Owen-Ethan_905452983_palatics_Lab2

Ethan Owen
SID: 905452983
EE201A, Winter 2026, MSOL

Problem 1

Compute:

- Average wirelength
- Number of wires with length > 50

Include:

- Screenshot of TCL script for this section
- Results for each wirelength and number of wires with length > 50

Screenshot of TCL Script:

```
#!/usr/bin/tclsh

# Set the input file path
set input_file "wirelength.txt"

# Check if file exists
if {![file exists $input_file]} {
    puts "Error: File '$input_file' not found!"
    exit 1
}

# Open and read the wirelength file
set fp [open $input_file r]
set wirelengths [list]

# Read each line and store in list
while {[gets $fp line] >= 0} {
    if {[string trim $line] ne ""} {
        lappend wirelengths [string trim $line]
    }
}
close $fp

# Get total count
set count [llength $wirelengths]
```

```
# Check if we have data
if {$count == 0} {
    puts "Error: $input_file is empty!"
    exit 1
}

# Calculate average wirelength
set sum 0.0
foreach wl $wirelengths {
    set sum [expr {$sum + $wl}]
}
set avg [expr {$sum / $count}]

# Count wires with length > 50
set count_over_50 0
foreach wl $wirelengths {
    if {$wl > 50} {
        incr count_over_50
    }
}

# Print results
puts ""
puts "  Lab 2 - Problem 1: Wirelength Analysis Results"
puts "  Total number of wires:            $count"
puts "  Average wirelength:               $avg"
puts "  Number of wires with length > 50: $count_over_50"
puts ""

# Optional: Write results to a file
set output_file "problem1_results.txt"
set out_fp [open $output_file w]
puts $out_fp "# Lab 2, Problem 1 Results"
puts $out_fp "# Total wires: $count"
puts $out_fp "AVERAGE_WIRELENGTH: $avg"
puts $out_fp "NUMBER_OF_WIRES_OVER_50: $count_over_50"
close $out_fp

puts "Results to: $output_file"
puts ""
```

Results from TCL Script:

- Average wirelength: 47.39
- Number of wires > 50 length: 89

Problem 2

2A - Minimum Period

Fastest achievable clock was $460ps$

History of edits:

```
# History for modifying clk_period:
# Period | Slack
# 1000 365
# 610 23
# 605 18
# 600 33
# 500 1
# 495 0
# 490 1
# 480 1
# 470 0
# 460 0 => 1/T = 1/460 ps = 2.17 GHz
# 455 -17
# 450 -20
# 400 -64
# 100  -372
```

Earlier cases of the period resulting in a slack of 0 could indiciate local minimum rather than a global minimum. For this reason, as we reduced the period, we found other places where the slack was zero at lower periods, e.g. the global min.

Specifically, `syn_opt` is a heuristic algorithm and could converge at local minima and not hte global minimum, and since the optimzation effort wasn't set (defaults to medium not high), the synthesis might not have explored out of the valleys of the local minima.

2B - Min Period w/ Optimization

New Optimization Code in TCL file

```
# Allow Genus to dissolve hierarchy boundaries to merge logic
# Optimize all negative slack endpoints.
set_db auto_ungroup both
set_db tns_opto true

# Set effort levels to high for all stages of synthesis
set_db syn_generic_effort high
set_db syn_map_effort high
set_db syn_opt_effort high
set_db retime_effort high
```

```
# Standard synthesis from prior skeleton example
syn_generic
syn_map

# Define a new cost group for timing optimization, give a higher weight, and tell
synthesis to focus on it.
define_cost_group -name critical_path -weight 10 -design [get_designs *]

# assign paths relating to the clock signal (critical) to the cost group
path_group -from [all_registers -clock ${clkpin}] -to [all_registers -clock
${clkpin}] -group critical_path -name clock_paths

# Standard simulation
syn_opt
retime -min_delay
syn_opt -incremental
```

Primary Command Breakdown:

- `set_db auto_ungroup both` : allow genus to dissolve heirarchical boundaries to find better timing paths
- `set_db tns_opto true` : Fix the worst case single path for worst negative slack
- `set_db syn_generic_effort high` : set syn_generic to high effort mode
- `set_db syn_map_effort high` : set syn_map to high effort mode
- `set_db syn_opt_effort high` : set syn_opt to high effort mode
- `set_db retime_effort high` : set retime_effort to high effort mode
- `define_cost_group` : associate a cost group called critical path to give synthesis tools a priority with optimization, allowing them to focus on critical path
- `path_group` : add signals from clock to the critical path group and let synthesis focus on them
- `retime` : help balance unbalanced logic stages
- `syn_opt -incremental` : use incremental to perform clean up at the end to smooth out design

Summary

Fastest Clocks

- No optimzations, using skeleton: $460ps$
- Optimizations: $360ps$

Conversion to Max Clock Frequency:

$$\text{(Base) } T = 460ps$$
$$f_{base} = \frac{1}{460ps} = 2.17 \text{ GHz}$$

$$\text{(Optimized) } T = 360ps$$
$$f_{opt} = \frac{1}{360ps} = 2.77 \text{ GHz}$$