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.tc**l.
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" |

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

|  |
| --- |
| tclsh digital\_calc.tcl |

1. Observe the output and verify the calculations.
2. Experiment with the script:
   1. Modify the *clock\_freq\_mhz* and observe how it affects other calculations.
   2. 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" |

1. Save the file and Run the script
2. Observe how the script manipulates lists and dictionaries to simulate working with design data.
3. Experiment with the script:
   1. Add more modules and sizes.
   2. 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](https://www.tutorialspoint.com/tcl-tk/tcl_overview.htm)
* [Tcl Manual](https://www.tcl.tk/man/tcl8.5/tutorial/tcltutorial.html)