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 the global minimum, and since the optimization 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 optimizations, using skeleton: $460ps$
- Optimizations: $360ps$

Conversion to Max Clock Frequency:

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

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

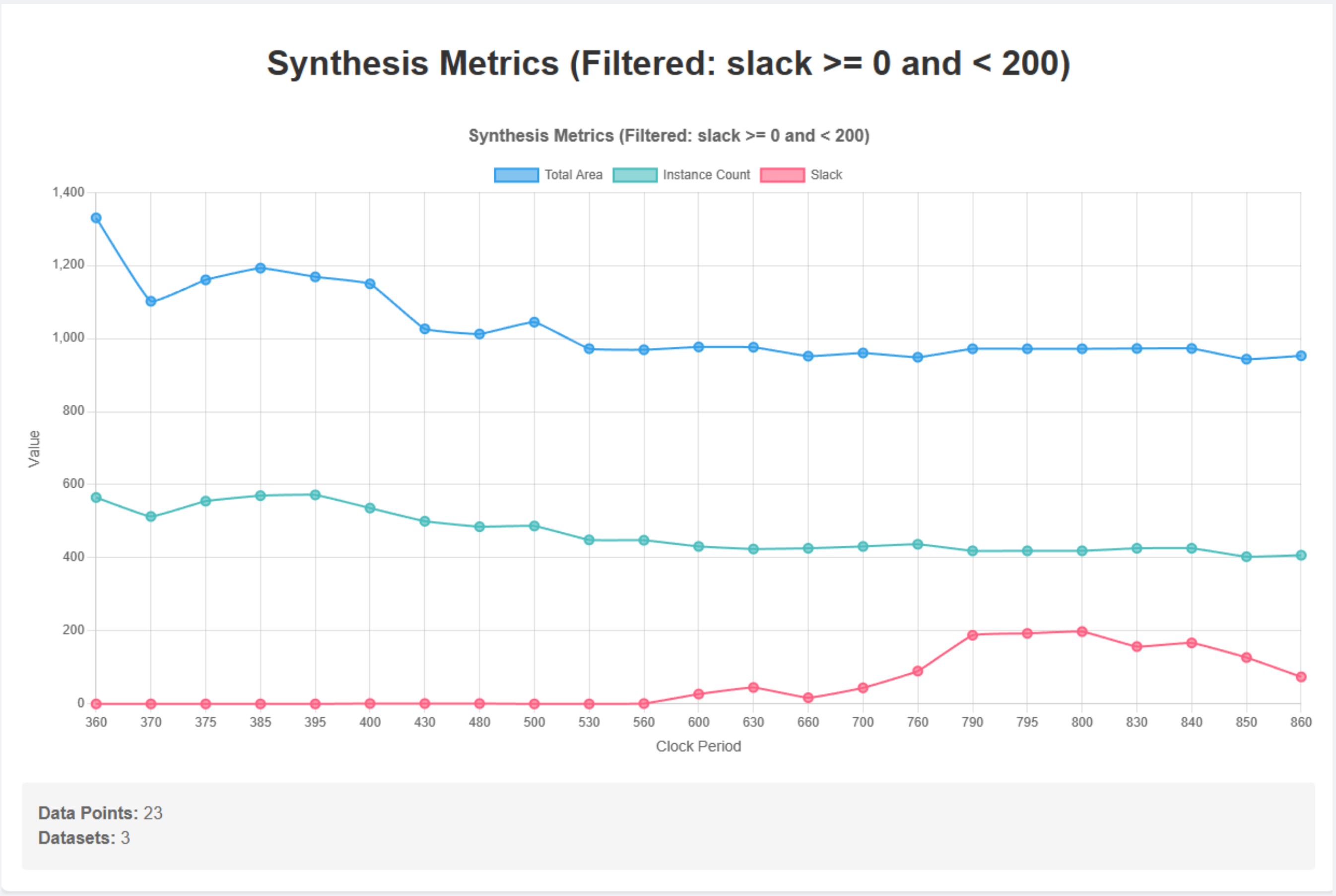
Problem 3

3A - Period and Area Analysis

List of data points used

```
period,total area,instances,slack
360,1330.266,565,0
370,1102.038,513,0
375,1160.558,555,0
385,1192.744,570,0
395,1168.804,572,0
400,1149.652,536,1
430,1026.494,500,1
480,1012.396,485,1
500,1044.848,487,0
530,972.23,449,0
560,969.304,448,1
600,977.018,431,27
630,976.22,424,45
660,951.748,426,17
700,960.526,431,44
760,948.822,437,90
790,971.964,419,188
795,971.964,419,193
800,971.964,419,198
830,972.762,426,157
840,972.762,426,167
850,943.502,403,127
860,952.812,407,74
```

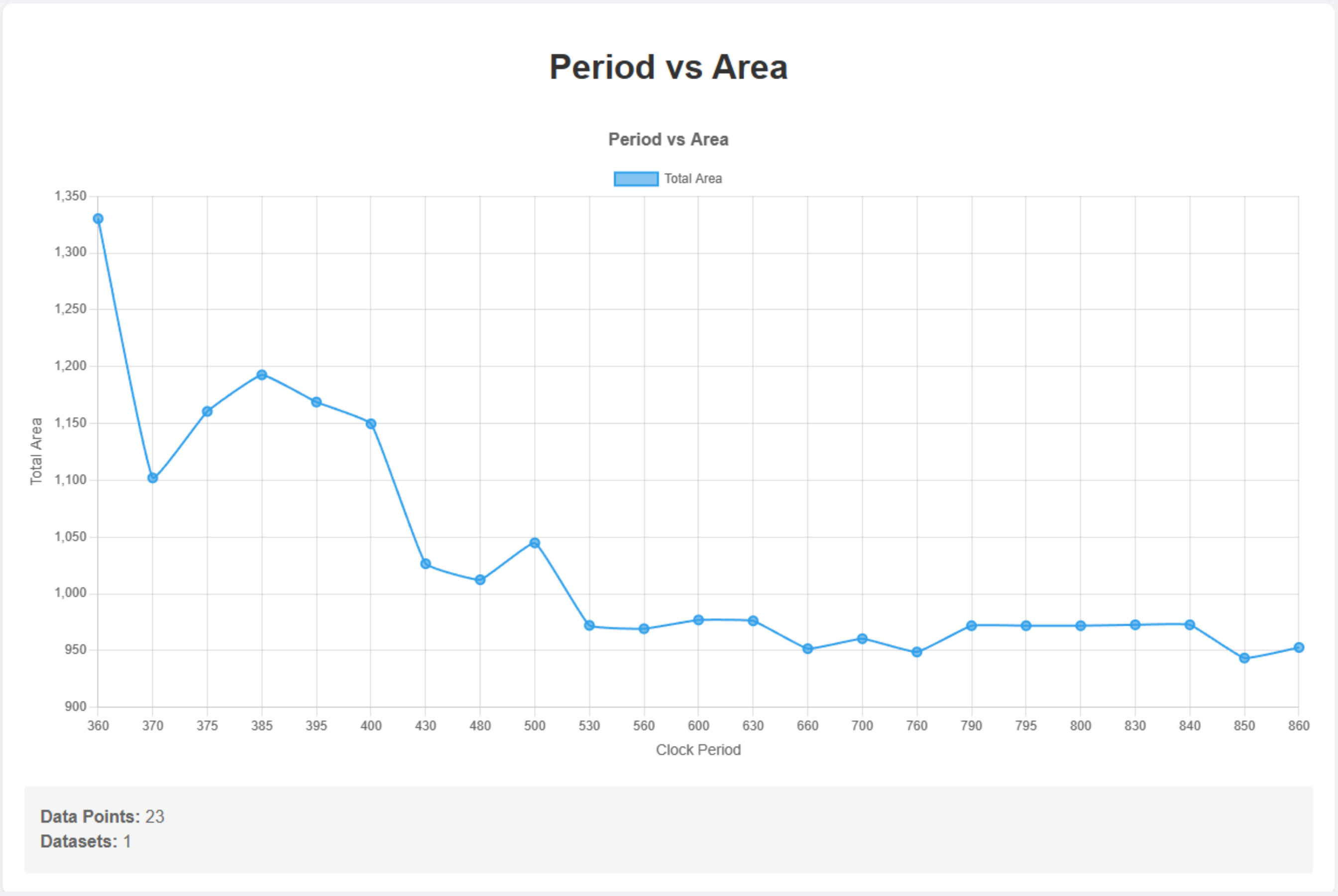
Plot of Period vs Area, Instances, and Slack



Data Points: 23
Datasets: 3

This was not requested for the problem, but it does show that the max slack was around 200 ps (198 ps specifically) and how it changes as we increase the period.

Plot of Period vs Area



Data Points: 23
Datasets: 1

Trend: As the period increases, the area tends to decrease. I think this is because for smaller periods, we are asking the logic to be faster, in which case the synthesis optimizations made in 2B force the design to trade off area for speed. Since we optimized for timing and not area/power, the area goes up to allow the circuit to be faster.

3B - Power Reduction

For this, I took the same commands used in 2B for the timing optimization and used them here in 3B as the baseline. Then I started adding/adjusting the code to work better for power optimization.

Pre-Elaboration Steps

These had to be done prior to the elaborate for the design. Doing this allows for flop insertion to break up the design and improve logic. This helps to reduce power but can have diminishing returns if it is not configured correctly. Then after elaborate, I set the min flops to 8. Setting this smaller greatly increases power because we insert too many small flops.

```
# Enable RTL clock-gating
set_db lp_insert_clock_gating true

# ... After elaborate
set_db [current_design] .lp_clock_gating_min_flops 8
```

Effort Adjustments

I first set effort levels for syn_generic, syn_map and retime to low instead of high. This stops the synthesis tool from immediately using massive areas right off the bat.

```
set_db syn_generic_effort low
set_db syn_map_effort low
set_db syn_opt_effort high
set_db retime_effort low
```

Adjust Power Optimization Settings

I did this to enable some additional optimization options for dynamic and leakage power. This drives synthesis to attempt to minimize the power as much as possible.

```
set_db opt_leakage_to_dynamic_ratio 0
set_db [current_design] .max_dynamic_power 0.0
set_db [current_design] .max_leakage_power 0.0
```

Retime and Simulation Passes

The intial prepare command for retime sets the design up. Then we do retime for min area and incremental syn_opt passes twice to really drive down the power. I noticed that with more repetitions, I got drastically reduced returns so I capped this part at 2 repetitions.

```
retime -prepare

# Synthesize with high effort and focus on area/power trade-off
```

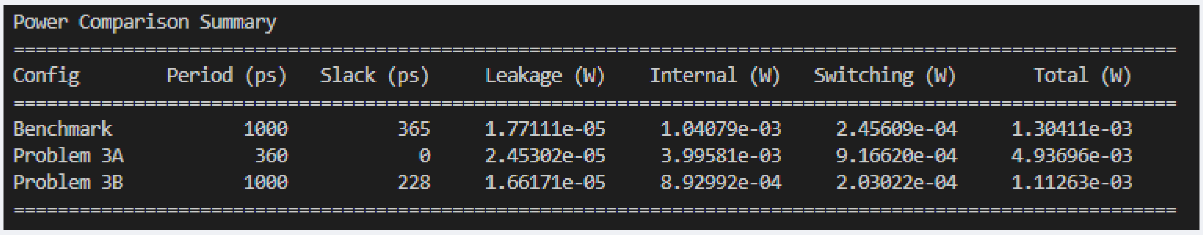
```
syn_generic
syn_map

# Initial optimization
syn_opt

# Perform retime -> synthesis passes to drive down power
retime -min_area
syn_opt -incremental

# Run a second time to further reduce power
# Running more than this has no returns, so just do this one as the final pass
retime -min_area
syn_opt -incremental
```

Results



```
Power Comparison Summary
=====================================================================================================
Config         Period (ps)   Slack (ps)     Leakage (W)    Internal (W)   Switching (W)      Total (W)
=====================================================================================================
Benchmark             1000          365     1.77111e-05     1.04079e-03     2.45609e-04     1.30411e-03
Problem 3A             360            0     2.45302e-05     3.99581e-03     9.16620e-04     4.93696e-03
Problem 3B            1000          228     1.66171e-05     8.92992e-04     2.03022e-04     1.11263e-03
=====================================================================================================
```

- *Total Power:* taken down from 1.304 mW to 1.112 mW
- *Leakage:* taken down from 17.7 uW to 16.6 uW
- *Switching:* take down from 245 uW to 203 uW
- 3A shows how much more power is used for optimizing for timing instead of power!