Efinity® Timing Closure User Guide

UG-EFN-TIMING-v1.0
June 2020
www.efinixinc.com

Contents

Introduction

You have created your RTL, you have designed your board, now you need to close timing, and you are stuck. Knowing what to do for that last bit of tweaking to achieve your desired f_{MAX} can be difficult. When creating the Efinity® software, Efinix® software engineers have chosen default values for the tool flow to achieve the best trade-off between performance and runtime for a large number of benchmark designs. Your design, however, is unique, and may benefit from non-default settings to get the performance you need.

The goal of this user guide is to give you some tools and tricks to close timing more quickly. The document describes how to change options in synthesis, placement, and routing to customize the Efinity® flow for your particular design, how to use SDC files to set constraints, and how to use the Tcl console to view reports.

Exploring Timing

You use static timing analysis (STA) to measure the timing performance of your design. The software generates a timing report based on the design's place and route results and the project's SDC file. The software provides several tools for viewing and cross-probing timing results:

- The Timing Browser helps you explore your design's critical paths and the cells of those paths.
- The Floorplan Editor shows the locations of the paths and cells in the fabric.
- The Tcl Command Console helps you analyze and explore timing.

After analyzing your design's timing, you can update your SDC file if needed. To apply the new SDC settings to see how they affect placement and routing, re-run the flow from synthesis to the end.

Note: Refer to SDC Constraints (Alphabetical) on page 11 for a list of supported SDC constraints and object specifiers.

To explore timing, open the Tcl Command Console, the Timing Browser, and the Floorplan Editor. In the Tcl Command Console, enter commands to query timing reports. The software reports specific timing paths based on their slack or propagation delay. For example:

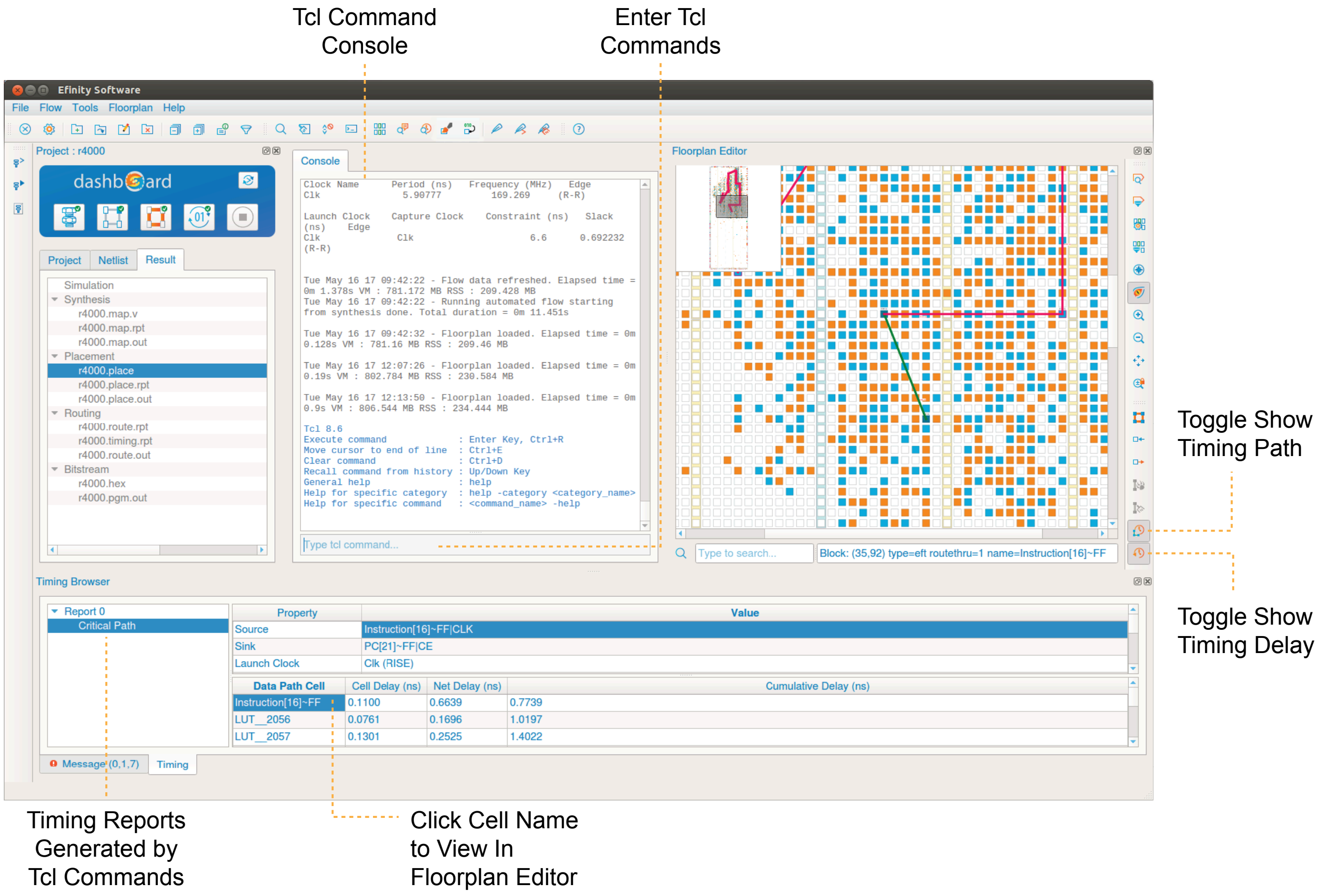
- Use the `report_path` command to query propagation delays from the source to the destination.
- Use the `report_timing` command to query the slack of a specific timing arc.

Specify the details of the timing arc that you want to analyze. You can specify the starting and ending arc point types explicitly or leave them as implicit. The software analyzes the timing arc based on the arguments provided to the constraints.

The software displays Tcl reports in the Timing Browser.

- Click on the report name to view details.
- Click on cell names under Data Path Cell to view the location of the cell in the Floorplan Editor.
- Turn on Show Timing Path or Show Timing Delay in the Floorplan Editor to see the path and delay for a particular cell.

Figure 1: Using the Timing Browser



```
Clock Name      Period (ns)   Frequency (MHz)   Edge
Clk               5.90777         169.269      (R-R)

Launch Clock    Capture Clock    Constraint (ns)   Slack
(ns)    Edge
Clk              Clk                       6.6      0.692232
(R-R)


Tue May 16 17 09:42:22 - Flow data refreshed. Elapsed time =
0m 1.378s VM : 781.172 MB RSS : 209.428 MB
Tue May 16 17 09:42:22 - Running automated flow starting
from synthesis done. Total duration = 0m 11.451s

Tue May 16 17 09:42:32 - Floorplan loaded. Elapsed time = 0m
0.128s VM : 781.16 MB RSS : 209.46 MB

Tue May 16 17 12:07:26 - Floorplan loaded. Elapsed time = 0m
0.19s VM : 802.784 MB RSS : 230.584 MB

Tue May 16 17 12:13:50 - Floorplan loaded. Elapsed time = 0m
0.9s VM : 806.544 MB RSS : 234.444 MB

Tcl 8.6
Execute command              : Enter Key, Ctrl+R
Move cursor to end of line   : Ctrl+E
Clear command                : Ctrl+D
Recall command from history : Up/Down Key
General help                 : help
Help for specific category   : help -category <category_name>
Help for specific command    : <command_name> -help
```

Block: (35,92) type=eft routethru=1 name=Instruction[16]~FF

Property	Value
Source	Instruction[16]~FF|CLK
Sink	PC[21]~FF|CE
Launch Clock	Clk (RISE)

Data Path Cell	Cell Delay (ns)	Net Delay (ns)	Cumulative Delay (ns)
Instruction[16]~FF	0.1100	0.6639	0.7739
LUT__2056	0.0761	0.1696	1.0197
LUT__2057	0.1301	0.2525	1.4022

Note: Refer to Tcl Timing Report and Flow Commands on page 16 for more information on available commands. For help on available Tcl commands, type `help -category <sdc or timing>` in the Tcl Command Console.

Methods for Closing Timing

Start at the beginning! In general, it is best to start by choosing good synthesis options, then placement options, then routing options. First choose high-level options that work well, then run a seed sweep using those options to take advantage of noise and use the best result.

Synthesis Options

Changes in synthesis results may or may not help you achieve your final f_{MAX} target. Usually it is best to use these options with different place and route options as part of your design exploration.

Table 1: Synthesis Options

Name	Choices	Description
--mode	speed, area	speed: Optimizes for fastest f_{MAX} (default). area: Optimizes for smallest area.
--max_ram	-1, 0, *n*	-1: Default. There is no limit to the number of RAM blocks to infer. 0: Disable. *n*: Any integer.
--max_mult	-1, 0, *n*	-1: Default. There is no limit to the number of multipliers to infer. 0: Disable. *n*: Any integer.
--infer-clk-enable	0, 1, 2, 3	Infer flip-flop clock enables from control logic. 0: disable. 1: Low effort. 2: Medium effort. 3: Default. High effort.
--infer-sync-set-reset	0, 1	Infer synchronous set/reset signals. 0: Disable. 1: Default. Enable.
--fanout-limit	0 to *n*	If something is high fanout, the tool duplicates the fanout source. 0: Default. Disable. *n*: Indicate the fanout limit at which to begin duplication.
--seq_opt	0, 1	Turn on sequential optimization. This option can reduce LUT usage but may impact f_{MAX}. 0: Default. Disable. 1: Enable.

Place-and-Route Options

The best place to start with adjusting the place-and-route settings is to use one of the six optimization levels. These are extra effort settings that control both placement and routing and Efinix developed them to introduce as much useful variation as possible. These options will not help all designs, and often the default settings are actually the best choice.

Note: Using these options can cause **significantly higher run-time**. In fact, some options trigger completely different optimization algorithms than the standard flow.

Try running all of these optimization levels and choose the one that works best for your design. The timing values are best for designs that are easy to route while the congestion values are best for designs that are very difficult to route. For congested designs, these options resolve congestion early in the process so that the router can focus on meeting timing. If the number of routing iterations is greater than 20, the design is hard to route.

The `anneal_random_seed` option introduces random noise in the placer. The f_{MAX} difference between the best and worst seed in a 5-seed sweep can be up to 10%. But keep in mind that a "good seed" only applies to one specific design for one Efinity® release on one operating system. So if you want to reproduce the same result for the same design, you need to use the same software release and same operating system.

Sometimes small design changes may appear to reduce f_{MAX} by up to 10%. You should run a 5-seed sweep to verify that the decrease was actually due to the design change and not simply noise from the placer.

Table 2: Optimization Options

Optimization	Value	Description
--optimization_level	NULL	Disabled (default).
	TIMING_1	Low effort to meet timing for a non-congested design.
	TIMING_2	Medium effort to meet timing for a non-congested design.
	TIMING_3	High effort to meet timing for a non-congested design.
	CONGESTION_1	Low effort to meet timing and help a congested design route.
	CONGESTION_2	Medium effort to meet timing and help a congested design route.
	CONGESTION_3	High effort to meet timing and help a congested design route.
--qp_options=anneal_random_seed	Integer	Enter any integer to insert a random seed into the place-and-route algorithm. Read more about seed sweeping in the next section.

Seed Sweeping

After you choose an appropriate optimization level (or the default), try running a seed sweep and taking the best of these runs to close timing. The annealer uses a random number generator that you can control using a seed parameter (`--qp_options=anneal_random_seed`). Running a 10-seed sweep typically results in an f_{MAX} variation of 10-20% in a well-behaved circuit but can be higher for random differences that only occur in one seed. Keep these tips in mind:

- Run a seed sweep every time you make a small change to the design.
- No seed value is better than any other
- Different seed values may be best on different machines

Efinix provides a helper script, **efx_run_pnr_sweep.py**, in the **scripts** directory that you can use to compile a design multiple times using various settings. The software summarizes the timing results of the runs in the **timing.sum.rpt** file. You can find the corresponding result files in the **run_sweep_**<*number*>/ directory.

The script uses this syntax:

```
efx_run_pnr_sweep.py [project XML file] {<sweep_seed>, <sweep_opt_levels>}
    [-h]
```

where:
- *project XML file* is your project's XML file
- <*sweep_seed*> is the seed number (using `--num_seeds` <*number*>) or a range (using `--start_seed` <*number*> and `--end_seed` <*number*>)
- <*sweep_opt_levels*> is a flag to sweep all of the optimization levels
- `-h, --help` shows the help

The following examples show how you can use the seed sweep script with the helloworld design.

Sweep seeds using the default settings. In this mode, the script runs 10 different seeds, from 0 to 9.

```
efx_run_pnr_sweep.py helloworld.xml sweep_seeds
```

Compile with 6 seeds:

```
efx_run_pnr_sweep.py helloworld.xml sweep_seeds --num_seeds 6
```

Compile with seeds that start from 3 to 5:

```
efx_run_pnr_sweep.py helloworld.xml sweep_seeds --start_seed 3 --end_seed 5
```

Compile with 6 seeds that start from 3:

```
efx_run_pnr_sweep.py helloworld.xml sweep_seeds --start_seed 3 --num_seeds 6
```

Sweep all of the optimization levels:

```
efx_run_pnr_sweep.py helloworld.xml sweep_opt_levels
```

SDC Tips and Tricks

The following sections contain a variety of tips for working with SDC files.

SDC Syntax

In SDC syntax:
- # starts a comment; remaining text on this line is ignored.
- \ at the end of a line indicates that a command wraps to the next line.

Set Timing Constraints

The order of the constraints specified in the SDC file is important. If there are dependencies among the different constraints specified, you must ensure that you have written them in the correct order for them to be valid. Additionally, if the specified constraint has incorrect syntax, the software ignores it and issues a warning message. For some constraints, the argument order is important for the constraint to be valid.

By default, if you specify no constraints, the software creates a virtual clock with period 1 ns and does not constrain any I/O pins. The Efinity® timing analyzer then identifies the critical path based on this default constraint. The minimum constraint required in an SDC file is the `create_clock` constraint. You should always set a clock constraint—even if it is a virtual clock—whenever you specify an SDC file.

The software assumes that all clocks are related (by default) and it analyzes all cross domain transfers that it finds.

- If you want to control the clock relationships, use the **set_clock_groups** constraint.
- Use relevant SDC constraints to specify other exceptions, such as **set_false_path** and **set_multicycle_path**.

Constraining Clocks and Associated I/O Pins

You have I/O pins that are associated with a clock. You set constraints to define the clock and any relationships it has to internally generate clocks. Then, you constrain the I/O pins relative to that clock.

Defining Clocks

You define and identify clocks using the `create_clock` and `create_generated_clock` constraints.

The `create_clock` constraint defines a real or virtual clock with specific duty cycle and period (ns). The Efinity® software does not support the -add option. Each target can only have one clock associated with it.

The `create_generated_clock` constraint defines a relationship between an internally generated clock and its source clock. This constraint only supports the `divide_by`, `multiply_by`, `duty_cycle`, `invert` options. Typically, you use this constraint to capture a PLL clock output's relationship to its input clock.

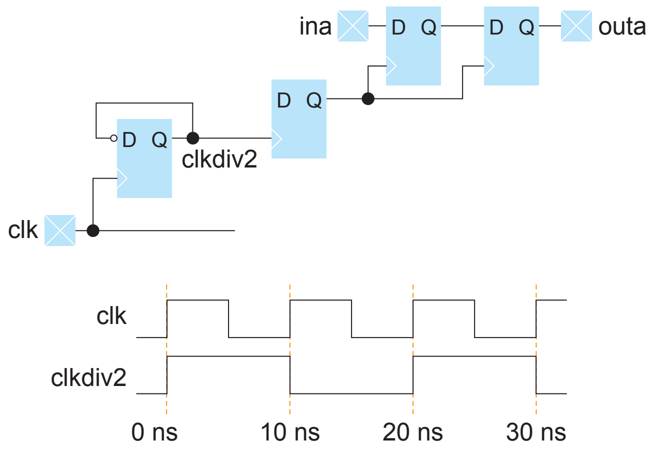
Constraining I/O Pins

You should constrain I/O pins to be timing-equivalent to a register that is clocked with the real or virtual clock name you defined. Then, use the set_input_delay and set_output_delay constraints.

Example

In this example, `clk` is divided to become `clkdiv2`, which feeds the `ina` and `outa` pins.

Figure 2: Clock and I/O Pin Constraint Example



Use these constaints to define the clock and set the delays for the pins:

```
create_clock -name clk -period 10 [get_ports clk]
create_generated_clock -source clk -divide_by 2 clkdiv2
set_input_delay -clock clkdiv2 -max 2.4 [get_ports ina]
set_output_delay -clock clkdiv2 -max 1.2 [get_ports outa]
```

Unrelated Clocks

If you set constraints for two clocks, and do not cut the path between them, the software tries to see if they are actually related. If the timer cannot find a common clock period for the two clocks after 1,000 clock cycles, it determines that they are *non-expandable*, and are unrelated.

The timer gives these clocks a default constraint of 0.01 ns. If you want to override this default, use the set_max_delay or set_min_delay constraint.

Wildcard Commands

An `*` indicates a wildcard. Use `*` by itself to match all signals, or use it to create a partial wildcard. For example `clk*` would match `clk` and `clk2`.

Example: Constraining with Wildcards

You want to constrain all `Oled` signals with respect to clock `clk`. The resulting constraints are:

```
set_input_delay -max 10.214 -clock clk Oled*
set_input_delay -min 3.607 -clock clk Oled*
```

Regular Expressions

You can use regular expressions (in the Perl regular expression format) with the object specifier. You must encapsulate the object specifiers in square brackets []; arguments must be enclosed in curly braces {}.

To use Perl regular expressions, include the `-regexp` option in your command. Escape Perl regular expression characters if the provided string argument contains those characters. For example:

Simple wildcard:

```
get_pins y_r[*]~FF|D
```

Using Perl regular expressions:

```
regexp get_pins -regexp { y_r\[.*\]~FF|D }
```

Inverted Clocks

For an inverted external clock (one that uses the negative edge), include the `clock_fall` option in your `set_input_delay` or `set_output_dalay` command. For example:

```
set_output_delay -clock <clock> -clock_fall -max -3.1 <ports>
set_output_delay -clock <clock> -clock_fall -min -2.85 <ports>
```

SDC Constraints (Alphabetical)

The Efinity software supports the following SDC constraints. Options in square brackets [ ] are optional.

create_clock	set_false_path	set_multicycle_path
create_generated_clock	set_input_delay	set_max_delay
set_clock_groups	set_output_delay	set_min_delay
set_clock_uncertainty		

create_clock Constraint

```
create_clock -period <float> [-waveform {rising_edge falling_edge}]  \
    [-name <clock name>] [<targets>]
```

This command defines a clock with the desired period (in ns) and waveform, and applies it to the target nodes. If you do not specify a target, the software considers the clock to be a virtual clock. You can use the virtual clock to constrain inputs and outputs to a clock external to the design. The tool does not support multiple clock assignments to the same target.

Note: You can refer to netlist clocks using regular expressions.

- `-period` indicates the clock period in ns.
- `-waveform` indicates the rising and falling edges (duty cycle) of the clock as two time values: the first rising edge and the next falling edge. If you omit the waveform option, the command creates a clock with a rising edge at 0 and a falling edge at the half period, which is equivalent to using `-waveform {0 <period/2>}`. Falling edges are not used in analysis, therefore, non-50% and 50% duty cycles behave the same.
- `-name` indicates the clock name. If omitted, the software gives the clock the name of the first target.

If you assign a virtual clock using the `create_clock` command, you must reference it elsewhere in a `set_input_delay` or `set_output_delay` constraint.

create_generated_clock Constraint

```
create_generated_clock -source <source clock object> [-divide_by <factor> | \
    -multiply_by <factor>] [-duty_cycle <percentage>] [-invert] \
    [-name <virtual clock name>] <target>
```

This constraint is useful for designs with internally generated clock signals because it provides more accurate timing analysis. First, use the `create_clock` constraint on the source clock that generates the internal clock signal. Then, use this constraint.

- `-source` is the generated clock's source node.
- `-divide_by` is the division factor.
- `-multiply_by` is the multiplication factor.
- `-duty_cycle` is the duty cycle as a percentage of the clock period.
- `-invert` inverts the clock.
- `-name` is the name of the generated clock.

- <*target*> is the name of the net that implies that it is an internally generated clock signal.

set_clock_groups Constraint

```
set_clock_groups -exclusive -group {<clock>} [-group {<clock>} -group ...]
```

This command instructs the timing analyzer to not analyze paths between one or more specified groups of clock domains in either direction. You can use this command with netlist or virtual clocks in any combination. The `set_clock_groups` constraint is equivalent to a `set_false_path` constraint between the clocks in different groups. For example, the command

- `-exclusive` indicates a mutually exclusive clock
- `-group` indicates a clock list

```
set_clock_groups -exclusive -group {clk} -group {clk2 clk3}
```

is equivalent to

```
set_false_path -from [get_clocks{clk}] -to [get_clocks{clk2 clk3}]
set_false_path -from [get_clocks{clk2 clk3}] -to [get_clocks{clk}]
```

If you specify only one clock group, it cuts all paths to and from the specified clock domain(s) to all others.

set_clock_uncertainty Constraint

```
set_clock_uncertainty [-setup] [-hold] [-from <clock>] [-rise_from <clock>] \
    [-fall_from <clock>] [-to <clock>] [-rise_to <clock>] [-fall_to <clock>] \
    <uncertainty>
```

This constraint specifies the uncertainty for clocks or clock-to-clocks transfers. The tool added the specified uncertainty value to the derived uncertainty. If you do not specify source or destination clocks, the tool applies the uncertainty to all clocks in the design. If you do not specify a setup or hold, the tool uses the uncertainty value for both setup and hold.

- `-setup` is the clock uncertainty for setup analysis
- `-hold` is the clock uncertainty for hold analysis
- `-from` source clock
- `-rise_from` source clock with rising edge
- `-fall_from` source clock with falling edge
- `-to` destination clock
- `-rise_to` destination clock with rising edge
- `-fall_to` destination clock with falling edge
- uncertainty clock uncertainty value

set_false_path Constraint

```
set_false_path [-setup] [-hold] -from <names> -to <names>
```

This command cuts paths unidirectionally:

- between clock domains

- from start or end points

Only one point can be a register or an I/O, not both. False paths are supported between entire clock domains and between individual registers. If you do not specify a setup or hold, the tool cuts both for setup and hold.

Either the `-to` or the `-from` must be entire clock domains.

- `-setup` is the false path for setup analysis
- `-hold` is the false path for hold analysis
- `-to` the clock domain destination, I/O, or register end point
- `-from` the clock domain source, I/O, or register start point
- <*names*> is a clock domain source/destination, I/O, or register start/end point

Note: Use the set_clock_groups constraint if both directions are false paths.

set_input_delay and set_output_delay Constraints

```
set_input_delay -clock <clock> [clock_fall] [-max] [-min] <delay> <ports>
set_output_delay -clock <clock> [clock_fall] [-max] [-min] <delay> <ports>
```

- Use `set_input_delay` to analyze timing paths from input I/Os.
- Use `set_output_delay` for timing paths to output I/Os. If you do not specify these commands in your SDC, paths from and to I/Os will not be analyzed.

These commands constrain each I/O pad specified to be timing-equivalent to a register clocked on the clock specified after `-clock`. This register can be either a clock signal in your design or a virtual clock that does not exist in the design but that you use to specify the I/O timing.

The command also adds <*delay*> through each pad, thereby tightening the time constraint along paths traveling through the I/O pad. You can use this additional delay to model board-level delays. `-max` is the setup constraint, `-min` is the hold constraint; if you specify neither, the tool uses <*delay*> for both max and min.

- `-clock` is the clock name
- `-clock_fall` is the input delay relative to the clock's falling edge
- `-max` is the maximum data arrival time
- -`min` is the minimum data arrival time
- `<delay>` is the delay value
- `<ports>` is the list of input ports

set_max_delay and set_min_delay Constraints

```
set_max_delay -from <clock> -to <clock> <delay>
set_min_delay -from <clock> -to <clock> <delay>
```

These commands override the default timing constraint (calculated using the information from `create_clock`) with a user-specified delay. This constraint may produce unexpected results.

- `-to` destination clock
- `-from` source clock
- `<delay>` is the delay value in ns

set_multicycle_path Constraint

```
set_multicycle_path [-setup] [-start] [-hold] [-end] -from <clock>] \
    -to <clock> <value>
```

This command creates a multicycle at the clock domain level. It adds (<*value*> - 1) times the period of the destination clock to the default setup time constraint. Multicycles are supported between entire clock domains but not between individual registers. If you do not specify `-setup` or `-hold`, the tool applies the constraint to both.

- `-setup` applies the multicycle value to setup analysis
- `-hold` applies the multicycle value to hold analysis
- `-start` the multicycle value is relative to the source clock
- `-end` the multicycle value is relative to the destination clock (default)
- `-to` clock net destination
- `-from` clock net destination
- <*value*> is the multicycle value

Constraint Object Specifiers

Constraints support explicit object specifiers. Implicit naming is implied if you do not use an object specifier with the constraint command. If you do not use an object specifier the software executes the search on the objects in the following order: nets, pins, cells.

The name you provide to the object specifier is based on the post-mapped design name; refer to the generated post-mapped Verilog HDL netlist—autogenerated by the software at the end of synthesis—for these names.

Note: The pipe (|) character is the separator between the instance name and the referenced port name.

- `all_clocks`—Retrieves all of the clocks in the design.
- `all_inputs`—Retrieves all of the input ports in the design.
- `all_registers`—Retrieves all of the register instances in the design.
- all_outputs—Retrieves all of the output ports in the design.
- `get_cells [-regexp] [<filter>]`—Retrieves all design instances that match the specified name or pattern.(1)

(1) By default, you do not need to escape brackets. However, if you use the -regexp option, you must escape all brackets.

- `get_clocks [-regexp] [<`*`filter`*`>]`—Retrieves the clock that matches the specified name or pattern. The tool looks first for the clock name, if it exists. Next, it checks the clock net name (includes virtual clocks).(1)
- `get_nets [-regexp] [<`*`filter`*`>]`—Retrieves the net that matches the specified name or pattern.(1)
- `get_pins [-regexp] [<`*`filter`*`>]`—Retrieve the pins that match the specified name or pattern. The pin name format is <*cell*> | <*port*>. Escape square brackets for cell names; you do not need to escape square brackets for ports if the port has bit indexing.(1)
- `get_ports [-regexp] [<`*`filter`*`>]`—Retrieve the ports that match the specified name or pattern.(1)

Tcl Timing Report and Flow Commands

Use these commands in the Tcl Command Console. See **Exploring Timing** on page 3 for more information.

Timing Commands

The tool supports the following timing commands:

- `delete_timing_results`—This command deletes path data, reported using the report_path and report_timing commands, that the software has stored in memory.
- `get_available_timing_model`—Returns a list of available operating conditions for the current device. The device must be loaded for this command to execute.
- `get_timing_model`—Returns the current operating conditions for the device. The device must be loaded for this command to execute.
- `read_sdc <`*`file`*`>`—This command reads in the specified SDC file. If you do not specify a file, the tool loads the SDC file that you set in the project. The SDC file overrides previous constraints.
- `reset_timing`—Removes the timing data, all constraints, and all reported paths.
- `set_timing_model`—Specify the operating conditions for the current design. Use get_available_timing_model to view the list of supported conditions. You can only use this command after you have performed place and route.
- `write_sdc <`*`file`*`>`—Write the timing constraints in memory to the specified SDC file. The tool does not add the SDC file to your project.

report_clocks Command

```
report_clocks [-file <file>] [-stdout]
```

This command generates a clock report. If you do not specify a file name, the software prints the report to the Console (stdout) by default.

report_path Command

```
report_path [-file <file>] [-npaths <number>] [-nworst <number>] \
    [-show_routing] [-stdout] [-summary] [-through <names>] -to <names> \
    -from <names> -id <number> [-min_path]
```

This constraint reports the longest delay path and the corresponding delay value.

- `-file` writes the results to a file in the **outflow** directory.
- `-npaths` is the maximum number of paths to report. If you do not specify the number of paths, the software only provides the single longest delay path.
- `-nworst` is the maximum number of paths reported for each unique endpoint. Without this option, the number of paths reported for each destination node is restricted by `-npaths` only. If you use this option but not `-npaths`, `-npaths` defaults to the value specified for `-nworst`.
- `-show_routing` displays detailed routing information.
- `-stdout` writes the results to the Console during compilation only. If you re-generate timing, use the `-file` option instead.
- `-summary` generates a table that summarizes the results.
- `-through` restricts analysis to paths that go through specified pins or nets. Paths that are reported can not start before or go beyond a keeper node (register or port); this restriction considers register pins as combinational nodes in the design.
- `-from` and `-to` limit the analysis to specific start and end points. Any node or cell in the design is a valid endpoint.
- `-id` is the position in the Timing Browser where the tool displays the result.
- `-min_path` reports the minimum delay paths.

report_timing Command

```
report_timing [-detail summary|path_only|path_and_clock|full_path] \
    [-file <name>] [-from_clock <names>] -from <names>
    [-fall_from_clock <names>] [-rise_from_clock <names>]
    [-less_than_slack <slack limit>] [-npaths <number>] [-nworst <number>]
    [-show_routing] [-stdout] [-through <names>] -to <names> -to_clock <names>
    [-rise_to_clock <names>] [-fall_to_clock <names>] [-id <number>]
    [-hold] [-setup]
```

This command reports the worst-case paths and their associated risk. The tool displays paths in order of increasing slack.

- `-detail` specifies how much detail is shown in the path report.
- `-file` writes the results to a file in the **outflow** directory.
- `-from_clock` and `-to_clock` are valid source and destination clocks, respectively.
- `-from` and `-to` limit the analysis to specific start and end points. Any node or cell in the design is a valid endpoint.
- `-fall_from_clock` and `-fall_to_clock` are the starting and ending points of the falling edge of the clock domain, respectively.
- `-rise_from_clock` and `-rise_to_clock` are the starting and ending points of the rising edge of the clock domain, respectively.
- `-less_than_slack` displays only those paths with slack less than the specified limit.
- `-npaths` sets the number of paths to report. If you do not specify the number of paths, the software only provides the single longest delay path.
- `-nworst` limits the number of paths reported for each unique endpoint. Without this option, the number of paths reported for each destination node is restricted by the `-npaths` only. If you use this option but not `-npaths`, `-npaths` defaults to the value specified for `-nworst`.
- `-show_routing` displays detailed routing information.
- `-stdout` writes the results to the Console.
- `-through` restricts analysis to paths that go through specified pins or nets. Paths that are reported can not start before or go beyond a keeper node (register or port); this restriction considers register pins as combinational nodes in the design.
- `-id` is the position in the Timing Browser where the tool displays the result. Use `-id` to overwrite existing constraints in the Timing Browser. `0` is reserved for the critical path. If the specified ID number is larger than the existing constraint ID, the tool ignores this option. By default, it returns clock setup paths if you do not specify `-setup` or `-hold`.
- `-setup` reports the clock setup paths.
- `-hold` reports the clock hold paths.

report_timing_summary Command

```
report_timing_summary [-file <file>] [-hold] [-setup]
```

This command performs timing analysis and generates the critical path timing report. The software saves the report as <*project*>**.timing.rpt** by default. To specify a different file name, use the `-file` option. By default, the tool prints setup and hold paths.

- `-file` writes the results to a file in the **outflow** directory.
- `-setup` reports the clock setup paths.
- `-hold` reports the clock hold paths.

Where to Learn More

The Efinity® software includes documentation as PDF user guides and on-line HTML help. This documentation is provided with the software. You can also access the latest versions of PDF documentation in the Support Center:

- Efinity Software User Guide
- Efinity Synthesis User Guide
- Efinity Timing Closure User Guide
- Efinity Software Installation User Guide
- Efinity Trion Tutorial
- Trion Interfaces User Guide
- Efinity Interface Designer Python API
- Efinity Software Quantum Primitives User Guide

In addition to documentation, Efinix field application engineers have created a series of videos to help you learn about aspects of the software. You can view these videos in the Support Center.

Revision History

Table 3: Revision History

Date	Version	Description
June 2020	1.0	Initial release.