ECE 524-Synthesis & Verification of Digital Circuits
Spring 2015

Synopsys-PrimeTime Tutorial

Instructor: Dr. Spyros Tragoudas
Office: E-0202b
Email: spyros@engr.siu.edu

Teaching Assistant: Wisam Aljubouri
Office: A-119
Email: wisam@siu.edu

Department of ECE
Southern Illinois University Carbondale

1 Introduction

This Tutorial aims in giving a brief introduction on how to use Synopsys PrimeTime(PT) to perform static timing analysis. **This is neither a full tutorial, nor a full manual about timing analysis using the Synopsys PrimeTime**. The user manual of this tool is available in the path **/synopsys/DOCS/primetime/ptugf.pdf**. Actually this user manual is one of the available manual of primetime under synopsys directory.

1.1 Brief Introduction About Timing analysis

Timing Analysis is a method of verifying the timing performance of a design by checking for all possible timing violations in all possible paths. Timing analysis is integral part of ASIC/VLSI design flow. Anything else can be compromised but not timing. Timing is important because just designing the chip is not enough; we need to know how fast the chip is going to run, how fast the chip is going to interact with the other chips, how fast the input reaches the output etc. Timing Analysis can be done in both ways; static as well as dynamic.

Dynamic Timing Analysis(DTA) requires a comprehensive set of input vectors to check the timing characteristics of the paths in the design. Basically it determines the full behavior of the circuit for a given set of input vectors. Dynamic simulation can verify the functionality of the design as well as timing requirements. For example if we have 100 inputs then we need 2^{100} simulations to complete the analysis. The amount of analysis is astronomical compared to static analysis.

Static Timing Analysis(STA) checks every path in the design for timing violations without checking the functionality of the design. This way, one can do timing and functional analysis same time but separately. This is faster than dynamic timing simulation because there is no need to generate any kind of test vectors. Thats why STA is the most popular way of doing timing analysis. STA is pessimistic, because all paths in the design may not run always in worst case delay.

In Static Timing Analysis(STA) static delays such as gate delay and net delays are considered in each path and these delays are compared against their required maximum and minimum values. Circuit to be analyzed is broken into different timing paths constituting of gates, flip flops and their interconnections. Each timing path has to process the data within a clock period which is determined by the maximum frequency of operation. Cell delays are available in the corresponding technology libraries. Net delays are calculated based on the Wire Load Models (WLM) or extracted resistance R and capacitance C.

1.1.1 Basic Definitions

1. **Clock:** Define the clock frequency you want your circuit to run at. This clock input controls all the timing in the chip/design.

2. **Rise Time:** It is defined as the time taken for a waveform to rise from 10% to 90% of its steady state value.

3. **Fall time:** It is defined as the time taken for a waveform to fall from 90% to 10% of its steady state value.

4. **Set-up Time:** It is defined as the time the data/signal has to stable before the clock edge.

5. **Hold Time:** It is defined as the time the data/signal has to be stable after the clock edge.

6. **Transition Time:** It is the time taken for the signal to go from one logic level to another logic level.

7. **Input Delay:** This delay is defined as the time taken by the signal to reach the input with respect to the clock.

8. **Output Delay:** This delay is the delay incurred outside the particular block/pin/port with respect to the clock.

9. **Interconnect Delay:** This is delay caused by wires. Interconnect introduces three types of parasitic effects capacitive, resistive, and inductive all of which influence signal integrity and degrade the performance of the circuit.

10. **Clock-Q Delay:** It is the delay from rising edge of the clock to when Q (output) becomes available. It depends on Input Clock transition and Output Load Capacitance.

11. **Clock Latency:** Clock latency means delay between the clock source and the clock pin. This is called as source latency of the clock. Normally it specifies the skew between the clock generation point and the Clock pin.

12. **Clock Skew:** It is defined as the time difference between the clock path reference and the data path reference. The clock path reference is the delay from the main clock to the clock pin and data path reference is the delay from the main clock to the data pin of the same block.

13. **Metastability:** It is a condition caused when the logic level of a signal is in an indeterminate state.

14. **Wire Load Model:** It is nothing but a statistical model. It consists of a table which gives the capacitance and resistance of the net with respect to fan-out.

15. **Critical Path:** The clock speed is normally determined by the slowest path in the design. This is often called as Critical Path.

16. **Delay Calculation of each timing path:** STA calculates the delay along each timing path by determining the Gate delay and Net delay. Gate Delay is defined as the amount of delay from the input to the output of a logic gate. It is calculated based on Input Transition Time and Output Load Capacitance. Net Delay is defined as amount of delay from the output of a gate to the input of the next gate in a timing path. It depends on the Parasitic Capacitance and Resistance of net.

1.2 Brief Introduction About PrimeTime

PrimeTime is the Synopsys stand-alone full chip, gate-level static timing analyzer. It analyzes the timing of large, synchronous, digital ASICs. PrimeTime works with designs at the gate level, providing a tight integration with other Synopsys tools. It determines whether the design works at the required speed. PT analyzes the timing delays in the design and flags violation that must be corrected. PT, similar to DC, provides a GUI interface along with the command-line interface. Both PT and DC have consistent commands, generate similar reports, and support common file formats.

PrimeTime performs the timing analysis in the following way

1. PrimeTime breaks the design down into a set of timing paths.
2. Calculates the propagation delay along each path. The different kinds of paths when checking the timing of a design are as follows.
   (a) Input pin/port $\rightarrow$ Sequential Element(Eg. Flipflops)
   (b) Sequential Element $\rightarrow$ Sequential Element
   (c) Sequential Element $\rightarrow$ Output pin/port
   (d) Input pin/port $\rightarrow$ Output pin/port
3. Checks for timing violations (depending on the constraints e.g. clock) on the different paths and also at the input/output interface.

For some more detailed information on paths, see the **TimingPath.pdf** posted in **Tutorials** folder.

2 Setting up Synopsys PrimeTime

1. Before starting the simulation, make sure that you are in correct directory.

   In this tutorial, the current directory is **/grad/aljubouri/ECE524/Prime**.

2. To initialize synopsys,

   Type: **synopsys** and click **enter** in the terminal.

   The terminal is changed to synopsys mode.

   Note: Synopsys should be initialized before using synopsys tools, whenever a new terminal is opened.

3. To check whether the synopsys Design Compiler is properly setup or not,

   Type: **which primetime** and click **enter**.

   If the terminal shows the path as **/synopsys/PT/bin/primetime**, then Primetime is properly setup and ready for use.

4. In order to know how to work on Synopsys PrimeTime, in this tutorial we are using **s27_syn.v**. This input file(**s27_syn.v**) is the output file created by the **Synopsys Design Compiler** which has no timing violations by using **s27.v**(design file) and **s27.tcl**(Dc commands).

2.1 To invoke PrimeTime

PrimeTime has both the command-line interface and a graphical user interface (GUI). Both provide the same synthesis functions. The tool takes synthesized description of Verilog and VHDL designs and perform timing analysis.

To launch the PrimeTime in the terminal, Type: **primetime** and click **enter**. This command starts PrimeTime (GUI) as shown in Figure 1. In the terminal, the path will be changed to **pt_shell**>. **Note: Don't use & in the end of the command**. The window has four areas:

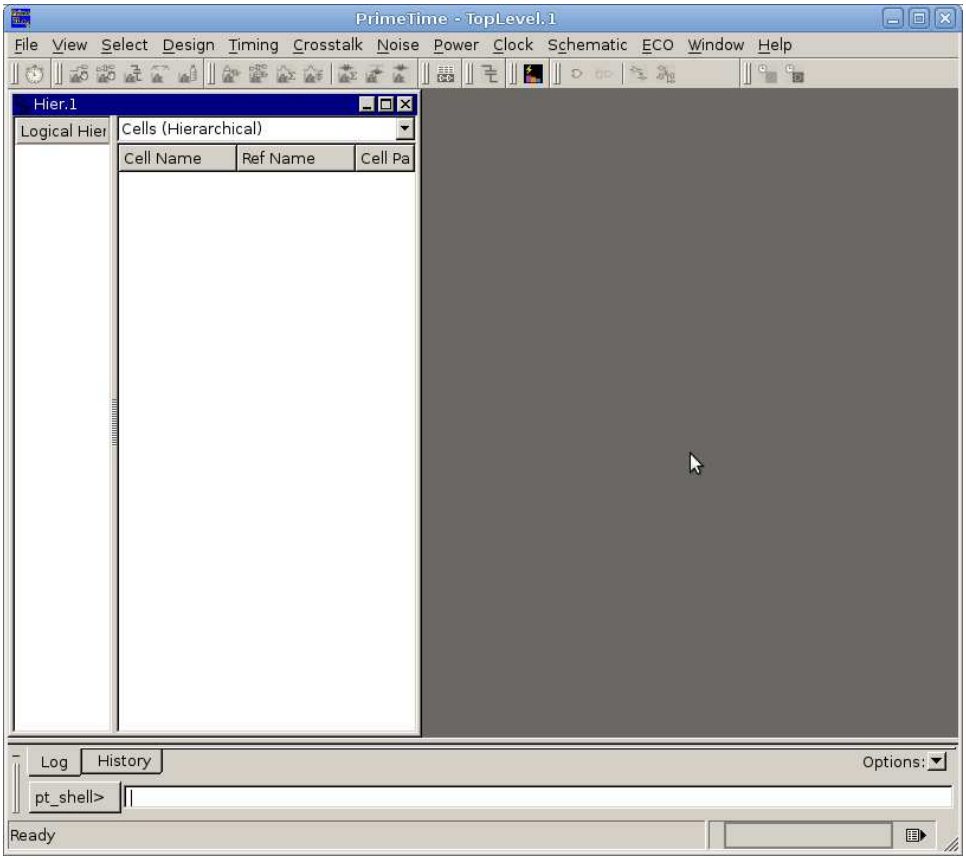

Figure 1. Initial GUI

1. The Left and the area adjacent to it are used it see the logical and cell hierarchy of the given design.

2. Schematic viewer (right) displays the design in schematic form. Unlike other GUI interfaces, this GUI is the initial window from which *PrimeTime* was launched that why it had to be launched in the foreground with out the *ampersand*.

3. Console(below) shows the messages same as in the terminal. Instead of the terminal, commands can be executed from the white space with a tab **pt_shell**> below the Console.

2.2 Setting the library of standard cells

There are many technology libraries available in synopsys. In this tutorial, we use the same **SAED 90nm** physical and timing libraries used in Design Compiler. *Primetime* needs three parameters to set the library.

1. **search_path**: This parameter is used to specify the synthesis tool that it should search this paths when looking for a technology library during timing analysis. To do so

   Type: **set search_path "/synopsys/GPDK/SAED_EDK90nm/Digital_Standard_Cell_Library/synopsys/models"** and click **enter**.

2. **link_library**: This parameter points to the library that contains information on the logic gates in the synthesis technology library. The tool uses this library solely for reference. To do so

   Type: **set link_library " * saed90nm_typ.db saed90nm_min.db saed90nm_max.db"** and click **enter**. The $*$ denotes the current directory.

3. **target_library**: This parameter specifies the file that contains all the logic cells that should used for mapping during timing analysis. In other words, the tool during timing analysis maps a design to the logic cells present in this library.

   Type: **set target_library "saed90nm_max.db"** and click **enter**.

   Note: The target library for the Primetime should be the same as used in Design Compiler.

The above commands will set the search path and library of standard cells to be used with the design. To know about all the commands used in PT, in the GUI *Right click* $\rightarrow$ *Help* $\rightarrow$ *Man Pages*.

2.3 Loading the synthesized Design File

To load any VHDL or Verilog file in PT,

Type: ***read_vhdl*** **or** ***read_verilog*** {<***filenamewithpath***>} and click **enter**.

In this example to load *s27_syn.v*,

Type: **read_verilog** {**/grad/aljubouri/ECE524/Prime/s27_syn.v**} and click **enter**.

This command will load the *s27_syn.v* file into **PT**. Since the input file contains many sub designs in it, we need the set TOP module of the design. To do so

Type: **current_design** <***topmodule***> and click **enter**.

In this example, s27 is the top module in *s27_syn.v* with three flip flops as it sub designs.

Type: **current_design *s27*** and click **enter**. This step will link the design to the specified target library.

Then we need to set wire model, which calculates the Interconnect wiring and transition delays using the wire load model defined in the target library. This wire model is to be same as used in *Design Compiler*. To do so

Type: **set_wire_load_model −name 8000 −library saed90nm_max** and click **enter**.

Generally during STA many numeric values are used. These numeric values can be declared as variables using the command set.

Type: **set load 10** and click **enter**. Type: **set intran 100** and click **enter**. These two command declare two variables *load* and *intran* for numeric values 10 and 100.

Similarly, strings are also declared as a variable.

Type: **set drivcell "INVX0"** and click **enter**. This command declares the *libcell INVX0 as drivcell*. The variables will be used with prefix of $.

After setting the top module, to see schematic diagram, *click* → *Schematic* → *New Design Schematic View*. The net-list of the design can be seen in the GUI window like as shown in Figure 2.

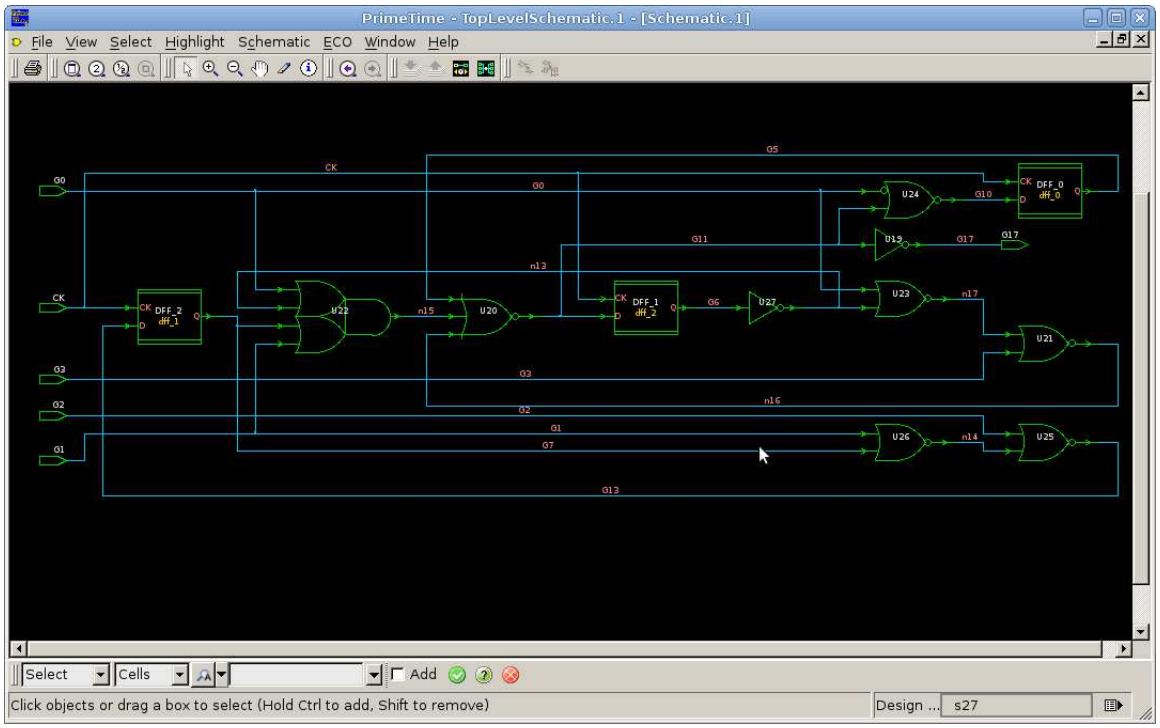

Figure 2. Schematic View

2.4 Static Timing analysis using PrimeTime

PrimeTime is used for static timing analysis. The input file to PT is the synthesized file generated by Synopsys Design Compiler. In Design Compiler, the design is synthesized with timing constraints such as clock period ,input delay and output delay. PrimeTime does not produce any synthesized file, it checks the synthesized design meet the environmental time constraints or not. So we need to same constraints used in PrimeTime also.

The following steps needs to be done in the order specified to to do static timing analysis with simple constraints using **PT**. The clock should be specified prior to any other attributes because all other timing constraints can be provided only with respect to the clock.

create_clock: This command is used to define a clock object with a particular period and waveform.

The **period** option defines the clock period, while the **waveform** option controls the duty cycle and the starting edge of the clock. This command is applied to a pin or port, object types.

Example: create_clock −period 40 −waveform {0 20} {CK}
This command means that a port named **CK** of the design s27 is of type **clock** that has a period of *40 ns*, with 50% duty cycle. The positive edge of the **CK** starts at time *0 ns*, with the falling edge occurring at *20 ns*. By changing the falling edge value, the duty cycle of the **CK** may be altered. This **clock** period value should be same as used in **DC**.

There are many timing constraints can be applied to a design during timing analysis in **PT**, we see just some of them.

1. **set_input_delay**: It specifies the input arrival time of a signal in relation to the clock. It is used at the input ports, to specify the time it takes for the data to be stable after the clock edge. The following two commands tells the minimum and maximum delay for input ports.

   Example: set_input_delay 0.1 −min −clock CK [remove_from_collection [all_inputs] [get_port CK]] and

   Example: set_input_delay 0.4 −max −clock CK [remove_from_collection [all_inputs] [get_port CK]]

   This command means that the data in all input ports except CK port takes *0.1 to 0.4 ns* to become stable after change in the **CK**.

2. **set_output_delay**: This command is used at the output port, to define the time it takes for the data to be available before the clock edge.

   Example: set_output_delay 0.4 −max −clock CK [all_outputs] and

   Example: set_output_delay 0.1 −min −clock CK [all_outputs]

   This command means that the data in the all output ports available *0.1 to 0.4 ns* before the **CK**.

3. **timing_slew_propagation_mode**: This variable specifies how slew is to be propagated through the circuit. Allowed values are worst_slew (the default), which causes Prime Time to propagate the worst slew selected from the inputs.

   Example: set timing_slew_propagation_mode worst_slew

4. **timing_report_unconstrained_paths**: Specifies if PrimeTime searches for unconstrained paths or not when you use the **report_timing** command. The default is false. To know the unconstrained paths, we need to set it to true.

   Example: set timing_report_unconstrained_paths true.

5. **power_enable_analysis**: Enables or disables PrimeTime, which provides power analysis. The default is false. To do the power analysis, we need to set it to true.

   Example: set power_enable_analysis true.

6. **clock_transition**: Specifies transition time of register clock pins.

   Example: set_clock_transition 0.3 [get_clocks CK]. This example specifies a rise transition time of 0.3 and fall transition time of 0.3 for register clock pins clocked by **CK**.

7. **clock_latency**: Specifies latency of clock network.

   Example: set_clock_latency 0.25 [get_clocks CK] . Refer the definition of clock latency.

8. **clock_uncertainty**: Specifies the uncertainty (skew) of specified clock networks.

   Example: set_clock_uncertainty −setup 0.3 [get_clocks CK] and

   Example: set_clock_uncertainty −hold 0.2 [get_clocks CK]

   These two command specify that the these skew values are used only during setup and hold time calculation only.

9. **Driving strength**: The given design may be used as a sub-design in a large circuit and hence it is driven by some unknown circuit. Hence during timing analysis, we assume the input ports of the given design is driven by some circuit.

   Example: set_driving_cell -lib_cell $drivcell −input_transition_rise $intran −input_transition_fall $intran [remove_from_collection [all_inputs] [get_port CK]]

   This command indicates that the input ports of s27 are driven by the lib cell "INVX0" with rising and falling transition value equal to $intran.

10. **Output load**: The given design may be used as a sub-design in a large circuit and hence it is going to drive(act as input) to some unknown circuit. Hence during timing analysis, we assume the output ports of the given design has some load value(input capacitance of unknown circuit).

    Example: set_load $load [all_outputs]

    This command indicates that the output ports of s27 are going to drive some unknown circuit whose input capacitance value equal to $load.

These are some timing and environmental constraints applied to the design for static timing analysis. Then the timing analysis is done by the following two commands.
Example: check_timing and
Example: update_timing

2.5 Redirecting Reports, Inspection of Results, and removing the Designs

To write the *constraints report* of the current design to a file called *constr.rpt*
Type, **report_constraint > constr.rpt** and click **enter**. This file indicates whether the constraints met or not.

To write the *transition time* of the current design to a file called *tran.rpt*

Type, **report_timing -justify -transition_time -delay min_max -capacitance -nets -input_pins -from [all_registers -clock_pins] -to [all_registers -data_pins]> tran.rpt** and click **enter**.

This file reports all transition time of each path. When a signal takes too long transiting from one logic level to another, a transition violation is reported. This violation is a function of the node resistance and capacitance and it can be removed by increasing the drive strength of the circuit or by buffering the some of the fan-out paths to reduce the capacitance seen by the output pin. The capacitance on a node is a combination of the fan-out of the output pin and the capacitance of the net. This check ensures that the device does not drive more capacitance than the device is characterized for.

To write the *timing report* of the four different paths(see section 1.2) of the current design to separate files is done by the following four commands.

Type, **report_timing -justify -from [all_inputs] -max_paths 20 -to [all_registers -data_pins] -delay_type min_max > timingid.rpt** and click **enter**.

Type, **report_timing -justify -from [all_register -clock_pins] -max_paths 20 -to [all_registers -data_pins] -delay_type min_max > timingqd.rpt** and click **enter**.

Type, **report_timing -justify -from [all_registers -clock_pins] -max_paths 20 -to [all_outputs] -delay_type min_max > timingqo.rpt** and click **enter**.

Type, **report_timing -justify -from [all_inputs] -to [all_outputs] -max_paths 20 -delay_type min_max > timingio.rpt** and click **enter**.

When PrimeTime analyzes a design, it is checking that each path meets the setup and hold specification for the design library that you specify. PrimeTime will time every path twice to ensure that path does not cause a setup or hold violation whether the path is rising or falling. Setup violations happen when data changes less than t_{Setup} nanoseconds before the rising edge of the clock. Hold violations are similar to setup violations but data changes less than t_{Hold} after the rising edge of the clock. There is a window around every rising clock edge that has a width of (t_{Setup}+ t_{Hold}) where the data can not change. Changing the data inside this window will cause metastability inside of the flip flop.

Every path in the design is a register to registers, even input and output paths. The time through the combination logic(T_{logic}) must be less than one clock period(Cp) minus the clock uncertainty(Cu), flip flop setup time(ff_{Setup}), and time it takes to transfer the data to the Q pin(ff_{C2q}) when clock goes high. Clock uncertainty counts for clock skew (phase difference) between the launching flip flop and the capturing flip flop. i.e, $T_{logic} < Cp$ - (ff_{C2q} - Cu - ff_{Setup}). Calculating T_{logic} is very simple. The delay though the cell is added to the time thought the net.

Setup violations are essentially where the data path is too slow compared to the clock speed at the capture latch. With that in mind there are several things a designer can do to fix the setup violations such as reduce the amount of buffering in the path, replace buffers with 2 inverters place farther apart, reduce larger than normal capacitance on a on output pin.

Hold violations have the opposite problem of setup. Hold violations are caused when the data is not held long enough at the launch flip flop to ensure that data was clocked in completely. To state to problem differently, hold violations are caused when data is too fast when compared to the clock speed. To fix hold violations in the design, the designer needs to simply add more delay to the data path. This can be done by adding buffers/inverter pairs/delay cells to the data path and add more capacitance to the output pin of books with light capacitance.

Reporting True or False Paths: When reporting a timing path, the **report_timing -justify** option uses a justification process to check whether the reported path is a true path or a false path. True path justification takes into account the logic function of the design and tries to find a set of input vectors that can sensitize the reported path. If PrimeTime cannot find an input vector, it considers the path a false path. If it finds an input vector, it considers the path a true path and shows the input vector at the end of the report.

2.5.1 Constraints Checking

Once the path delays are calculated, PT, checks the delays against the constraints provided to it. For example it check the setup or hold timing violations. The amount of time by which a violation is avoided is called the *slack*. For example, for a setup time constraint, if a signal must reach a cell input at no later than $8ns$ and is determined to arrive at $5ns$, the slack is $3ns$. A slack of 0 means that the constraint is just barely satisfied. A negative slack indicates a *timing violation*.

2.5.2 Removing the Designs

If you want to remove any designs from your *PT* memory.

Type, **remove_design -all** and click **enter**. This command will remove the current design. Since the design$(s27)$ contains sub modules, we need to use the option **-all** to remove them.