EE 5327 VLSI Design Laboratory

Lab 11 (1 week) – Signoff Analysis [1], [2]

PURPOSE:
The purpose of this lab is to perform signoff analysis on the gate-level netlist generated from IC Compiler II using Synopsys PrimeTime®. The signoff analysis is composed of Timing Analysis, including On-Chip Variation (OCV) Analysis, Power Analysis and Signal Integrity Analysis.

Background

Signoff is the final step in the ASIC design process. It refers to the various physical and electrical checks and analyses performed on the design before it is handed to a semiconductor fabrication facility (a "fab") for tape-out. This is a very crucial stage in the design cycle, as the chip is undergoing mass production and hence it is expected that everything works as per the specifications. The analysis broadly consists of the following:

1. Timing Analysis: Static Timing Analysis (STA), Statistical Static Timing Analysis (SSTA)
2. Power Analysis: Dynamic Power, Static (Leakage) Power
3. Signal Integrity (SI) Analysis: Crosstalk, Noise, Electro-Migration (EM) and IR-drop effects
4. Electrical Analysis: Design Rule Check (DRC), Layout Versus Schematic (LVS), Electrical Rule Check (ERC)

The Synopsys PrimeTime suite provides both post-synthesis and post-layout full-chip analysis of static timing, signal integrity, statistical timing, and full-chip power in a single integrated environment. In this lab, we will be using Synopsys PrimeTime to carry out the timing, power and signal integrity analyses mentioned above.

Static Timing Analysis (STA): Static timing analysis (STA) is a method of validating the timing performance of a design by checking all possible paths for timing violations. The design is broken down into timing paths, the signal propagation delay along each path is calculated, and violations of timing constraints inside the design and at the input/output interface are checked. Another way to perform timing analysis, called Dynamic Timing Analysis, is to use dynamic simulation, which determines the full behavior of the circuit for a given set of input stimulus vectors. Compared to dynamic simulation, STA is much faster because it is not necessary to simulate the logical operation of the circuit. STA is also more thorough because it checks all timing paths, not just the logical conditions that are sensitized by a set of test vectors. However, it can only check the timing, not the functionality, of a circuit design.

PrimeTime carries out STA in the following manner:

- Timing Paths: First the design is broken down into timing paths consisting of a start point (an input port or a register clock pin), a combinational logic network and an endpoint (a register data input pin or an output port).

  For example, the path shown in the below Figure 1 consists of the following:

  i. Path 1: Input path
  ii. Path 2: Reg2reg path
  iii. Path 3: Output path
  iv. Path 4: Comb path

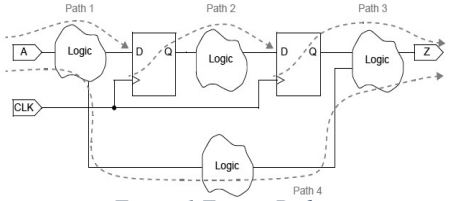

Figure 1 Timing Paths

- Delay calculation: The total delay is the sum of the cell delays and the net delay in the path.

i. Cell delay: Cell delay is the amount of delay from input to output of a logic gate in a path. In the absence of back-annotated delay information from a layout, PrimeTime calculates the cell delay from delay tables provided in the logic library for the cell.

ii. Net delay: Net delay is the amount of delay from the output of a cell to the input of the next cell in a timing path and is caused by the parasitic capacitance of the interconnection between the two cells, combined with net resistance and the limited drive strength of the cell driving the net. Before the layout phase, the delays are estimated from wire load models, while post-layout, the delay values are back-annotated.

- Constraint checks: After PrimeTime determines the timing paths and calculates the path delays, it can check for violations of timing constraints, such as setup and hold constraints. A setup constraint specifies how much time is necessary for data to be available at the input of a sequential device before the clock edge that captures the data in the device. This constraint enforces a maximum delay on the data path relative to the clock edge. A hold constraint specifies how much time is necessary for data to be stable at the input of a sequential device after the clock edge that captures the data in the device. This constraint enforces a minimum delay on the data path relative to the clock edge. In addition to setup and hold constraints, PrimeTime can also check recovery and removal constraints, data-to-data constraints, clock-gating setup and hold constraints, and the minimum pulse width for clock signals.

- Timing exception: If certain paths are not intended to operate according to the default setup and hold behavior assumed by PrimeTime, you need to specify those paths as timing exceptions. These include:

i. False path: This is a path that is never sensitized due to the logic values, the expected data sequence, or operating mode.

ii. Multicycle path – A path designed to take more than one clock cycle from launch to capture.

iii. Minimum or maximum delay path – A path that must meet a delay constraint that you explicitly specify as a time value.

During STA, a single operating condition is considered for the analysis i.e., there exists a single set of delay parameters for the entire design based on one set of Process, Voltage and Temperature (PVT) conditions. However, semiconductor device parameters can vary with such conditions and therefore, it becomes imperative to carry out the analysis in presence of such variations. Such an analysis is called On-Chip Variation (OCV) analysis and such a timing analysis falls in the category of the *Statistical STA* (SSTA). The goal of SSTA is to analyze each type of variation to arrive at a probability density function (PDF) that represents a statistical look at how the design will operate across variations in the underlying parameter. [3]

PrimeTime supports 3 types of OCV analysis:

1. Traditional OCV: A traditional OCV analysis allows both minimum and maximum delays to apply to different paths at the same time. For a setup check, it uses maximum delays for the launch clock path and data path, and minimum delays for the capture clock path. For a hold check, it uses minimum delays for the launch clock path and data path, and maximum delays for the capture clock path. You can also have PrimeTime adjust minimum delays and maximum delays by specified factors to model the effects of operating conditions. This adjustment of calculated delays is called derating.

   In the OCV mode, when a path segment is shared between the clock paths that launch and capture data, the path segment might be treated as having two different delays at the same time. Not accounting for the shared path segment can result in a pessimistic analysis. For a more accurate analysis, this pessimism can be corrected and is called Clock Reconvergence Pessimism Removal (CRPR), and is described next.

   CRPR: Clock Reconvergence Pessimism is an accuracy limitation that occurs when two different clock paths partially share a common physical path segment and the shared segment is assumed to have a minimum delay for one path and a maximum delay for the other path. This condition can occur any time that launch and capture clock paths use different delays, most commonly with OCV analysis. CRPR is the automated correction of this inaccuracy.

   In the Figure 2 below, each delay has a minimum and maximum value. The setup check at DFF2 considers the clock path to the source flip flop (CLK to DFF1/CLK) at the maximum value and the clock path to the destination flip flop (CLK to DFF2/CLK) at the minimum value.

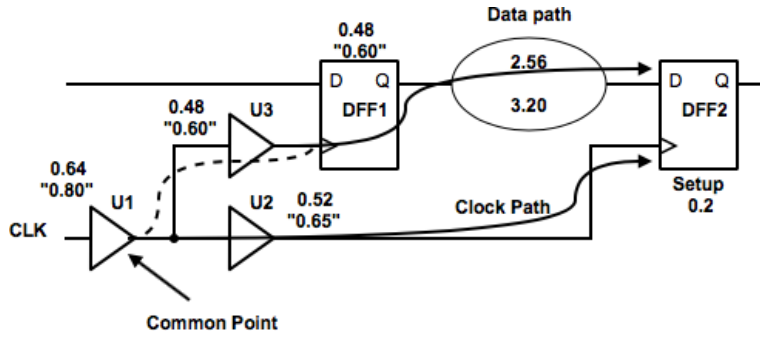

Figure 2 CRPR example

Although this is a valid approach, the test is pessimistic because clock path1 (CLK to DFF1/CLK) and clock path2 (CLK to DFF2/CLK) share the clock tree until the output of U1. Such a shared segment (cell U1) is called the common portion and the last cell output (output of U1) in the shared clock segment is called a common point. The setup check considers that cell U1 simultaneously has 2 different delays, 0.64 and 0.80, resulting in a pessimistic analysis in the amount of 0.16. This amount, obtained by subtracting the earliest arrival time from the latest arrival time at the common point, is called the *clock reconvergence pessimism*.

2. Advanced OCV (AOCV): Advanced on-chip variation (AOCV) analysis reduces unnecessary pessimism by taking the design methodology and fabrication process variation into account. AOCV determines derating factors based on metrics of path logic depth and the physical distance traversed by a particular path. A longer path that has more gates tends to have less total variation because the random variations from gate to gate tend to cancel each other out. A path that spans a larger physical distance across the chip tends to have larger systematic variations. AOCV is less pessimistic than a traditional OCV analysis.

3. Parametric OCV (POCV): Parametric on-chip variation (POCV) models the delay of an instance as a function of a variable that is specific to the instance. That is, the instance delay is parameterized as a function of the unique delay variable for the instance.

Power Analysis: As we saw in Lab 7, the power in a chip is comprised of 2 major components:

1. Leakage Power: Leakage power is the power dissipated by a cell when it is not switching, that is, when it is inactive or static. It consists of Intrinsic (Subthreshold) Leakage Power and Gate Leakage Power.

2. Dynamic Power: Dynamic power is the power dissipated when the circuit is active. A circuit is active any time the voltage on a net change due to some stimulus applied to the circuit. It is composed of Internal Power as well as Switching Power.

PrimeTime PX is a static and dynamic full-chip power analysis tool for complex multimillion-gate designs, intended for use within the PrimeTime environment. It provides a high degree of accuracy, performance, ease of use and comprehensive power diagnostics. Its high-capacity power analysis includes gate-level average and peak power verification.

i. Averaged Power Analysis: In the averaged power analysis mode, the switching activity consists of toggle rates and static probabilities. The accuracy of the power analysis results is proportional to the accuracy of the switching activity values. Therefore, the switching activity obtained from the RTL or gate-level simulation must be annotated correctly.

ii. Time-Based Power Analysis: In the time-based power analysis mode, the PrimeTime PX tool uses the events in the event-based switching activity file to calculate the power per event. It supports both RTL and gate-level activity files. When RTL-based switching activity files are provided, the tool propagates the RTL events per clock cycle to determine the activity on the non-annotated nets.

Signal Integrity: Signal integrity is the ability of an electrical signal to carry information reliably and to resist the effects of high-frequency electromagnetic interference from nearby signals. Crosstalk is the undesirable electrical interaction between two or more physically adjacent nets due to capacitive cross-coupling. As integrated circuit technologies advance toward smaller geometries, crosstalk effects become increasingly important compared to cell delays and net delays.

As circuit geometries become smaller, wire interconnections become closer together and taller, thus increasing the cross-coupling capacitance between nets. At the same time, parasitic capacitance to the substrate becomes less as interconnections become narrower, and cell delays are reduced as transistors become smaller.

PrimeTime SI is an add-on tool included in the PrimeTime Suite which is used to perform signal integrity analysis. PrimeTime SI can analyze and report two major types of crosstalk effects: delay and static noise.

Crosstalk delay effects: Crosstalk can affect signal delays by changing the times at which signal transitions occur. For example, Figure 3 shows the signal waveforms on cross-coupled nets A, B, and C.

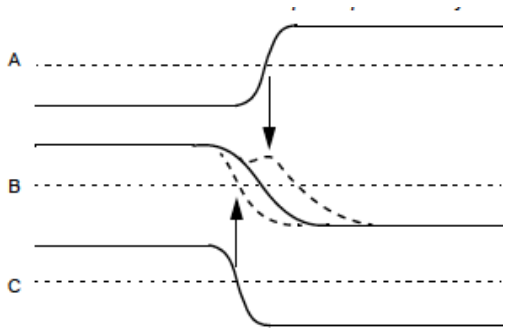

Figure 3 Transition slowdown or speedup caused by crosstalk

Because of capacitive cross-coupling, the transitions on net A and net C can affect the time at which the transition occurs on net B. A rising-edge transition on net A at the time shown in Figure 3 can cause the transition to occur later on net B, possibly contributing to a setup violation for a path containing B. Similarly, a falling-edge transition on net C can cause the transition to occur earlier on net B, possibly contributing to a hold violation for a path containing B.

A net that receives undesirable cross-coupling effects from a nearby net is called a victim net. A net that causes these effects in a victim net is called an aggressor net. Note that an aggressor net can itself be a victim net, and a victim net can also be an aggressor net. The terms aggressor and victim refer to the relationship between the two nets being analyzed.

The timing effect of an aggressor net on a victim net depends on several factors:

- The amount of cross-coupled capacitance
- The relative times and slew rates of the signal transitions
- The switching directions (rising, falling)
- The combination of effects from multiple aggressor nets on a single victim net

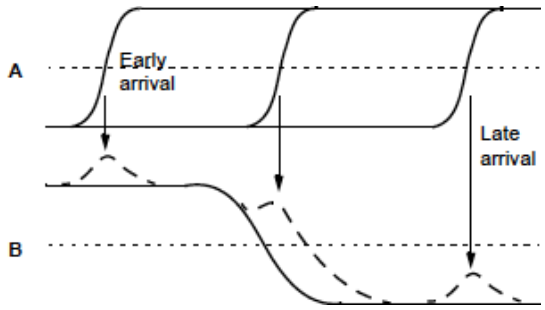

Figure 4 Effects of crosstalk at different arrival times

As shown in Figure 4, if the transition on A occurs at about the same time as the transition on B, it could cause the transition on B to occur later, possibly contributing to a setup violation; otherwise, it could cause the transition to occur earlier, possibly contributing to a hold violation.

If the transition on A occurs early, it induces an upward bump or glitch on net B before the transition on B, which has no effect on the timing of signal B. However, a sufficiently large bump can cause unintended current flow by forward-biasing a pass transistor. PrimeTime SI reports the worst-case occurrences of noise bumps. Similarly, if the transition on A occurs at a late time, it induces a bump on B after the transition on B, also with no effect on the timing of signal B. However, a sufficiently large bump can cause a change in the logic value of the net, which can be propagated down the timing path. PrimeTime SI reports occurrences of bumps that cause incorrect logic values to be propagated.

Static Noise analysis: Static noise analysis is a technique for finding the worst noise effects in a design so that they can be reduced or eliminated, thereby maximizing reliability. Static noise analysis, like static timing analysis, does not rely on test vectors or circuit simulation to determine circuit behavior. Instead, it considers the cross-coupling capacitance between aggressor nets and the victim net, the arrival windows of aggressor transitions, the drive characteristics of the aggressor nets, the steady-state load characteristics of the victim net driver, and the propagation of noise through cells. Using this information, PrimeTime SI determines the worst-case noise effects and reports conditions that could lead to logic failure.

Figure 5 compares static timing analysis and static noise analysis done by PrimeTime SI. For either type of analysis, the tool considers the cross-coupling between aggressor nets and

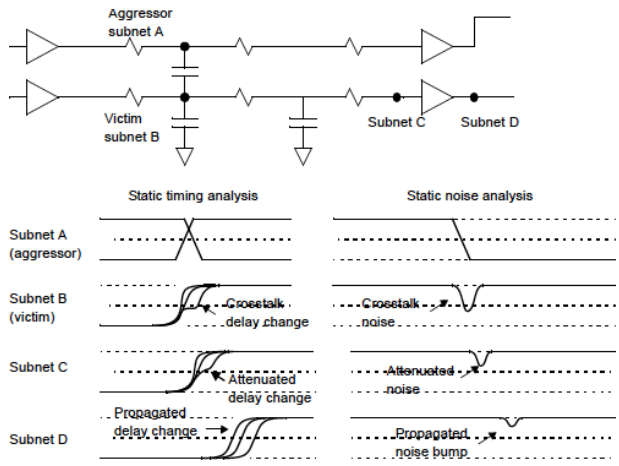

Figure 5 Crosstalk effects on timing and steady-state voltage

victim nets. For static timing analysis, the tool determines the worst-case change in delay on the victim net caused by crosstalk. For static noise analysis, it determines the worst-case noise bump or glitch on a steady-state victim net. (Steady-state means that the net is constant at logic 1 or logic 0.) A noise bump on a steady-state net can be propagated down the timing path and could be captured as incorrect data at the path endpoint.

Noise Bump Characteristics:

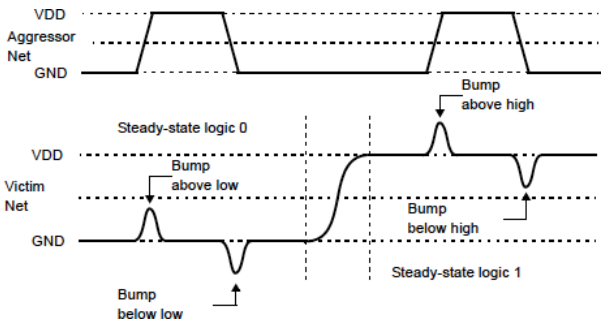

Figure 6 Noise bump types

The term "noise" in electronic design generally means any undesirable deviation in voltage of a net that ought to have a constant voltage, such as a power supply or ground line. In CMOS circuits, this includes data signals being held constant at logic 1 or logic 0. There are many different causes of noise such as charge storage effects at p-n junctions, power supply noise, and substrate noise. However, the dominant noise effect in deep-submicron CMOS circuits is crosstalk noise between physically adjacent logic nets. Noise effects, when plotted as voltage versus time, can have many different forms: bumps, ripples, random noise, and so on. PrimeTime SI concentrates on four types of noise bumps typically caused by aggressor net transitions: above low, below low, above high, and below high, as shown in Figure 6.

Bumps between the two rail voltages (above low and below high) can cause logic failure if they exceed the logic thresholds of the technology. Bumps outside of the range between the two rail voltages (below low and above high) are also important because they can forward-bias pass gates at the inputs of flip-flops and latches, allowing incorrect values to be latched.

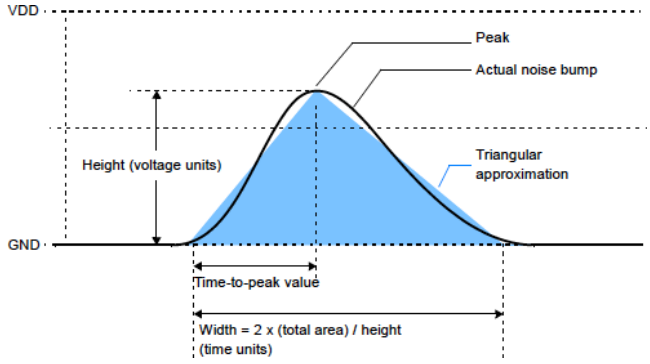

Figure 7 Noise bump characteristic

The characteristics of a noise bump are:

- Height in voltage units
- Total width in time units
- Time-to-peak ratio, which is the time-to-peak divided by the total bump width; a time-peak-ratio of 0.5 indicates a symmetrical bump with equal rising and falling times.

Now to the actual work:

PrimeTime tool and data setup:

1. Create a directory `lab11` under your `labs` directory and change into that:
   ```
   $ mkdir lab11
   $ cd lab11
   ```
2. Create the directories `inputs` and `scripts` inside the `lab11` directory.
   ```
   $ mkdir ./inputs ./scripts
   ```
3. Download the project-level setup file, `.synopsys_pt.setup` file from the Canvas page to the `lab11` directory that you just created. The project-level setup file, like the one we used in Design Compiler and IC Compiler II, consists of the definitions for the following application variables: `search_path`, `target_library`, `link_library` and `mw_reference_library`.
4. Download the PrimeTime reference flow script `pt_flow.tcl` from the Canvas page to the `scripts` directory.
5. Change to the inputs directory and copy all required output files from icc2.

```
$ cd ./inputs
$ cp ../../lab9/outputs/counter_cr.sdf .
$ cp ../../lab9/outputs/counter_cr.saif .
$ cp ../../lab9/outputs/counter_cr.v counter_cr.vg
$ cp ../../lab9/outputs/ counter_cr.spef.full_chip_counter_cr.tlu_min_25.spef.gz counter_cr.spef.min.gz
$ cp ../../lab9/outputs/ counter_cr.spef.full_chip_counter_cr.tlu_max_25.spef.gz counter_cr.spef.max.gz
```

Under the inputs directory, you will have the following back-annotated files:

i. Gate-level netlist, counter_cr.vg

ii. Timing constraints file, counter_cr.sdc

iii. Delay information file, counter_cr.sdf. A floorplanner or router can provide more detailed and accurate delay information, which you can provide to PrimeTime for a more accurate analysis. This process is called delay back-annotation. Back-annotated information is often provided in an SDF file.

iv. Switching-activity annotated file, counter_cr.saif

v. Extracted parasitics files for the max and min corners, counter_cr.max.spef.gz and counter_cr.min.spef.gz

We will now see how we can generate the reports for analysis.

6. First we will read in the gate level netlist using the `read_verilog` command and then set the `current_design` to reflect the full chip module of our design.

```
read_verilog ./inputs/${DESIGN_NAME}.vg
current_design full_chip_${DESIGN_NAME}
```

7. Next, we will read in the SDC file and the SDF file using the `read_sdc` and `read_sdf` commands, respectively.

```
read_sdc ./inputs/${DESIGN_NAME}.sdc
read_sdf -analysis_type bc_wc ./inputs/${DESIGN_NAME}.sdf
```

If you encounter error while reading in the SDF, run the below command in the terminal.

```
sed -i 's/\[/\\[/g; s/\]/\\]/g' inputs/counter_cr.sdf
```

8. For doing power analysis, we need to turn on the power analysis, as PrimeTime turns off the power analysis by default. We can do this by setting the value of the variable `power_enable_analysis` to `true`. We can then read the SAIF file using the `read_saif` command.

```
set power_enable_analysis true
read_saif ./inputs/${DESIGN_NAME}.saif -strip_path full_chip_${DESIGN_NAME}
```

9. For the SPEF file, note that there are 2 files available corresponding to the max and min corners. IC Compiler II generates 2 files consisting of max and min delays for the same elements corresponding to these two corners. However, PrimeTime can read and annotate only 1 SPEF file for the single-operation mode. Any attempt to read another file consisting of delay information for the same elements will result in an error. Therefore, we will read only the max SPEF file using the `read_parasitics` command for the setup analysis now. Later, for the hold analysis, we can remove the annotated parasitic information using the `remove_annotated_parasitics` command and read the min SPEF file.

```
read_parasitics -format SPEF ./inputs/${DESIGN_NAME}.spef.min.gz
```

10. Next, we set the fast and slow corners for the operating conditions using the `set_operating_conditions` command:

```
set_operating_conditions  -max tt0p8v25c -min tt0p8v25c
```

11. We can now create the timing path groups using the `group_path` command:

```
group_path -name inputs -from [all_inputs]
group_path -name output -to [all_outputs]
group_path -name comb -from [all_inputs] -to [all_outputs]
group_path -name reg2reg -from [all_registers -clock_pins] -to [all_registers
-data_pins]
```

12. We are now ready to begin our initial analysis. First, we need to check if our timing report from PrimeTime is consistent with the timing from IC Compiler II after the APR stage. This is called IC Compiler to PrimeTime (ICC2PT) correlation and is carried out as a sanity check to ensure the application settings and design collaterals are consistent between both of the tools.

    Before that, comment out the following lines as shown below:

```
############################
# Power report for clock-gating savings
redirect -file ./reports/${DESIGN_NAME}.power_clock_gating.rpt
{report_clock_gate_savings}
lappend COMMENTED_OUT {
############################
# Setting the operating conditions to OCV
set_operating_conditions -analysis_type on_chip_variation -max tt0p8v25c -
min tt0p8v25c
..............
puts [format "##### Done. Runtime: %.2f mins, Memory Used: %.1f GB #####"
$elapsed_time [expr [mem] / 1000000.0]]
}
```

    Source the reference flow script using the command below:

```
$ pt_shell -f ./scripts/pt_flow.tcl
```

    Open the GUI window of PrimeTime by typing `start_gui` in the `pt_shell` console

```
pt_shell > start_gui
```

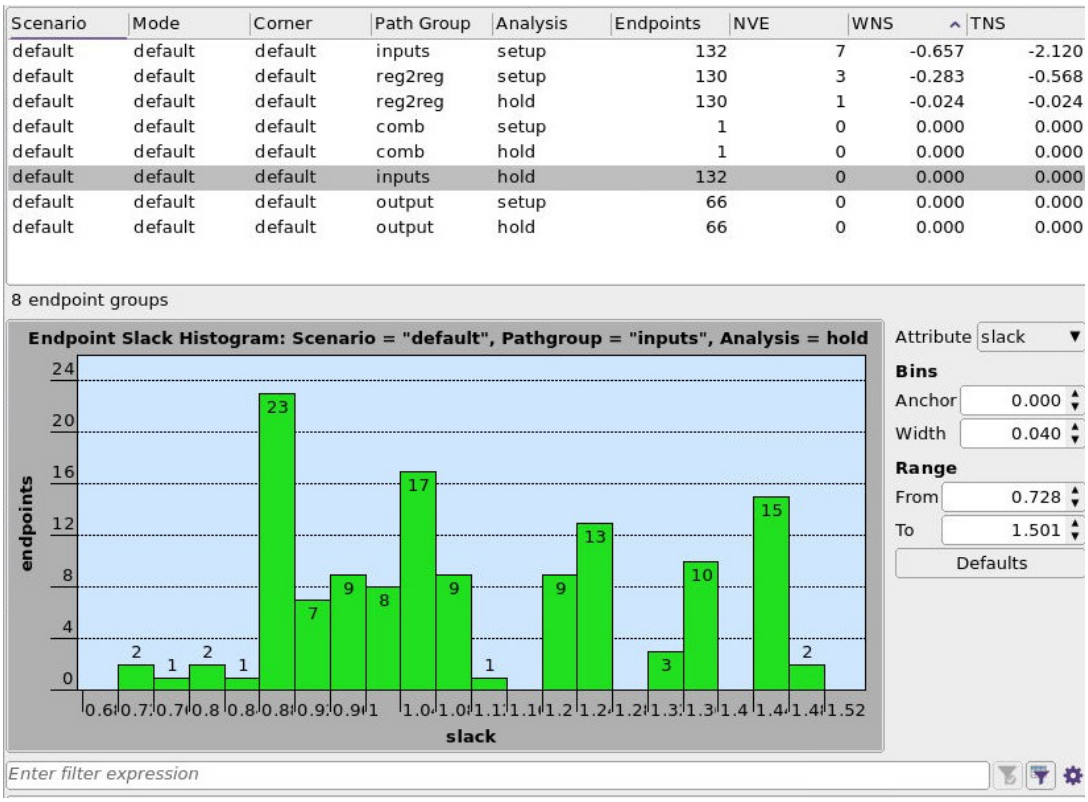

Figure 8 Path Slack histogram

After the GUI opens, you can see a histogram like in Figure 8. Right above the histogram, there is a box with options that lets you choose to display a histogram based on different scenarios. Save screenshots of the histograms.

13. In a separate terminal, open the timing report `counter_cr.timing_setup.rpt` in the `./reports` directory and compare it with the corresponding `counter_cr.timing_setup.rpt (or counter_cr.qor.rpt)` report generated in Lab 10. We see that the timing exactly correlates for the reg2reg path group whereas there is a slight difference in the slack for the input and output path groups. Hence, we can conclude that our timing correlation is successful. We can do a similar analysis for the hold constraint as well by removing the annotated parasitics and reading the min SPEF file.

14. Switch to the PrimeTime GUI window. Click on **Power ➔ Show Power Analysis Driver**. A window similar to Figure 9 appears.

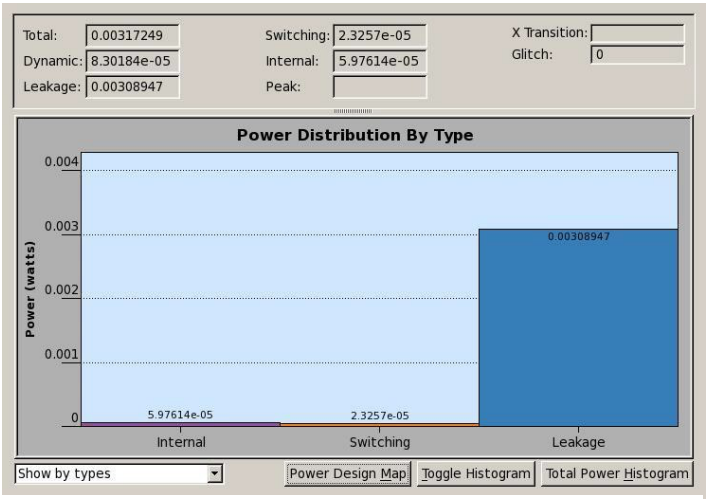

Figure 9 Power Density histogram

Save the screenshot of the histogram of Power Distribution by clicking on **File ➔ Save Screenshot As.** Click on **Power Design Map** button. A map of the **Total Power Density** appears. Switch the Analysis type to **Internal Power Density**, **Switching Power Density** and **Leakage Power Density** and check the corresponding Power Density Maps.

15. Similarly, we will now analyze the power report generated for initial analysis. Open the power report, `counter_cr.power.rpt` in the `./reports` directory and compare it with the corresponding report generated during Lab 10, `counter_cr.power.rpt`.
    Similarly, open the Threshold Voltage Group Report `counter_cr.power_vth.rpt`. This report shows the composition of the Multi-Vt cells for the different power groups. Now open the report `counter_cr.power_leakage_vth.rpt` to view the distribution of Leakage Power for the different Multi-Vt cells.
    Now open the `counter_cr.power_clock_gating.rpt` to view the Clock-Gating Efficiency of the design.
16. Let's now move on to OCV analysis. First, we will have to change our operating conditions from single operation mode to OCV mode using the `set_operating_mode` command. We will use `ss0p95v125c` for the max delay and `ff0p95vn40c` for the min delay.

```
set_operating_conditions -analysis_type on_chip_variation -max ss0p95v125c
-min ff0p95vn40c
```

    Comment out the lines shown below.

```
lappend COMMENTED_OUT {
################################
# Generating timing reports for hold analysis -OCV
remove_annotated_parasitics -all
.............
puts [format "##### Done. Runtime: %.2f mins, Memory Used: %.1f GB #####"
$elapsed_time [expr [mem] / 1000000.0]]
}
```

Source the reference flow script using the below command to generate the timing reports for setup analysis:

```
$ pt_shell -f ./scripts/pt_flow.tcl
```

Open the timing report `counter_cr.timing_ocv_setup.rpt` in the `./reports` directory and check the clock reconvergence pessimism value for each of the timing paths reported. You can see that the value is 0 for input and output path groups but has a non-zero value for the reg2reg path groups. Note the startpoint and endpoint of the reg2reg path and report the CRPR calculation for the path using the `report_crpr` command in the shell:

```
pt_shell > redirect -file ./reports/${DESIGN_NAME}.ocv_setup.crpr.rpt \
{report_crpr -from counter_cr_inst_count_regx0x/CLK -to \
counter_cr_inst_count_regx63x/CLK}
```

Open the GUI and save the screenshot of the histogram of the path slack. Similarly, generate the timing report, the histogram of the path slack and the CRPR report for the OCV hold analysis by commenting the lines below. Save the CRPR report as `${DESIGN_NAME}.ocv_hold.crpr.rpt`

```
lappend COMMENTED_OUT {
##########################
# Setting the timing derating factors
...............
puts [format "##### Done. Runtime: %.2f mins, Memory Used: %.1f GB #####"
$elapsed_time [expr [mem] / 1000000.0]]
}
```

17. The OCV analysis takes into consideration the minimum and maximum delays in the SDF file for the early and late cases of different paths. A more pessimistic analysis can be done by applying derating factors for these delays using the `set_timing_derate` command. Use the man pages to learn more about the `set_timing_derate` command, and set the timing derate as follows:

	Cell Delay	Net delay
Early	0.9	0.95
Late	1.05	1.1

```
##########################
# Setting the timing derating factors
set_timing_derate _____________________________________
.
.
.
```

18. Now comment the lines below to generate the timing reports for the setup analysis with the applied derating factors.

```
lappend COMMENTED_OUT {
################################
# Generating timing reports for hold analysis - with Derate
remove_annotated_parasitics -all
.............
puts [format "##### Done. Runtime: %.2f mins, Memory Used: %.1f GB #####"
$elapsed_time [expr [mem] / 1000000.0]]
}
```

Run the reference flow using the command below to generate the timing reports:

```
$ pt_shell -f ./scripts/pt_flow.tcl
```

Open the GUI and save the screenshot of the histogram of the path slack for the setup analysis. Open the timing report `counter_cr.timing_derated_setup.rpt` and report the CRPR of the top violating path of the reg2reg path group. Save the report as `${DESIGN_NAME}.derate_setup.crpr.rpt`.

Similarly, generate the timing reports, histogram of the path slack report and the CRPR of the top violating path of the reg2reg path for the hold analysis. Save the report as `${DESIGN_NAME}.derate_hold.crpr.rpt`. Comment the lines below to achieve this:

```
lappend COMMENTED_OUT {
############################
# Setting PT-SI app variable to enable SI analysis
set_app_var si_enable_analysis true
.............
puts [format "##### Done. Runtime: %.2f mins, Memory Used: %.1f GB #####"
$elapsed_time [expr [mem] / 1000000.0]]
}
```

19. We will now move on to SI analysis with the max SPEF. To do this, we need to set the variable `si_enable_analysis` to true. It is also recommended to set the variable `timing_save_pin_arrival_and_slack` to true before reporting the SI analysis.

```
set_app_var si_enable_analysis true
set timing_save_pin_arrival_and_slack true
```

20. We can now report any SI bottlenecks in the design using the `report_si_bottleneck` in the command. Comment out the lines below in the reference flow script:

```
lappend COMMENTED_OUT {
############################
# Set noise calculation related parameters and check noise
```

```
set_noise_parameters -include_beyond_rails

.............
puts [format "##### Done. Runtime: %.2f mins, Memory Used: %.1f GB #####"
$elapsed_time [expr [mem] / 1000000.0]]
}
```

Run the reference flow using the following command to generate the timing reports:

```
$ pt_shell -f ./scripts/pt_flow.tcl
```

21. You can open the report `counter_cr.si_analysis.rpt` to see if there are any nets reported in the design. In our design, there are no nets reported for SI issues. If there are any nets in the design reported for such issues, you can report their crosstalk delays using the following command:

    ```
    report_delay_calculation -crosstalk -from pin -to pin
    ```

22. Next, we need to analyze the crosstalk (static) noise in our design. First, we must set the noise parameters to include the voltages below the low rail and above the high rail for the noise analysis. (By default, only voltages above the low rail and below the high rail are considered.) We can do this by the `set_noise_parameters` command.

    ```
    set_noise_parameters -include_beyond_rails
    ```

23. Next, we can annotate the pins with the noise pin in the library using the `set_noise_lib_pin` command. For this lab purpose, we will be using libarary information that is applied by default.

24. We can report the noise in the design using the `report_noise` command. Before that we should check if all the pins are constrained for the noise analysis by using the `check_noise` command and include the beyond rail voltages for the analysis.

    ```
    check_noise -beyond_rail
    ```

25. Comment out the below lines in the reference flow script:

    ```
    lappend COMMENTED_OUT {
    ############################
    # Check crosstalk effects on the design (using min spef)

    .............
    puts [format "##### Done. Runtime: %.2f mins, Memory Used: %.1f GB #####"
    $elapsed_time [expr [mem] / 1000000.0]]
    }
    ```

Run the reference flow using the following command to generate the noise reports:

```
$ pt_shell -f ./scripts/pt_flow.tcl
```

26. To check if all of the pins in the design are constrained for noise analysis, open the `counter_cr.check_noise.rpt` and verify if the number of pins reported in the "none" row of the report is zero.
27. Open the `counter_cr.noise.rpt` to analyze the noise characteristics. By default, the `report_noise` command by itself, without any options, reports the bump characteristics and noise slack for the pin with the smallest (or most negative) slack for each type of noise bump. Width values are in library time units such as nanoseconds, height values are in library voltage units such as volts, and noise slack area values are in library voltage-time units such as volt-nsec.
28. Finally, remove the commented lines to perform the noise analysis with min SPEF:

```
puts [format "##### Done. Runtime: %.2f mins, Memory Used: %.1f GB #####"
$elapsed_time [expr [mem] / 1000000.0]]
lappend COMMENTED_OUT {
}
```

Run the reference flow using the command below to generate the noise reports:

```
$ pt_shell -f ./scripts/pt_flow.tcl
```

29. The results are appended to the same report files, `counter_cr.check_noise.rpt` and `counter_cr.noise.rpt` for the analysis with the min SPEF.
30. Finally, repeat the above steps 1-29 to perform the signoff analysis for the gate-level netlist of the stop_watch design from Lab 10.

What to submit:

1. **Summarize your results for both designs (i.e., counter_cr and stop_watch), in separate tables for each design, as shown below: (Mention appropriate units in all column headers.)**

Timing analysis:

Timing Slack	**Single operating point**		**With OCV**		**With derates**	
	Setup	**Hold**	**Setup**	**Hold**	**Setup**	**Hold**
Input						
Ouput						
Reg2reg						

Power analysis:

Switching Power	Internal Power	Leakage power			Total Power	Average Register Gating efficiency
		HVT	LVT	SVT		

Noise analysis:

SPEF	Type	Width	Height	Slack
Max SPEF	above_low			
	below_high			
	below_low			
	above_high			
Min SPEF	above_low			
	below_high			
	below_low			
	above_high			

2. **Attach screenshots of the path slack histogram for the setup and hold analysis with the single operating condition, OCV and timing derates.**
3. **Attach screenshots of the Total Power Density histogram.**
4. **Submit the** `EE5327_Lab11_x500_*.zip` **file generated in the** `./reports` **directory.**

Also, answer the following questions:

1. What is the difference in the slack values obtained from ICC2 runs and PT runs for the input and output path groups? What is the reason behind the difference? Explain in detail with calculations/figures. (Hint: Observe the output of the `read_sdc` command in your PT runs.)
2. Why is the difference in slack 0 (or at most 10 ps) in the case of the reg2reg path group? Explain in detail with calculations/figures.
3. What are the different modes available for the `power_analysis_mode` in PrimeTime? What are the differences between them? Indicate the default mode.
4. What is the effect of setting the variable `timing_save_pin_arrival_and_slack` to `true`?

Acknowledgments

Portions of this lab guide have included material from the following sources:

[1] PrimeTime® User Guide, Version M-2016.12, December 2016

[2] PrimeTime® PX User Guide Version M-2016.12, December 2016

[3] http://vlsicad.ucsd.edu/SSTA.htm