steps:

1. Open your text editor and create a new file named **`digital_calc.tcl`**.
2. Copy the following script into the file:

```
# Basic digital design calculations

# Define clock frequency and calculate period
set clock_freq_mhz 100
set clock_period_ns [expr {1000.0 / $clock_freq_mhz}]
puts "Clock period: $clock_period_ns ns"

# Calculate power for a simple CMOS circuit
proc calc_power {capacitance voltage frequency}
{
  return [expr {$capacitance * $voltage * $voltage * $frequency}]
}
set cap_pf 10.0
set voltage 1.2
set power_mw [calc_power $cap_pf $voltage $clock_freq_mhz]
```

```
puts "Power consumption: $power_mw mW"

# Simple timing calculation
set prop_delay_ns 2.5
set setup_time_ns 0.5
set max_freq_mhz [expr {1000 / ($prop_delay_ns + $setup_time_ns)}]
puts "Maximum frequency: $max_freq_mhz MHz"
```

3. Save the file.
4. Open a terminal, navigate to the directory containing your script, and run:

```
tclsh digital_calc.tcl
```

5. Observe the output and verify the calculations.
6. Experiment with the script:
    a. Modify the *clock_freq_mhz* and observe how it affects other calculations.
    b. Add a new calculation, such as determining the number of clock cycles in a given time period.

Discussion:

- Explain this TCL script?
- What advantages does using procedures (like calc_power) offer in script organization?

Lab 2: Working with Lists and Digital Design Data

Create a TCL script to manipulate lists, simulating operations on digital design data.

Steps:

1. Create a new file named **`design_data.tcl`**.
2. Copy the following script into the file:

```
# Simulating digital design data operations
```

```
# Define a list of module names
set modules {ALU Register_File Decoder Multiplexer}

# Print all modules
puts "All modules:"
foreach module $modules { puts " $module" }

# Add a new module
lappend modules "Control_Unit"
puts "\nAfter adding Control_Unit:"
puts $modules

# Remove a module
set modules [lsearch -all -inline -not $modules "Decoder"]
puts "\nAfter removing Decoder:"
puts $modules

# Define a dict of module sizes (simulated gate count)
dict set module_sizes ALU 1000
dict set module_sizes Register_File 5000
dict set module_sizes Multiplexer 200
dict set module_sizes Control_Unit 1500

# Calculate total gate count
set total_gates 0
dict for {module size} $module_sizes {
    set total_gates [expr {$total_gates + $size}]
}

puts "\nTotal gate count: $total_gates"

# Find the largest module
set max_size 0
set largest_module ""
dict for {module size} $module_sizes {
   if {$size > $max_size} {
        set max_size  $size
        set largest_module $module
     }
    }
puts "Largest module: $largest_module with $max_size gates"
```

3. Save the file and Run the script

4. Observe how the script manipulates lists and dictionaries to simulate working with design data.
5. Experiment with the script:
    a. Add more modules and sizes.
    b. Implement a procedure to find modules larger than a given size.

Conclusion:

These labs demonstrate basic TCL scripting concepts in a digital design context. As you become more comfortable with TCL, you can apply these concepts to more complex scripts that could be used with actual EDA tools.

Helping Material:

- Tutorialspoint
- Tcl Manual