APPENDIX

Part Three

Trial Exhibit 1440, pages 116-167

Trial Exhibit 1441, pages 1-70

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793788

rc_receiver_model_mode

Specifies which receiver model type to use for RC delay calculation.

TYPE

string

DEFAULT

advanced

DESCRIPTION

PrimeTime supports two types of receiver models for RC delay calculation, basic and advanced. The basic model is a single capacitance dependent only on the rise, fall, min, or max arc condition. The advanced model is a voltage-dependent capacitance additionally dependent on input-slew and output capacitance. The advanced model has many advantages, one of which is that the accuracy of **both** delays *and* slews is improved. Another advantage is that nonlinearities such as the Miller effect are addressed. The advanced receiver model is part of the Synopsys Composite Current-Source (CCS) model.

The advanced receiver model will only be used when the network is driven by the advanced driver model; please see the man page on the **rc_driver_model_mode** shell variable for more information.

When the shell variable **rc_receiver_model_mode** is set to *basic*, RC delay calculation will always use the pin capacitances specified in the design libraries. When set to *advanced*, RC delay calculation will use the advanced receiver model if data for it is present *and* if the network is driven by the advanced driver model. The **report_delay_calculation** command used on a network arc will show the message "Advanced receiver-modeling used" as appropriate.

To determine the current value of this variable, enter the following command:

pt_shell **printvar rc_receiver_model_mode**

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012952

rc_receiver_model_mode

Persistent parameter that controls whether the basic or the advanced CCS model is used for input pins driven by nets.l

Syntax:

```
rc_receiver_model_mode basic | advanced
```

where the values have the following meaning:

basic	Use the simplified CCS model for selected net receivers if the ccsd_auto_switch parameter is enabled; Otherwise, use no CCS model.
advanced	Use the advanced CCS model for all net receivers regardless of the ccsd_auto_switch parameter setting.

The default value is *basic*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793819

si_enable_analysis

Enables or disables PrimeTime-SI, which provides crosstalk analysis.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

When true, enables PrimeTime-SI, so that the crosstalk-aware timing calculation mode is used by **update_timing** and **report_timing**. By default, PrimeTime-Si is disabled; this variable is set to false.

If you set this variable to true and enable PrimeTime-SI, you must also do the following:

1. Obtain a PrimeTime-SI license. You cannot use PrimeTime-SI without a license.

2. Use **read_parasitics -keep_capacitive_coupling** to read in the coupling parasitics for your design. PrimeTime-SI is useful only if the design has coupling parasitics data.

For complete information about PrimeTime-SI, see the *PrimeTime Signal Integrity User Guide*.

To determine the current value of this variable, type **printvar si_enable_analysis**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012953

si_enable_analysis

Persistent parameter that controls whether to calculate the noise, and the impact of noise on timing.

Syntax:

```
si_enable_analysis true | false
```

where the values have the following meaning:

true	Perform a noise analysis and take the impact noise has on timing into account during timing analysis.
false	Ignore the effects of noise during timing analysis.

The default value is *false*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793820

si_filter_accum_aggr_noise_peak_ratio

Specifies the threshold for the accumulated voltage bumps introduced by aggressors at a victim node, divided by Vcc, below which aggressor nets can be filtered out during electrical filtering.

TYPE

float

DEFAULT

0.03

DESCRIPTION

Specifies the threshold for the accumulated voltage bumps introduced by aggressors at a victim node; the default is 0.03. This variable, along with **si_filter_per_aggr_noise_peak_ratio**, makes up a pair of variables used by PrimeTime-SI during the electrical filtering phase, to determine whether an aggressor net can be filtered.

An aggressor net, along with its coupling capacitors, is filtered when either of the following are true:

1. The peak voltage of the voltage bump induced on the victim net divided by Vcc is less than the value of **si_filter_per_aggr_noise_peak_ratio**.

2. The accumulated peak voltage of voltage bumps induced on the victim by aggressor to the victim net divided by Vcc is less than the value of **si_filter_accum_aggr_noise_peak_ratio**.

To determine the current value of this variable, type **printvar si_filter_accum_aggr_noise_peak_ratio**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012982

filter_accum_aggr_noise_peak_ratio

Persistent parameter that specifies the aggregated noise peak voltage threshold below which all aggressors on a net are ignored. The aggregated noise peak contains the contributions of all aggressors. The threshold is specified as a fraction of the supply voltage.

Syntax:

```
filter_accum_aggr_noise_peak_ratio ratio
```

where *ratio* is the minimum aggregated noise peak value as a fraction of the supply voltage below which the noise from all aggressors on a net are ignored.

The default value is *0.030*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793821

si_filter_per_aggr_noise_peak_ratio

Specifies the threshold for the voltage bump introduced by an aggressor at a victim node, divided by Vcc, below which the aggressor net can be filtered out during electrical filtering.

TYPE

float

DEFAULT

0.01

DESCRIPTION

Specifies the threshold for the voltage bump introduced by an aggressor at a victim node; the default is 0.01. This variable, along with **si_filter_accum_aggr_noise_peak_ratio**, makes up a pair of variables used by PrimeTime-SI during the electrical filtering phase, to determine whether an aggressor net can be filtered.

An aggressor net, along with its coupling capacitors, is filtered when either of the following are true:

1. The peak voltage of the voltage bump induced on the victim net divided by Vcc is less than the value of **si_filter_per_aggr_noise_peak_ratio**.

2. The accumulated peak voltage of voltage bumps induced on the victim by aggressors to the victim net divided by Vcc is less than the value of **si_filter_accum_aggr_noise_peak_ratio**.

Parasitic filtering criteria previously checked are controlled by the variables **si_filter_per_aggr_xcap**, **si_filter_per_aggr_xcap_to_gcap_ratio**, and **fi_filter_single_to_average_all_xcap_ratio**.

To determine the current value of this variable, type **printvar si_filter_per_aggr_noise_peak_ratio**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012983

filter_per_aggr_noise_peak_ratio

Persistent parameter that specifies the noise peak voltage threshold below which an aggressor is ignored. The threshold is specified as a fraction of the supply voltage.

Syntax:

```
filter_per_aggr_noise_peak_ratio ratio
```

where *ratio* is the minimum noise peak value as a fraction of the supply voltage below which the noise peak is ignored.

The default value is *1.000e-02*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793833

si_noise_composite_aggr_mode

Specifies the composite aggressor mode for noise analysis.

TYPE

String

DEFAULT

disabled

DESCRIPTION

This variable specifies which composite aggressor mode is used in PrimeTime SI noise analysis. Allowed values are *disabled* (the default), which turns off the composite aggressor feature. *normal*, causes PrimeTime SI to calculate noise by utilizing the non-statistical composite aggressor feature. Selecting *statistical* causes PrimeTime SI to calculate noise by using the statistical composite aggressor flow.

In *disabled* composite aggressor mode, PrimeTime SI uses its original flow with composite aggressor completely off to analyze the noise.

In *normal* composite aggressor mode, PrimeTime SI aggregates the effect of multiple small aggressors into a single composite aggressor, thereby reducing the computational complexity and improving the performance.

statistical composite aggressor mode reduces the pessimism for noise analysis by reducing the effect of composite aggressor.

For the current value of this variable, type **printvar si_noise_composite_aggr_mode**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012983

noise_composite_aggr_mode

Persistent parameter that controls whether a composite aggressor is used to model the combined noise of all small aggressors, including the filtered ones, when performing noise analysis using the PTSI model.

Syntax:

```
noise_composite_aggr_mode disabled | normal | statistical
```

where the values have the following meaning:

disabled	Do not use a composite aggressor to model the combined effect of all weak aggressors.
normal	Use a composite aggressor to model the combined effect of all weak aggressors and assume the worst possible aggressor alignment.
statistical	Use a composite aggressor to model the combined effect of all weak aggressors, and use a statistical model. Compared to the *normal* mode, this reduces the pessimism.

The default value is *disabled*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793844

si_use_driving_cell_derate_for_delta_delay

Allows crosstalk delta delay for one net to be derated using the relevant derate factor for the cell driving that net.

TYPE

boolean

DEFAULT

FALSE

GROUP

si_variables

DESCRIPTION

When this variable is set to *true* the crosstalk delta delays for each net will be derated using the derate factors from the cell driving that net.

The relevant derate factor to be applied will adhere to the same precedence rules as the driving cell itself. For example, if no instance-specific derate factor was set on the driving cell then the hierarchical cell, the library cell and finally the global derate factors will be checked for a relevant derate factor.

To see what derate factors are to be applied to the net in question, first obtain the driving cell ($driving_cell) and use: pt_shell **report_timing_derate [get_cells $driving_cell]**

If the command **report_timing** is invoked with the **-derate** option then the un-derated crosstalk delta delay will be reported as before. In addition the derate column will report the net derate factor used to derate the delta-free net delay.

To determine the current value of this variable, enter the following command: pt_shell **printvar si_use_driving_cell_derate_for_delta_delay** or pt_shell **echo $si_use_driving_cell_derate_for_delta_delay**

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012956

si_use_driving_cell_derate_for_delta_delay

Persistent parameter that controls whether crosstalk delta delay on a victim net will be derated using the same derate factor for the cell driving that net.

Syntax:

```
si_use_driving_cell_derate_for_delta_delay true | false
```

where the values have the following meaning:

true	Use the derating factor for the delay on a victim net also as a derating factor when calculating the delta delay on a victim net caused by any aggressor net.
false	Do not use the derating factor on a victim net when calculating the delta delay that an aggressor causes on a victim net.

The default value is *false*.

PI's Ex. 336 (PrimeTime 2006.12) at ATopTec001857347

si_xtalk_composite_aggr_mode

Specifies the composite aggressor mode for crosstalk delay.

TYPE

String

DEFAULT

disabled

DESCRIPTION

This variable specifies which composite aggressor mode is used in PrimeTime SI crosstalk delay analysis. Allowed values are *disabled* (the default), which turns off the composite aggressor feature. *normal*, causes PrimeTime SI to calculate crosstalk delay by utilizing the non-statistical composite aggressor feature. Selecting *statistical* causes PrimeTime SI to calculate crosstalk by using the statistical composite aggressor flow.

In *disabled* composite aggressor mode, PrimeTime SI uses its original flow with composite aggressor completely off to calculate the xtalk delay.

In *normal* composite aggressor mode, PrimeTime SI aggregates the effect of some small aggressors (including filtered ones) into a single composite aggressor, thereby reducing the computational complexity and improving the performance.

statistical composite aggressor mode reduces the pessimism for xtalk delay analysis by reducing the effect of composite aggressor.

For the current value of this variable, type **printvar si_xtalk_composite_aggr_mode**.

PI's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012985

xtalk_composite_aggr_mode

Persistent parameter that controls whether a composite aggressor is used to model the combined impact of all small aggressors, including the filtered ones, when performing crosstalk delay analysis using the PTSI model.

Syntax:

```
xtalk_composite_aggr_mode disabled | normal | statistical
```

where the values have the following meaning:

disabled	Do not use a composite aggressor to model the combined effect of all weak aggressors.
normal	Use a composite aggressor to model the combined effect of all weak aggressors and assume the worst possible aggressor alignment.
statistical	Use a composite aggressor to model the combined effect of all weak aggressors, and use a statistical model. Compared to the *normal* mode, this reduces the pessimism.

The default value is *disabled*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793848

si_xtalk_composite_aggr_noise_peak_ratio

Used to control the composite aggressor selection for xtalk analysis.

TYPE

float

DEFAULT

0.01

DESCRIPTION

Specifies the threshold value in crosstalk bump to Vdd ratio, below which aggressors are selected into composite aggressor group. The default value is 0.01, which means all the aggressor nets with crosstalk bump to Vdd ratio less than 0.01 will be selected into composite aggressor group. It works together with other filtering thresholds **si_filter_per_aggr_noise_peak_ratio** and **si_filter_accum_aggr_noise_peak_ratio** to determine which aggressors can be selected into composite aggressor group.

To determine the current value of this variable, type **printvar si_xtalk_composite_aggr_noise_peak_ratio** at the PT shell prompt.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012984

xtalk_composite_aggr_noise_peak_ratio

Persistent parameter that specifies the noise peak threshold of an individual aggressor. If the noise peak is above the threshold, the noise impact of the aggressor is calculated individually. If the noise peak is below the threshold, the aggressor is considered weak and its noise impact is modeled by the composite aggressor.

Syntax:

```
xtalk_composite_aggr_noise_peak_ratio ratio
```

where *ratio* is the threshold noise voltage as a fraction of the supply below which the impact of the aggressor is modeled as part of the composite aggressor.

The default value is *1.000e-02*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793849

si_xtalk_composite_aggr_quantile_high_pct

Used to control the composite aggressor creation for statistical analysis.

TYPE

float

DEFAULT

99.73

DESCRIPTION

This variable is set to the desired probability in percentage format that any given real combined bump height will be less than or equal to the computed composite aggressor bump height. Given the desired probability, the resulting quantile value for the composite aggressor bump height will be calculated.

The default value of this variable is 99.73, which corresponds to a 3-sigma probability that the real bump height from any randomly-chosen combination of aggressors will be covered by the composite aggressor bump height.

To determine the current value of this variable, type **printvar si_xtalk_composite_aggr_quantile_high_pct** at the PT shell prompt.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012986

xtalk_composite_aggr_quantile_high_pct

Persistent parameter that specifies the required confidence level to use for the statistical model of the composite aggressor. The value is the probability in percentage that a real noise bump will be smaller than the calculated bump.

Syntax:

`xtalk_composite_aggr_quantile_high_pct` *`percentage`*

where *`percentage`* is the likelihood that a real bump is smaller than or equal to the calculated composite bump.

The default value is *99.73%*, which corresponds to a 3-sigma probability.

si_xtalk_delay_analysis_mode

Specifies the arrival window alignment mode for crosstalk delay.

TYPE

String

DEFAULT

all_paths

DESCRIPTION

This variable specifies how the alignment between victim & aggressors is performed in crosstalk delay analysis PrimeTime SI. Allowed values are *all_paths* (the default), which causes PrimeTime SI to calculate crosstalk for all paths through the victim net. *worst_path*, causes PrimeTime SI to calculate crosstalk for all the worst paths(the earliest/latest path) through the victim net. Selecting *all_violating_paths* causes PrimeTime SI to calculate crosstalk for all worst paths and paths with the negative slack.

In *all_paths* alignment mode, PrimeTime SI considers the largest possible crosstalk delta delay for the given victim & aggressor arrival windows. This gurantees that all paths going through the victim net with different arrival times are conservative. This is default and tradational way PTSI calculated delta delay. The limitation of this approach is worst crosstalk delta delay applied to all paths including the worst path which causes slack of the design to be pessimistic. When the path based analysis is done on a path using *get_recalculated_timing_paths* the above pessimism is removed for the specific path. However it is too expensive do path based analysis analysis on all the paths of the design.

In *worst_path* alignment mode, PrimeTime SI aligns aggressors for the the earlist/ latest paths on the victim, so only crosstalk affecting to these worst path is considered. So only the crosstalk affect that makes the slowest (earliest) path any slower(faster) is calculated. If the slowest/earliest path is a *set_false_path*, the true path is considerd. Considering the worst path instead of all paths, typically generates smaller delta delays and the worst paths and the design slack becomes less pessimistic. This approach makes sure that design slack & worst path are conservative.

There is a caveat to *worst_path* alignment mode, the crosstalk delay is applicable to worst path only, so the sub-critical path delay may be inaccurate. The side affect of this limitation *report_timing -nworst N*, N1 could report paths with optimistic slacks. Also *report_constraint -all_violators -max_delay -min_delay* will report less number of violations than really exist on the design. Also as violating critical paths is fixed, the optimistic sub-critical paths will be critical and start violating. Also bottleneck commands like *report_timing_bottleneck* and *report_si_bottleneck* will be less effective.

For some design flows the sub-critical path optimism is less of an issue if the design meets the timing constraints, i.e. all endpoints in the design show positive slacks. However, when the design has not met the timing yet, getting conservative crosstalk deltas for all the violating paths (whose slack is negative) is essential for the fixing flow. The alignment mode *all_violating_paths* addresses this by

xtalk_delay_analysis_mode

Persistent parameter that specifies the alignment that is assumed between victim and aggressors during crosstalk delay analysis.

Syntax:

```
xtalk_delay_analysis_mode all_paths | all_violating_paths \
                          | worst_path
```

where the values have the following meaning:

all_paths	Calculate crosstalk for all paths through the victim net. This is the most conservative approach.
all_violating_paths	Calculate crosstalk only for paths with a negative slack.
worst_path	Calculate crosstalk only for the worst paths through the net, that is, the earliest and latest path.

The default value is *all_paths*.

aligning the aggressors for all the violating and the worst path through any pin in the design. This means, all paths with negative slacks and also all the critical paths through any pin in the design (even if the slack for that worst path is positive) have a conservative slack. This mode may show more pessimism on worst paths than the worst_path mode and also the runtime might get slightly higher than *worst_path*. To reduce pessimism furthur, in the new modes *worst_path* and *all_violating_path*, another feature is enabled, where in the clock n/w seperate delay calculation is	

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793858

si_xtalk_exit_on_max_iteration_count

Specifies a maximum number of incremental timing iterations, after which PrimeTime-SI exits the analysis loop.

TYPE

integer

DEFAULT

2

DESCRIPTION

Specifies a maximum number of incremental timing iterations. PrimeTime-SI exits the analysis loop after performing this number of iterations.

The default value of this variable is 2, meaning that PrimeTime-SI exits the analysis loop after performing two iterations. You can override this default by setting the variable to another integer; the minimum allowed value is 1.

This variable is one of a set of six variables that determine exit criteria; PrimeTime-SI exits the analysis loop after completing the current iteration if one or more of the following is true:

1. The number of iterations performed equals the value of the **xtalk_max_iteration_count** variable.

2. All delta delays fall between the values of the **si_xtalk_exit_on_min_delta_delay** and **si_xtalk_exit_on_max_delta_delay** variables.

3. The number of nets selected for reevaluation in the next iteration is less than the value of the **si_xtalk_exit_on_number_of_reevaluated_nets** variable.

4. The percentage of nets (relative to the total number of nets) selected for reevaluation is less than the value of the **si_xtalk_exit_on_reevaluated_nets_pct** variable.

5. The percentage of nets (relative to the number of cross-coupled nets) selected for reevaluation is less than the value of the **si_xtalk_exit_on_coupled_reevaluated_nets_pct** variable.

6. You manually exit the analysis loop by pressing Control-C to send an interrupt signal to the PrimeTime process. The interrupt is handled as any other exit criteria, at the end of the current iteration of the crosstalk analysis. You cannot interrupt iteration immediately without exiting PrimeTime.

To determine the current value of this variable, type **printvar si_xtalk_exit_on_max_iteration_count**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012987

xtalk_exit_on_max_iteration_count

Persistent parameter that specifies the maximum number of iterations allowed between total delay calculation and crosstalk-induced delay calculation. If there is no convergence after the specified maximum number of iterations, Aprisa fails and issues an error.

Syntax:

```
xtalk_exit_on_max_iteration_count maxcount
```

where *maxcount* is the maximum number of iterations allowed.

The default value is 2.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793870

si_xtalk_reselect_delta_delay

Specifies the threshold of net delay change caused by crosstalk analysis, above which PrimeTime-SI reselects the net for subsequent delay calculations.

TYPE

float

DEFAULT

5

DESCRIPTION

This variable specifies a reselection threshold in terms of absolute delta delay. Nets that have at least one net arc with a crosstalk-annotated delta delay above this threshold are selected for the next iteration of PrimeTime-SI delay calculations. Note that delta delays are annotated on net arcs, but they capture the change of stage delay (cell plus net) because of crosstalk.

This variable is one of a set of four variables that determine net reselection criteria. The other three variables are as follows:

si_xtalk_reselect_delta_delay_ratio
si_xtalk_reselect_max_mode_slack
si_xtalk_reselect_min_mode_slack

All four variables are ignored if the variable **si_xtalk_reselect_critical_path** is true.

To determine the current value of this variable, type **printvar si_xtalk_reselect_delta_delay**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012988

xtalk_reselect_delta_delay

Persistent parameter that specifies the threshold delta delay in ns above which the net is reselected for window filtering, provided reselection is enabled with the xtalk_reselect_delta_and_slack parameter.

Syntax:

```
xtalk_reselect_delta_delay deltaT
```

where *deltaT* is the threshold delta delay above which the net is reselected.

The default value is *5.000*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793871

si_xtalk_reselect_delta_delay_ratio

Specifies the threshold of the ratio of net delay change caused by crosstalk analysis to the total stage delay, above which PrimeTime-SI reselects a net for subsequent delay calculations.

TYPE

float

DEFAULT

0.95

DESCRIPTION

This variable specifies a reselection threshold in terms of the delta delay ratio. Nets that have at least one net arc with a crosstalk-annotated delta delay, where the ratio of the annotated delta to the stage delay is above this threshold, are selected for the next iteration of PrimeTime-SI delay calculations.

If a net has multiple stage delays (because of a net fanout greater than one or multiple cell arcs), PrimeTime-SI considers the stage delta delay and stage delay that result in higher delta to stage delay ratio, thus making reselection conservative.

This variable is one of a set of four variables that determine net reselection criteria. The other three variables are as follows:

si_xtalk_reselect_delta_delay
si_xtalk_reselect_max_mode_slack
si_xtalk_reselect_min_mode_slack

All four variables are ignored if the variable **si_xtalk_reselect_critical_path** is true.

To determine the current value of this variable, type **printvar si_xtalk_reselect_delta_delay_ratio**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012989

xtalk_reselect_delta_delay_ratio

Persistent parameter that specifies the threshold delta delay as a fraction of the stage delay above which the net is reselected for window filtering, provided reselection is enabled with the xtalk_reselect_delta_and_slack parameter.

Syntax:

```
xtalk_reselect_delta_delay_ratio deltaT
```

where *deltaT* is the fraction of stage delay above which the net is reselected.

The default value is *0.950*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793872

si_xtalk_reselect_max_mode_slack

Specifies the max mode pin slack threshold, below which PrimeTime-SI reselects a net for subsequent delay calculations.

TYPE

float

DEFAULT

0

DESCRIPTION

This variable specifies the pin slack threshold in the max mode. Nets that have at least one pin with a max mode slack below this threshold are selected for the next iteration of PrimeTime-SI delay calculations. Max-mode pin slack is the slack of the worst max-mode (setup) path through the pin.

This variable is one of a set of four variables that determine net reselection criteria. The other three variables are as follows:

si_xtalk_reselect_delta_delay
si_xtalk_reselect_delta_delay_ratio
si_xtalk_reselect_min_mode_slack

All four variables are ignored if the variable **si_xtalk_reselect_critical_path** is true.

To determine the current value of this variable, type **printvar si_xtalk_reselect_max_mode_slack**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012990

xtalk_reselect_max_mode_slack

Persistent parameter that specifies the threshold delta delay as a fraction of the stage delay above which the net is reselected for window filtering, provided reselection is enabled with the xtalk_reselect_delta_and_slack parameter.

Syntax:

```
xtalk_reselect_max_mode_slack setupSlack
```

where *setupSlack* is the threshold slack below which a net is reselected.

The default value is *0.000*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793873

si_xtalk_reselect_min_mode_slack

Specifies the min mode pin slack threshold, below which PrimeTime-SI reselects a net for subsequent delay calculations.

TYPE

float

DEFAULT

0

DESCRIPTION

This variable specifies the pin slack threshold in the min mode. Nets that have at least one pin with a min mode slack below this threshold are selected for the next iteration of PrimeTime-SI delay calculations. Min-mode pin slack is the slack of the worst min-mode (hold) path through the pin.

This variable is one of a set of four variables that determine net reselection criteria. The other three variables are as follows:

si_xtalk_reselect_delta_delay
si_xtalk_reselect_delta_delay_ratio
si_xtalk_reselect_max_mode_slack

All four variables are ignored if the variable **si_xtalk_reselect_critical_path** is true.

To determine the current value of this variable, type **printvar si_xtalk_reselect_min_mode_slack**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012990

xtalk_reselect_min_mode_slack

Persistent parameter that specifies the threshold delta delay as a fraction of the stage delay above which the net is reselected for window filtering, provided reselection is enabled with the xtalk_reselect_delta_and_slack parameter.

Syntax:

`xtalk_reselect_min_mode_slack` *`holdSlack`*

where *`holdSlack`* is the threshold slack below which a net is reselected.

The default value is *0.000*.

PI's Ex. 36 (PrimeTime D-2010.06) at SYNPS-00010152

timing_aocvm_enable_analysis

Enable PrimeTime's graph-based AOCVM analysis.

TYPE

Boolean

DEFAULT

false

GROUP

timing_variables

DESCRIPTION

When *false* (default) the graph-based AOCVM timing update is not performed. A path-based aocvm analysis can be performed in this mode using the **-pba_mode** option on the **report_timing** and **get_timing_paths** commands. In this mode constant timing derates specified using the **set_timing_derate command are required to pessimistically bound the analysis. You should specify constant derates that do not clip the range of the path-based AOCVM derates to avoid optimism.**

When *true* the graph-based aocvm timing update is performed as part of the **update_timing** command. A path-based aocvm analysis can also be performed in this mode. In this mode constant timing derates are not required and, in fact, constant derates for *static delays* are ignored. Graph-based AOCVM derates computed during **update_timing** tightly bound the path-based AOCVM derates without clipping their range. Note that setting this variable to *true* will automatically switch the design into *on_chip_variation* analysis mode using the **set_operating_conditions** command.

PI's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012959

timing_aocvm_enable_analysis

Persistent parameter that controls whether to enable advanced on-chip variation analysis. This advanced analysis uses knowledge about geographical distance and logic depth to better calculate the impact of systematic and random variations within one chip.

Use the *read_aocvm* Tcl command to specify the two-dimensional distance- and logic-depth-dependent derating models.

Use the *set_timing_derate -aocvm_guardband* Tcl command to specify an additional guardband derating factor to use for either early or late paths.

Use the *timing_aocvm_analysis_mode* Tcl variable to control how to combine these early and late guardbands with the advanced OCV derating factor.

- If this variable is not set, then these derating factors are considered independent, and the total effect of these derating factors is the square root of the sum of the squares of the effects of derating.
- If the variable is set with the value *correlated_components*, then the derating factors are considered fully correlated and the total effect of deration is the sum of the effects of each derating factor.

Use the *report_timing* Tcl command with the *-aocvm* argument to see the resulting derating factors that were applied to net and cell delay models in the reported timing paths.

Syntax:

```
timing_aocvm_enable_analysis true | false
```

where the values have the following meaning:

true	Enable advanced on-chip variation analysis.
false	Disable advanced on-chip variation analysis.

The default value is *false*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793884

timing_clock_reconvergence_pessimism

Select signal transition sense matching for computing clock reconvergence pessimism removal.

TYPE

string

DEFAULT

normal

DESCRIPTION

Determines how the value of the clock reconvergence pessimism removal (crpr) is computed with respect to transition sense. Allowed values are *normal* (the default) and *same_transition*.

When set to *normal*, the crpr value is computed even if the clock transitions to the source and destination latches are in different directions on the common clock path. It is computed separately for rise and fall transitions and the value with smaller absolute value is used.

When set to *same_transition*, the crpr value is computed only when the clock transition to the source and destination latches have a common path and the transition is in the same direction on each pin of the common path. Thus if the source and destination latches are triggered by different edge types, crpr has a value of zero.

To determine the current value of this variable, type **printvar timing_clock_reconvergence_pessimism** or **echo $timing_clock_reconvergence_pessimism**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012961

timing_clock_reconvergence_pessimism

Persistent parameter that controls the clock transitions that are used to reduce reconvergence pessimism.

Syntax:

```
timing_clock_reconvergence_pessimism \
                                    normal | same_transition
```

where the values have the following meaning:

normal	Use both same and inverse transition conditions to reduce reconvergence pessimism.
same_transition	Only use same transition conditions to reduce reconvergence pessimism.

The default value is *normal*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793888

timing_crpr_remove_clock_to_data_crp

Allows the removal of Clock Reconvergence Pessimism (CRP) from paths that fan out directly from clock source to the data pins of sequential devices.

TYPE

boolean

DEFAULT

FALSE

DESCRIPTION

When this variable is set to *true* then CRP will be removed for all paths that fan out directly from clock source pins to the data pins of sequential devices.

It should be noted that when this variable is set to *true* all sequential devices that reside in the fanout of clock source pins must be handled seperately in the subsequent timing update. This may cause a severe performance degradation to the timing update.

```
pt_shell echo $timing_crpr_remove_clock_to_data_crp
or
      pt_shell printvar timing_crpr_remove_clock_to_data_crp
```

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012962

timing_crpr_remove_clock_to_data_crp

Persistent parameter that controls whether to remove Clock-to-data Reconvergence Pessimism (CRP) during timing analysis.

By default, arrival time uncertainty of the clock signal and data signal are handled independent of each other. This approach may be overly pessimistic for cases where the clock path and data path share some logic. To reduce that pessimism, and to obtain a more accurate analysis, you may want to enable this parameter.

Syntax:

```
timing_crpr_remove_clock_to_data_crp true | false
```

where the values have the following meaning:

true	Remove clock-to-data CRP. This setting causes Aprisa to consume a lot more memory, and to run for a much longer time.
false	Do not remove clock-to-data CRP. This is much more efficient, but may be overly pessimistic.

The default value is *false*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793889

timing_crpr_threshold_ps

Specifies amount of pessimism that clock reconvergence pessimism removal (CRPR) is allowed to leave in the report.

TYPE

float

DEFAULT

20

DESCRIPTION

Specifies amount of pessimism that clock reconvergence pessimism removal (CRPR) is allowed to leave in the report. The unit is in pico seconds (ps), regardless of the units of the main library.

The threshold is per reported slack: setting the this variable to the *TH1* value means that reported slack is no worse than *S - TH1*, where *S* is the reported slack when **timing_crpr_threshold_ps** is set close to zero (the minimum allowed value is **1** picosecond).

The variable has no effect if CRPR is not active (**timing_remove_clock_reconvergence_pessimism** is false). The larger the value of **timing_crpr_threshold_ps**, the faster the runtime when CRPR is active. The recommended setting is about one half of the stage (gate plus net) delay of a typical stage in the clock network. It provides a reasonable trade-off between accuracy and runtime in most cases. You may want to use different settings throughout the design cycle: larger during the design phase, smaller for sign-off. You might have to experiment and set a different value when moving to a different technology.

To determine the current value of this variable, type **printvar timing_crpr_threshold_ps**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012963

timing_crpr_threshold_ps

Persistent parameter that specifies the amount of pessimism that clock reconvergence pessimism removal (CRPR) is allowed to leave in the report. So, this parameter effectively controls how much effort CRPR should spend on a reconvergent path.

Syntax:

```
timing_crpr_threshold_ps threshold
```

where *threshold* is a real number greater than *1.0*, defining the maximum amount of pessimism that can be left in a reconvergent path after CRPR.

The default value is *20.000*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793890

timing_disable_bus_contention_check

Disable checking for timing violations resulting from transient contention on design busses.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

Applies only to bus designs that have multiple three-state drivers.

When *true*, PrimeTime ignores timing setup and hold (max and min) violations that occur as a result of transient bus contention. When *false* (the default), PrimeTime reports these timing violations.

Bus contention occurs when more than one driver is enabled at the same time. By default, PrimeTime treats the bus as if it is in an unknown state during this region of contention, and reports a timing violation if the setup and hold regions extend into the contention region. Note that checking is done only for timing violations, and not for logical and excessive power dissipation violations, which are outside the scope of static timing analysis tools.

Set this variable to *true* only if you are certain that transient bus contention regions will never occur. By setting the value to *true*, you guarantee that on a multi-driven three-state bus, the drivers in the previous clock cycle are disabled before the drivers in the current clock cycle are enabled. If you set this variable to *true*, you must ensure that the variable **timing_disable_bus_contention_check** is *false*. The variables **timing_disable_bus_contention_check** and **timing_disable_floating_bus_check** cannot both be *true* at the same time.

During the switching between the high-impedance (Z) state and the high/low state, the timing behavior (for example, intrinsic delay) of three-state buffers is captured in the Synopsys library using the timing arc types three_state_disable and three_state_enable. These timing arcs connect the enable pin to the output pin of the three-state buffers. For details, see the *Library Compiler Reference Manual*.

To determine the current value of this variable, type **printvar timing_disable_bus_contention_checks** or **echo $timing_disable_bus_contention_checks**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012963

timing_disable_bus_contention_check

Persistent parameter that controls whether the timing analyzer should perform bus contention checks.

Syntax:

```
timing_disable_bus_contention_check true | false
```

where the values have the following meaning:

true	Do not perform bus contention checks.
false	Perform bus contention checks.

The default value is *false*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793891	Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012964

timing_disable_clock_gating_checks

Disable checking for setup and hold clock gating violations.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

When *true*, disables clock-gating setup and hold checks. When *false* (the default), PrimeTime automatically determines clock-gating and performs clock-gating setup and hold checks.

To determine the current value of this variable, type **printvar timing_disable_clock_gating_checks** or **echo $timing_disable_clock_gating_checks**.

timing_disable_clock_gating_checks

Persistent parameter that controls whether to perform clock-gating setup and hold checks.

Syntax:

```
timing_disable_clock_gating_checks true | false
```

where the values have the following meaning:

true	Do not perform clock gating setup and hold checks.
false	Perform clock gating setup and hold checks.

The default value is *false*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793892

timing_disable_cond_default_arcs

Disable the default, non-conditional timing arc between pins that do have conditional arcs.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

When *true*, disables nonconditional timing arcs between any pair of pins that have at least one conditional arc. When *false* (the default), these nonconditional timing arcs are not disabled. This variable is primarily intended to deal with the situation between two pins that have conditional arcs, where there is always a default timing arc with no condition.

Set this variable to *true* when the specified conditions cover all possible state-dependent delays, so that the default arc is useless. For example, consider a 2-input XOR gate with inputs as A and B and with output as Z. If the delays between A and Z are specified with 2 arcs with respective conditions 'B' and 'B~", the default arc between A and Z is useless and should be disabled.

To determine the current value of this variable, type **printvar timing_disable_cond_default_arcs** or **echo $timing_disable_cond_default_arcs**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012965

timing_disable_cond_default_arcs

Persistent parameter that controls whether the default condition of any conditional timing arc should be ignored.

Syntax:

```
timing_disable_cond_default_arcs true | false
```

where the values have the following meaning:

true	Do not examine the default condition of conditional timing arcs.
false	Do consider the default condition as any other condition of conditional timing arcs.

The default value is *false*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793893

timing_disable_floating_bus_check

Disable checking for timing violations resulting from transient floating design buses.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

Applies only to bus designs that have multiple three-state drivers.

When *true*, PrimeTime ignores timing setup and hold (max and min) violations that occur as a result of transient floating buses. When *false* (the default), PrimeTime reports these timing violations.

Floating bus condition occurs when no driver controls the bus at a given time. By default, PrimeTime treats the bus as if it is in an unknown state during this region of contention, and reports a timing violation if the setup and hold regions extend into the floating region. Note that checking is done only for timing violations, and not for logical violations, which are outside the scope of static timing analysis tools.

Set this value to *true* only if you are certain that transient floating bus regions will never occur. By setting the value to *true*, you guarantee that on a multi-driven three-state bus, the drivers in the previous clock cycle are disabled before the new drivers in the current clock cycle are enabled. If you set this variable to *true*, you must ensure that the variable **timing_disable_bus_contention_check** is *false*. The variables **timing_disable_floating_bus_check** and **timing_disable_bus_contention_check** cannot both be *true* at the same time.

During the switching between the high-impedance (Z) state and the high/low state, the timing behavior (for example, intrinsic delay) of three-state buffers is captured in the Synopsys library using the timing arc types three_state_disable and three_state_enable. These timing arcs connect the enable pin to the output pin of the three-state buffers. For details, see the *Library Compiler Reference Manual*.

To determine the current value of this variable, type **printvar timing_disable_floating_bus_check** or **echo $timing_disable_floating_bus_check**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012965

timing_disable_floating_bus_check

Persistent parameter that controls whether to check for floating busses.

Syntax:

```
timing_disable_floating_bus_check true | false
```

where the values have the following meaning:

true	Do not check for floating busses.
false	Check for floating busses.

The default value is *false*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1973894

timing_disable_internal_inout_cell_paths

Enable bidirectional feedback paths within a cell.

TYPE

Boolean

DEFAULT

true

DESCRIPTION

When *true* (the default), PrimeTime automatically disables bidirectional feedback paths in a cell. When *false*, bidirectional feedback paths in cells are enabled.

This variable has no effect on timing of bidirectional feedback paths that involve more than one cell (that is, if nets are involved); these feedback paths are controlled by the variable **timing_disable_internal_inout_net_arcs**.

To determine the current value of this variable, type **printvar timing_disable_internal_inout_cell_paths** or **echo $timing_disable_internal_inout_cell_paths**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012966

timing_disable_internal_inout_cell_paths

Persistent parameter that controls whether to analyze feedback paths inside a cell through a bidirectional pin of that cell.

Syntax:

```
timing_disable_internal_inout_cell_paths true | false
```

where the values have the following meaning:

true	Do not analyze timing paths inside a cell through a bidirectional pin of that cell.
false	Also analyze timing paths inside a cell through a bidirectional pin of that cell.

The default value is *true*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793895

timing_disable_internal_inout_net_arcs

Controls whether bidirectional feedback paths across nets are disabled or not.

TYPE

Boolean

DEFAULT

true

DESCRIPTION

When *true* (the default), PrimeTime automatically disables bidirectional feedback paths that involve more than one cell; no path segmentation is required. When *false*, these bidirectional feedback paths are enabled.

This variable has no effect on timing of bidirectional feedback paths that are completely contained in one cell (that is, if nets are not involved); these feedback paths are controlled by the variable **timing_disable_internal_inout_cell_paths**.

To determine the current value of this variable, type **printvar timing_disable_internal_inout_net_arcs** or **echo $timing_disable_internal_inout_net_arcs**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012967

timing_disable_internal_inout_net_arcs

Persistent parameter that controls whether to analyze feedback paths through a bidirectional pin.

Syntax:

```
timing_disable_internal_inout_net_arcs true | false
```

where the values have the following meaning:

true	Ignore feedback paths through bidirectional pins.
false	Analyze feedback paths through bidirectional pins.

The default value is *false*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793896

timing_disable_recovery_removal_checks

Disable or enable the timing analysis of recovery and removal checks in the design.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

When *true*, disables recovery and removal timing analysis. When *false* (the default), PrimeTime performs recovery and removal checks; for descriptions of these checks, see the man page for the **report_constraint** command.

To determine the current value of this variable, type **printvar timing_disable_recovery_removal_checks** or **echo $timing_disable_recovery_removal_checks**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012969

timing_disable_recovery_removal_checks

Persistent parameter that controls whether to perform recovery and removal timing checks. Recovery and removal checks are similar to setup and hold checks, but are intended for asynchronous signals, such as set and reset signals.

Syntax:

```
timing_disable_recovery_removal_checks true | false
```

where the values have the following meaning:

true	Do not perform recovery and removal checks.
false	Perform recovery and removal checks.

The default value is *false*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793898

timing_early_launch_at_borrowing_latches

Removes clock latency pessimism from the launch times for paths which begin at the data pins of transparent latches.

TYPE

Boolean

DEFAULT

true

DESCRIPTION

In the following description we assume that the data paths of interest are setup paths since we refer specifically to time borrowing scenarios. However, if **timing_allow_short_path_borrowing** is enabled then the same discussion applies to borrowing hold paths too.

When a latch is in its transparent phase, data arriving at the D-pin passes through the element as though it were combinational. To model this scenario, whenever PrimeTime determines that time borrowing occurs at such a D-pin, paths which originate at the D-pin are created.

Sometimes there is a difference between the launching and capturing latch latencies, due either to reconvergent paths in the clock network or different min and max delays of cells in the clock network. For setup paths, PrimeTime uses the late value to launch and the early value to capture. This achieves the tightest constraint and avoids optimism. However, for paths starting from latch D-pins this is pessimistic since data simply passes through and thus does not even "see" the clock edge at the latch.

When this timing variable is set to *true* (the default), such pessimism is eliminated by using the early latch latency to launch such paths. Note that only paths which originate from a latch D-pin are affected. When the variable is set to *false*, late clock latency is used to launch all setup paths in the design.

It is recommended that the user avail of this form of pessimism removal since it does not cause the run-time of the analysis to increase. However, it is also advised that the user disable it when clock reconvergence pessimism removal (CRPR) is enabled (i.e. when **timing_remove_clock_reconvergence_pessimism** is *true*). CRPR may not be applied to paths which have been launched using an early latency or the results may be optimistic. Since CRPR is a more sophisticated and accurate means of pessimism removal, the user should disable **timing_early_launch_at_borrowing_latches** when CRPR is enabled so that CRPR applies to all paths in the design. In this mode, note that the D-pin launch time is not modified by the open edge CRP - since late launch latency is used at the path startpoint, to additionally add CRP would be pessimistic, representing a "double-counting" of early-late differences.

To determine the current value of this variable, type **printvar timing_early_launch_at_borrowing_latches** or **echo $timing_early_launch_at_borrowing_latches**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012968

timing_early_launch_at_borrowing_latches

Persistent parameter that controls the clock latency to use for latches in paths requiring time borrowing.

Syntax:

```
timing_early_launch_at_borrowing_latches true | false
```

where the values have the following meaning:

true	Use the early clock latency for the latch cell.
false	Use the late clock latency for the latch cell.

The default value is *true*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTec0018543

timing_enable_preset_clear_arcs

Controls whether PrimeTime enables or disables preset and clear arcs.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

When *true*, permanently enables asynchronous preset and clear timing arcs, so that you use them to analyze timing paths. When *false* (the default), PrimeTime disables all preset and clear timing arcs.

Note that if there are any minimum pulse width checks defined on asynchronous preset and clear pins they are performed regardless of the value of this variable. Also note the the -true and the -justify options of report_timing cannot be used unless this variable is at its default value.

To determine the current value of this variable, type **printvar timing_enable_preset_clear_arcs**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012969

timing_enable_preset_clear_arcs

Persistent parameter that controls whether to analyze preset and clear timing constraints.

Syntax:

```
timing_enable_preset_clear_arcs true | false
```

where the values have the following meaning:

true	Take into account preset and clear timing constraints.
false	Ignore preset and clear timing constraints.

The default value is *false*.

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793911

timing_input_port_default_clock

Determines whether a default clock is assumed at input ports for which the user has not defined a clock with set_input_delay.

TYPE

Boolean

DEFAULT

true

DESCRIPTION

This Boolean variable affects the behavior of PrimeTime when the user sets an input delay without a clock on an input port. When *true* (the default value), the input delay on the port is set with respect to one imaginary clock so that the inputs are constrained. This also causes the clocks along the paths driven by these input ports to become related. When *false*, no such imaginary clock is assumed.

To determine the current value of this variable, type **printvar timing_input_port_default_clock**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012969

timing_input_port_default_clock

Persistent parameter that controls whether to assign a default clock to an input port if no input delay is specified on that port.

Syntax:

```
timing_input_port_default_clock true | false
```

where the values have the following meaning:

true	Assign a default clock if no input delay is specified.
false	Do not assign a default clock if no input delay is specified.

The default value is *false*.

timing_remove_clock_reconvergence_pessimism

Enables or disables clock reconvergence pessimism removal

TYPE

Boolean

DEFAULT

false

DESCRIPTION

When this variable is set to *true*, PrimeTime removes clock reconvergence pessimism from slack calculation and minimum pulse width checks. This variable replaces the following discontinued options:

-report_clock_reconvergence_pessimism
-remove_clock_reconvergence_pessimism

of the **report_timing**, **report_constraint**, and **get_timing_paths** commands.

Clock reconvergence pessimism (CRP) is a difference in delay along the common part of the launching and capturing clock paths. The most common causes of CRP are reconvergent paths in the clock network, and different min and max delay of cells in the clock network.

CRP is independently calculated for rise and fall clock paths. You can use the variable **timing_clock_reconvergence_pessimism** to control CRP calculation with respect to transition sense. In the case of the capturing device being a level-sensitive latch two CRP values will be caculated:
• crp_open, which is the CRP corresponding to the opening edge of the latch
• crp_close, which is the CRP corresponding to the closing edge of the latch The required time at the latch will be increased by the value of crp_open and hence reduce the amount of borrowing (if any) at the latch. Meanwhile, the maximum time borrow allowed at the latch is affected by shifting the closing edge by crp_close. For more details, see the *PrimeTime User Guide: Fundamentals*.

For a more detailed description of a CRP calulation, use the **report_crpr** command.

CRP is calculated differently for minimum pulse-width checks. It is given as the minimum of (maximum rise arrival time - minimum rise arrival time) and (maximum fall arrival time - minimum fall arrival time) at the pin where the check is being made.

If the variable **si_enable analysis** is set to *true* delays in the clock network may also include delta delays resulting from crosstalk interaction. Such delays are dynamic in nature, that is, they may vary from one clock cycle to the next, causing different delay variations (either speed-up or slow-down) on the same network, but during different clock cycles.

Starting with U-2003.03 release PrimeTime only considers SI delta delays as part of the CRP calculation if the type of timing check deployed derives its data from the same clock cycle.

In transparent-latch based designs, it is recommended that the variable **timing_early_launch_at_borrowing_latches** should be set to *false* when CRP removal (CRPR) is enabled. In this case, CRPR will apply even to paths whose startpoints are borrowing, leading to better pessimism reduction overall.

Any effective change in the value of the **timing_remove_clock_reconvergence_pessimism** variable causes full update_timing. You cannot perform one report_timing operation that considers CRP and one that does not without full update_timing in between.

timing_remove_clock_reconvergence_pessimism

Persistent parameter that controls whether the timing analyzer will remove clock reconvergence pessimism. This pessimism arises in the default timing analysis when, in presence of on-chip-variation (OCV) and reconvergent clocks, overly conservative clock skew is calculated because one clock path uses minimum delay and another path uses maximum delay for the same cell. This pessimism can be removed but at a significant run time penalty.

Syntax:

```
timing_remove_clock_reconvergence_pessimism true | false
```

where the values have the following meaning:

true	Remove clock reconvergence pessimism.
false	Do not remove clock reconvergence pessimism.

The default value is *false*.

For backward compatibility, the discontinued options will appear for the first few releases after they are obsoleted. However, if the design is not up to date at the time they are executed, they will only set timing_remove_clock_reconvergence_pessimism to *true*

If the design is up to date, then the command with the discontinued option fails. Since the discontinued command options only set timing_remove_clock_reconvergence_pessimism to *true*, the **-report_clock_reconvergence_pessimism** option behavior is not backward compatible. It causes slack to be removed prior to selecting the worst path. In other words, it behaves the same as the discontinued **-remove_clock_reconvergence_pessimism** option of the **report_timing**, **reyport_constraint**, and **get_timing_paths** commands. As soon as possible, update your scripts to set the **timing_remove_clock_reconvergence_pessimism** variable to *true* instead of using the discontinued options.

Limitations: CRPR does not support paths that fan out directly from clock source pins to the data pins of sequential devices. To enable support for such paths the variable **timing_crpr_remove_clock_to_data_crp** must be set to TRUE. CRPR does not support ideal clock latency set on pins or ports. CRPR does not support propagated clocks set on pins or ports as opposed to clock objects.

To turn CRP removal on:

```
   pt_shell set timing_remove_clock_reconvergence_pessimism TRUE
TRUE
pt_shell report_timing
```

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793838

si_noise_nmos_threshold_ratio

Specifies the technology threshold voltage for NMOS devices divided by the Vcc.

TYPE

float

DEFAULT

0.2

DESCRIPTION

Specifies the technology threshold voltage for NMOS devices divided by Vcc; the default is 0.2. This variable, along with **si_noise_pmos_threshold_ratio**, makes up a pair of variables used by PrimeTime-SI during the noise analysis phase, to determine the steady state resistance value of the drivers in absence of a noise library.

When noise library is not present for a cell or for a design, this activates the PrimeTime-SI steady state resistance estimation mode. In this mode, steady state resistance gets estimated based on the value of the PMOS and NMOS threshold voltage values as well as other information extracted from the delay and slew tables.

Pl's Ex. 646 (AP 09.10.rel.1 Executable)

```
set_param si1 noise_nmos_threshold_ratio 0.200
#     Type    : float
#     Usage   : NMOS threshold for SI noise analysis
```

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793839

si_noise_pmos_threshold_ratio

Specifies the technology threshold voltage for PMOS devices divided by the Vcc.

TYPE

float

DEFAULT

0.2

DESCRIPTION

Specifies the technology threshold voltage for PMOS devices divided by Vcc; the default is 0.2. This variable, along with **si_noise_nmos_threshold_ratio**, makes up a pair of variables used by PrimeTime-SI during the noise analysis phase, to determine the steady state resistance value of the drivers in absence of a noise library.

When noise library is not present for a cell or for a design, this activates the PrimeTime-SI steady state resistance estimation mode. In this mode, steady state resistance gets estimated based on the value of the PMOS and NMOS threshold voltage values as well as other information extracted from the delay and slew tables.

Pl's Ex. 646 (AP 09.10.rel.1 Executable)

```
set_param si1 noise_pmos_threshold_ratio 0.200
#     Type    : float
#     Usage   : PMOS threshold for SI noise analysis
```

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793845

si_xtalk_analysis_effort_level

Specifies the level of effort for the PrimeTime-SI timing calculation mode.

TYPE

string

DEFAULT

medium

DESCRIPTION

Specifies the effort level for the PrimeTime-SI timing calculation mode. Allowed values are low, medium (the default) and high.

In the high effort mode, the most accurate crosstalk delay calculation is performed, which results in the highest run time.

In the medium effort mode, efficient heuristics are employed in certain situations to enable faster calculation of crosstalk delay. This can result in slightly pessimistic results.

In the low effort mode, the fastest crosstalk delay calculation is performed, which results in the smallest run time.

To determine the current value of this variable, type **printvar si_xtalk_analysis_effort_level**.

Pl's Ex. 646 (AP 09.10.rel.1 Executable)

```
set_param si1 xtalk_analysis_effort_level high
#     Values: low high
#     Usage  : ptsi xtalk analysis effort level
```

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793854-55

si_xtalk_exit_on_coupled_reevaluated_nets_pct

Specifies a maximum percentage of nets selected for reevaluation relative to the total number of coupled nets, below which PrimeTime-SI exits the analysis loop.

TYPE

float

DEFAULT

0

DESCRIPTION

Specifies a maximum percentage of nets selected for reevaluation relative to the total number of coupled nets. PrimeTime-SI exits the analysis loop after completing the current iteration, when the percentage of nets selected for reevaluation in the next iteration is less than this number. The number of coupled nets is based on detailed parasitics as read in by **read_parasitics**. That is, crosstalk filtering does not impact the count of coupled nets for the purpose of this variable. The number of coupled nets counts all individual net segments in the same way that [get_nets -hierarchical *] counts all nets in the design.

This variable is one of a set of six variables that determine exit criteria; PrimeTime-SI exits the analysis loop after completing the current iteration if one or more of the following is true:

1. The number of iterations performed equals the value of the **xtalk_max_iteration_count** variable.

2. All delta delays fall between the values of the **si_xtalk_exit_on_min_delta_delay** and **si_xtalk_exit_on_max_delta_delay** variables.

3. The number of nets selected for reevaluation in the next iteration is less than the value of the **si_xtalk_exit_on_number_of_reevaluated_nets** variable. the analysis loop.

4. The percentage of nets (relative to the total number of nets) selected for reevaluation is less than the value of the **si_xtalk_exit_on_reevaluated_nets_pct** variable.

5. The percentage of nets (relative to the number of cross-coupled nets) selected for reevaluation is less than the value of the **si_xtalk_exit_on_coupled_reevaluated_nets_pct** variable.

6. You manually exit the analysis loop by pressing Control-C to send an interrupt signal to the PrimeTime process. The interrupt is handled as any other exit criteria, at the end of the current iteration of the crosstalk analysis. You cannot interrupt iteration immediately without exiting PrimeTime.

To determine the current value of this variable, type **printvar si_xtalk_exit_on_coupled_reevaluated_nets_pct**.

Pl's Ex. 646 (AP 09.10.rel.1 Executable)

```
set_param si1 xtalk_exit_on_coupled_reevaluated_nets_pct 0.000
#     Type    : float
#     Usage   :
```

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793863

si_xtalk_exit_on_number_of_reevaluated_nets

Specifies a maximum number of nets selected for reevaluation, below which PrimeTime-SI exits the analysis loop.

TYPE

integer

DEFAULT

0

DESCRIPTION

Specifies a maximum number of nets selected for reevaluation. PrimeTime-SI exits the analysis loop after completing the current iteration, when the number of nets selected for reevaluation in the the next iteration is less than this number.

This variable is one of a set of six variables that determine exit criteria; PrimeTime-SI exits the analysis loop after completing the current iteration if one or more of the following is true:

1. The number of iterations performed equals the value of the **xtalk_max_iteration_count** variable.

2. All delta delays fall between the values of the **si_xtalk_exit_on_min_delta_delay** and **si_xtalk_exit_on_max_delta_delay** variables.

3. The number of nets selected for reevaluation in the next iteration is less than the value of the **si_xtalk_exit_on_number_of_reevaluated_nets** variable.

4. The percentage of nets (relative to the total number of nets) selected for reevaluation is less than the value of the **si_xtalk_exit_on_reevaluated_nets_pct** variable.

5. The percentage of nets (relative to the number of cross-coupled nets) selected for reevaluation is less than the value of the **si_xtalk_exit_on_coupled_reevaluated_nets_pct** variable.

6. You manually exit the analysis loop by pressing Control-C to send an interrupt signal to the PrimeTime process. The interrupt is handled as any other exit criteria, at the end of the current iteration of the crosstalk analysis. You cannot interrupt iteration immediately without exiting PrimeTime.

To determine the current value of this variable, type **printvar si_xtalk_exit_on_number_of_reevaluated_nets**.

Pl's Ex. 646 (AP 09.10.rel.1 Executable)

```
set_param si1 xtalk_exit_on_number_of_reevaluated_nets 0
#     Type    : uint
#     Usage   :
```

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793865

si_xtalk_exit_on_reevaluated_nets_pct

Specifies a maximum percentage of nets selected for reevaluation relative to the total number of nets, below which PrimeTime-SI exits the analysis loop.

TYPE

float

DEFAULT

0

DESCRIPTION

Specifies a maximum percentage of nets reselected for evaluation, relative to the total number of nets. PrimeTime-SI exits the analysis loop after completing the current iteration, when the percentage of nets selected for reevaluation in the next iteration is less than this number.

This variable is one of a set of six variables that determine exit criteria; PrimeTime-SI exits the analysis loop after completing the current iteration if one or more of the following is true:

1. The number of iterations performed equals the value of the **xtalk_max_iteration_count** variable.

2. All delta delays fall between the values of the **si_xtalk_exit_on_min_delta_delay** and **si_xtalk_exit_on_max_delta_delay** variables.

3. The number of nets selected for reevaluation in the next iteration is less than the value of the **si_xtalk_exit_on_number_of_reevaluated_nets** variable.

4. The percentage of nets (relative to the total number of nets) selected for reevaluation is less than the value of the **si_xtalk_exit_on_reevaluated_nets_pct** variable.

5. The percentage of nets (relative to the number of cross-coupled nets) selected for reevaluation is less than the value of the **si_xtalk_exit_on_coupled_reevaluated_nets_pct** variable.

6. You manually exit the analysis loop by pressing Control-C to send an interrupt signal to the PrimeTime process. The interrupt is handled as any other exit criteria, at the end of the current iteration of the crosstalk analysis. You cannot interrupt iteration immediately without exiting PrimeTime.

To determine the current value of this variable, type **printvar si_xtalk_exit_on_reevaluated_nets_pct**.

Pl's Ex. 646 (AP 09.10.rel.1 Executable)

```
set_param si1 xtalk_exit_on_reevaluated_nets_pct 0.000
#    Type   : float
#    Usage  :
```

Pl's Ex. 336 (PrimeTime 2006.12) at ATopTech 1793869

si_xtalk_reselect_delta_and_slack

Reselect nets that satisfy both delta delay and slack reselection criteria.

TYPE

Boolean

DEFAULT

false

DESCRIPTION

When true, the intersection of sets of nets reselected by delta delay and slack based criteria is used. For a net to be reselected the following must be true: - The net is reselected by absolute delta delay AND - The net is reselected by relative delta delay AND - The net is reselected by setup OR hold slack OR borrowing AND - Critical path reselection is not enabled.

When true, the nets that satisfy only one of the above criteria (e.g., absolute delta) but not others (e.g., slack) are not counted in the report associated with **XTALK-004**.

When false, the union of sets of nets reselected by delta delay and slack based criteria is used. For a net to be reselected the following must be true: - The net is reselected by absolute delta delay OR - The net is reselected by relative delta delay OR - The net is reselected by setup OR hold slack OR borrowing OR - Critical path reselection is enabled AND the net is on the critical path.

To determine the current value of this variable, type **printvar si_xtalk_reselect_delta_and_slack**.

Pl's Ex. 235 (AP 09.10.rel.1) at ATopTech 0012988

xtalk_reselect_delta_and_slack

Persistent parameter that controls whether window filtering is performed on nets based on the reselection criteria xtalk_reselect_delta_delay, xtalk_reselect_delta_delay_ratio, xtalk_reselect_max_mode_slack, and xtalk_reselect_min_mode_slack.

Syntax:

```
xtalk_reselect_delta_and_slack true | false
```

where the values have the following meaning:

true	Perform window filtering on nets based on the delta delay, max_mode_slack and min_mode_slack criteria.
false	Do not perform window filtering.

The default value is *false*.

Pl's Ex. 6 (PrimeTime 2006.12) at SYNPS-00002587-89			Pl's Ex. 646 (AP 09.10.rel.1 Executable) at ZZATopTech 00000035-44	
Attributes of the cell Object Class				
area	float	The area of a cell. If the cell is hierarchical, this includes net area.	area	: double (read-only) : area
base_name	string	The leaf name of a cell. For example, the base_name of cell U1/U2/U3 is U3.	base_name	: string (read-only) : base name
dont_touch	boolean	Identifies cells to be excluded from optimization in Design Compiler. Cells with the dont_touch attribute set to true are not modified or replaced during compilation in Design Compiler. Setting dont_touch on a hierarchical cell sets the attribute on all cells below it. Set with set_dont_touch, and used by characterize_context and create_timing_context. You can set and unset the dont_touch attribute.	dont_touch	: bool : dont touch in optimization
full_name	string	The complete name of a cell. For example, the full name cell U3 within cell U2 within cell U1 is U1/U2/U3. The full_name attribute is not affected by current_instance.	full_name	: string (read-only) : cell instance fullname
is_sequential	boolean	A cell is sequential if it is not combinational.	is_sequential	: bool (read-only) : is sequential cell
number_of_pins	integer	Number of pins on the cell. The number of pins can be different before and after linking. For example, if some pins were unconnected in a Verilog instance, after linking to the lower-level design, additional pins can be created on the cell.	number_of_pins	: uint (read-only) : number of pins

PI's Ex.18 (PrimeTime 2006.12 Executable) at SNPS_EXE_011				PI's Ex. 646 (AP 09.10.rel.1 Executable) at ZZATopTech 00000035-44
Attributes of the cell Object Class				
is_clock_gating_check	cell	boolean	A	is_clock_gating_check : bool (read-only) : is an ICG or combinational cell with clock gating check
ref_name	cell	string	A	ref_name : string (read-only) : referred lib_cell name

PI's Ex. 6 (PrimeTime 2006.12) at SYNPS-00002592-95			PI's Ex. 646 (AP 09.10.rel.1 Executable) at ZZATopTech 00000035-44
Attributes of the clock Object Class			
`full_name`	`string`	The name of the clock. This is set with `create_clock`. It is either the name given with the `-name` option, or the name of the first object to which the clock is attached. Once set, this attribute is read only.	`full_name : string (read-only)` `: full name`
`period`	`float`	The clock period (or cycle time) is the shortest time during which the clock waveform repeats. For a simple waveform with one rising and one falling edge, the period is the difference between successive rising edges. Set with `create_clock -period`.	`period : float` `: clock period`
`propagated_clock`	`boolean`	Specifies that clock latency (insertion delay) be determined by propagating delays from the clock source to destination register clock pins. If this attribute is not present, ideal clocking is assumed. Set with `set_propagated_clock`.	`propagated_clock : bool` `: compute propagate delay on clock network for latency`
`sources`	`string`	This is a collection of the source pins or ports of the clock. The sources are defined with the `create_clock` command.	`sources : collection (read-only)` `: clock source`

PI's Ex.18 (PrimeTime 2006.12 Executable) at SNPS_EXE_011	PI's Ex. 646 (AP 09.10.rel.1 Executable) at ZZATopTech 00000035-44
Attributes of the clock Object Class	
`is_generated clock boolean A`	`is_generated : bool` `: is generated clock`

Pl's Ex. 6 (PrimeTime 2006.12) at SYNPS-00002605-06			Pl's Ex. 646 (AP 09.10.rel.1 Executable) at ZZATopTech 00000035-44
Attributes of the lib_cell Object Type			
area	float	A floating-point value representing the area of a library cell.	area : double (read-only) : cell area
base_name	string	The name of a library cell. For example, the base_name of library cell tech1/AN2 is AN2.	base_name : string (read-only) : base name
dont_touch	boolean	Identifies library cells to be excluded from optimization. Values are true (the default) or false. Library cells with the dont_touch attribute set to true are not modified or replaced during compile. Set in Design Compiler with set_dont_touch.	dont_touch : bool : do not touch in optimization, equal to (user_dont_touch OR auto_dont_touch)
full_name	string	The fully qualified name of a library cell. This is the name of the library followed by the library cell name. For example, the full_name of library cell AN2 in library tech1 is tech1/AN2.	full_name : string (read-only) : full name
is_sequential	boolean	This attribute is true if the library cell is sequential.	is_sequential : bool (read-only) : is sequential cell
number_of_pins	integer	Number of pins on the library cell.	number_of_pins : uint (read-only) : number of pins

PI's Ex.18 (PrimeTime 2006.12 Executable) at SNPS_EXE_011	PI's Ex. 646 (AP 09.10.rel.1 Executable) at ZZATopTech 00000035-44
Attributes of the lib_cell Object Type	
dont_use lib_cell boolean A	dont_use : bool : do not use in optimization, equal to (user_dont_use OR auto_dont_use)

Pl's Ex. 6 (PrimeTime 2006.12) at SYNPS-00002607-11			Pl's Ex. 646 (AP 09.10.rel.1 Executable) at ZZATopTech 00000035-44	
Attributes of the lib_pin Object Type				
base_name	string	The leaf name of the library cell pin. For example, the base_name of tech1/AN2/Z is Z.	base_name	: string (read-only) : base name
fanout_load	float	A floating-point value representing the fanout load value of a library pin. This value is used in computing max_fanout design rule cost.	fanout_load	: float (read-only) : fanout load
full_name	string	The fully qualified name of a library cell pin. This is the name of the library followed by the library cell name followed by a pin name. For example, the full_name of pin Z on library cell AN2 in library tech1 is tech1/AN2/Z.	full_name	: string (read-only) : full name
max_capacitance	float	A floating-point value representing the maximum capacitance design rule limit for a library pin.	max_capacitance	: float : max capacitance loa
max_fanout	float	A floating-point value representing the maximum fanout design rule limit for a library pin.	max_fanout	: float (read-only) : max fanout
pin_capacitance	float	A floating-point value representing the capacitance of a library pin.	pin_capacitance	: float (read-only) : pin cap

Pl's Ex.18 (PrimeTime 2006.12 Executable) at SNPS_EXE_011	Pl's Ex. 646 (AP 09.10.rel.1 Executable) at ZZATopTech 00000035-44
Attributes of the lib_pin Object Type	
pin_direction lib_pin string A	pin_direction : in out inout internal unknown (read-only) : signal direction

Pl's Ex. 6 (PrimeTime 2006.12) at SYNPS-00002611-14			Pl's Ex. 646 (AP 09.10.rel.1 Executable) at ZZATopTech 00000035-44
Attributes of the net Object Type			
base_name	string	The leaf name of a net. For example, the base name of net i1/i1z1 is i1z1. You cannot set this attribute.	base_name : string (read-only) : base name
dont_touch	boolean	Identifies nets to be excluded from optimization in Design Compiler. Values are true (the default) or false. Nets with the dont_touch attribute set to true are not modified or replaced during compile with Design Compiler. Set with set_dont_touch.	dont_touch : bool : dont touch in optimization
full_name	string	The complete name of a net. For example, the full_name of net i1z1 within cell i1 is i1/i1z1. The full_name attribute is not affected by current instance. The full_name attribute is read-only.	full_name : string (read-only) : full name
total_capacitance_max	float	A floating-point value representing the sum of all pin capacitances and the wire capacitance of a net for maximum conditions. You cannot set this attribute.	total_capacitance_max : float [range: 0.0000 .. inf] (read-only,application) : max total cap
total_capacitance_min	float	A floating-point value representing the sum of all pin capacitances and the wire capacitance of a net for minimum conditions. You cannot set this attribute.	total_capacitance_min : float [range: 0.0000 .. inf] (read-only,application) : min total cap

PI's Ex. 6 (PrimeTime 2006.12) at SYNPS-00002616-25			PI's Ex. 646 (AP 09.10.rel.1 Executable) at ZZATopTech 00000035-44
Attributes of the pin Object Class			
actual_fall_transition_max	float	A floating-point value representing the largest falling transition time for a pin.	actual_fall_transition_max : float [range: -inf .. inf] (read-only,application) : actual max fall transition
actual_fall_transition_min	float	A floating-point value representing the smallest falling transition time for a pin.	actual_fall_transition_min : float [range: -inf .. inf] (read-only,application) : actual min fall transition
actual_rise_transition_max	float	A floating-point value representing the largest rising transition time for a pin.	actual_rise_transition_max : float [range: -inf .. inf] (read-only,application) : actual max rise transition
actual_rise_transition_min	float	A floating-point value representing the smallest rising transition time for a pin.	actual_rise_transition_min : float [range: -inf .. inf] (read-only,application) : actual min rise transition
clocks	string	The collection of clock objects which propagate through this pin. It is undefined if no clocks are present.	clocks : collection (read-only,application) : clocks

direction	string	The direction of a pin. Value can be in, out, inout, or internal. The pin_direction attribute is a synonym for direction. Directions can change as a result of linking a design, as references are resolved.	direction : in out inout internal unknown (read-only) : signal direction
full_name	string	The complete name of a pin to the top of the hierarchy. For example, the full name of pin Z on cell U2 within cell U1 is U1/U2/Z. The setting of the current instance has no effect on the full name of a pin. See also the lib_pin_name attribute.	full_name : string (read-only) : pin full name
is_three_state	boolean	This attribute is true if a pin is a three-state driver.	is_three_state : bool (read-only) : is tri-state

PI's Ex.18 (PrimeTime 2006.12 Executable) at SNPS_EXE_011	PI's Ex. 646 (AP 09.10.rel.1 Executable) at ZZATopTech 00000035-44
Attributes of the pin Object Class	
is_clock_gating_clock pin boolean A	is_clock_gating_clock : bool (read-only) : is the clock pin of an clock gating cell
is_clock_gating_enable pin boolean A	is_clock_gating_enable : bool (read-only) : is the enable pin of an clock gating cell

PI's Ex. 6 (PrimeTime 2006.12) at SYNPS-00002630-42			PI's Ex. 646 (AP 09.10.rel.1 Executable) at ZZATopTech 00000035-44
Attributes of the port Object Class			
actual_fall_transition_max	float	A floating-point value representing the largest falling transition time for a port. You cannot set this attribute.	actual_fall_transition_max : float [range: -inf .. inf] (read-only,application) : actual max fall transition
actual_fall_transition_min	float	A floating-point value representing the smallest falling transition time for a port. You cannot set this attribute.	actual_fall_transition_min : float [range: -inf .. inf] (read-only,application) : actual min fall transition
actual_rise_transition_max	float	A floating-point value representing the largest rising transition time for a port. You cannot set this attribute.	actual_rise_transition_max : float [range: -inf .. inf] (read-only,application) : actual max rise transition
actual_rise_transition_min	float	A floating-point value representing the smallest rising transition time for a port. You cannot set this attribute.	actual_rise_transition_min : float [range: -inf .. inf] (read-only,application) : actual min rise transition
clocks	string	The collection of clock objects which propagate through this port. It is undefined if no clocks are present.	clocks : collection (read-only,application) : clocks
direction	string	The direction of a port. Value can be in, out, inout, or internal. The port_direction attribute is a synonym for direction. You cannot set this attribute.	direction : in out inout internal unknown : signal direction
full_name	string	The name of a port. You cannot set this attribute.	full_name : string (read-only) : full name

PI's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790140

add_to_collection

Adds objects to a collection, resulting in a new collection. The base collection remains unchanged.

SYNTAX

```
collection add_to_collection
base_collection
object_spec
[-unique]
collectionbase_collection
list        object_spec
```

ARGUMENTS

base_collection
: Specifies the base collection to which objects are to be added. This collection is copied to the result collection, and objects matching *object_spec* are added to the result collection. *base_collection* can be the empty collection (empty string), subject to some constraints, explained in the DESCRIPTION.

object_spec
: Specifies a list of named objects or collections to add.
If the base collection is heterogeneous, only collections can be added to it.
If the base collection is homogeneous, the object class of each element in this list must be the same as in the base collection. If it is not the same class, it is ignored. From heterogeneous collections in the *object_spec*, only objects of the same class of the base collection are added. If the name matches an existing collection, the collection is used. Otherwise, the objects are searched for in the database using the object class of the base collection.
The *object_spec* has some special rules when the base collection is empty, as explained in the DESCRIPTION.

-unique
: Indicates that duplicate objects are to be removed from the resulting collection. By default, duplicate objects are not removed.

PI's Ex. 266 (AP 13.11.rel.4) at ATopTech 0055559

add_to_collection

Given a collection *base_collection* and one or more *objects*, returns a new collection with all objects of the base collection and all the listed objects. In addition, you can choose to filter out all duplicate objects from the new collection. The base collection is not modified.

Syntax

```
add_to_collection \
   base_collection \
   objects \
   [-unique]
```

where the arguments have the following meaning:

base_collection	The collection to which you want to add an object.
objects	The list of objects you want to add.
[-unique]	Removes duplicate objects from the resulting collection.

Case No. 3:13-cv-02965-MMC
PLNTF Exhibit No. 1441
Date Entered FEB 29 2016
Signature TRACY LUCERO

all_connected

Creates a collection of objects connected to a net, pin, or port object. You can assign this collection to a variable or pass it into another command.

SYNTAX

```
collection all_connected object_spec [-leaf]
list object_spec
```

ARGUMENTS

object_spec
: Specifies the object whose connections are returned. This is a collection of one element which is a net, pin, or port collection, or the name of a net, pin, or port.

-leaf
: Specifies that the connections of the net that are being returned should be global or leaf pins. When specified, this gives the leaf pins of a hierarchical net. For non-hierarchical nets, there is no difference in output.

all_connected

Returns objects connected to a specified set of other objects. For each net, the command retrieves all pins connected to the net, and for each pin, the command returns all nets connected to the pin. Furhtermore, you can limit the returned objects based on their name or based on an Aprisa filter. This filter accepts logic expressions of attributes. Only objects whose attributes satisfy the filter expression are retained.

Syntax

```
all_connected objects \
   [-all_functions] \
   [-quiet] \
   [-regexp] \
   [-nocase] \
   [-filter string] \
   [-leaf]
```

where the argument has the following meaning:

objects	List of nets for which to return the pins and ports, or list of pins and ports for which to return the net, or a combination of pins, nets, and ports. If the *-regexp* argument is specified, then *objects* is a regular expression. The connected objects are returned of all pins, ports, and nets whose names match the regular expression.
[-all_functions]	Return the pins of all functions (power, ground, or diode). By default, only connections of type signal are considered.
[-quiet]	Suppress the messages.

2

	[-regexp]	*objects* is a regular expression. The connected objects of all objects whose names match the regular expression are returned.
	[-nocase]	*objects* is case insensitive. This argument is only relevant if the *-regexp* argument is used.
	[-filter *expression*]	Use the logic expression of attributes to only return those objects whose attributes satisfy the expression.
	[-leaf]	Return leaf pins only.

3

all_fanin

Creates a collection of pins/ports or cells in the fanin of specified sinks.

SYNTAX

```
collection all_fanin -to sink_list
[-flat] [-only_cells]
[-startpoints_only]
[-levels level_count]
[-pin_levels pin_count]
[-step_into_hierarchy]

list  sink_list
int   level_count
```

ARGUMENTS

-to *sink_list*
: Specifies a list of sink pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. The timing fanin of each sink in *sink_list* becomes part of the resulting collection. If a net is specified, the effect is the same as listing all driver pins on the net. This argument is required.

-startpoints_only
: When this option is specified, only the timing startpoints will be included in the result.

-only_cells
: The result will include only cells in the timing fanin of the *sink_list* and not pins or ports.

-flat
: There are two major modes in which **all_fanin** functions: hierarchical (the default) and flat. When in hierarchical mode, only objects within the same hierarchical level as the current sink are included in the result. In flat mode, the only non-leaf objects in the result will be hierarchical sink pins.

-levels cell_count
: The traversal will stop when reaching a depth of search of *cell_count* hops, where the counting is performed over the layers of cells of same distance from the sink.

-pin_levels pin_count
: The traversal will stop when reaching a depth of search of *pin_count* hops, where the counting is performed over the layers of pins of same distance from the sink.

-step_into_hierarchy
: This option may only be used in hierarchical mode and only has effect with either **-levels** or **-pin_levels**. Without the switch, a hierarchical block at the same level of hierarchy as the current sink is considered to be a cell;

all_fanins

Retrieves either all pins and ports or cells that belong to the fan-in cone of one or more specified pins. Pins that are referenced in timing constraints, for example, in case-analysis, are also considered part of the fan-in cone.

Syntax

```
all_fanins -to collection \
   [-only_cells] \
   [-flat] \
   [-startpoints_only]\
   [-level cellLevel] \
   [-pin_level pinLevel]
```

where the arguments have the following meaning:

-to *collection*	Set of pins and ports whose fan-in cone is retrieved.
[-only_cells]	Return the cells of the fanin cone instead of the pins and ports.
[-flat]	Hierarchical pins, that is, pins that do not exist in a flattened netlist must not be returned.
[-startpoints_only]	Only return pins and ports at the start of the fan-in cones.
[-level *cellLevel*]	Include pins of all cells that are at a distance up to *cellLevel* cell arcs from the root pin.
[-pin_level *pinLevel*]	Include all pins that are at a distance up to *pinLevel* pin arcs from the root pin.

all_fanout

Creates a collection of pins/ports or cells in the fanout of the specified sources.

SYNTAX

```
collection all_fanout -from source_list
-clock_tree [-flat]
[-only_cells] [-endpoints_only]
[-levels level_count]
[-pin_levels pin_count]
[-step_into_hierarchy]

list  source_list
int   level_count
```

ARGUMENTS

-from *source_list*
Specifies a list of source pins, ports, or nets in the design. Each object is a named pin, port, or net, or a collection of pins, ports, or nets. The timing fanout of each source in *source_list* becomes part of the resulting collection. If a net is specified, the effect is the same as listing all load pins on the net. This option is exclusive with the **-clock_tree** option.

-clock_tree
Indicates that all clock source pins and/or ports in the design are to be used as the list of sources. Clock sources are specified using **create_clock**. If there are no clocks, or if the clocks have no sources, the result is the empty collection. This option is exclusive with the **-from** option.

-endpoints_only
When this option is specified, only the timing endpoints will be included in the result.

-only_cells
The result will include only cells in the timing fanout of the *source_list* and not pins or ports.

-flat
There are two major modes in which **all_fanout** functions: hierarchical (the default) and flat. When in hierarchical mode, only objects within the same hierarchical level as the current source are included in the result. In flat mode, the only non-leaf objects in the result will be hierarchical source pins.

-levels cell_count
The traversal will stop when reaching a depth of search of *cell_count* hops, where the counting is performed over the layers of cells of same distance from the source.

all_fanouts

Retrieves either all pins and ports or cells that belong to the fan-out cone of one or more specified pins. Pins that are referenced in timing constraints, such as the effect of disable timing or case-analysis, are also considered part of the fan-out cone.

Syntax

```
all_fanouts -from collection \
   -only_cells \
   -flat \
   -endpoints_only \
   [-level cellLevel] \
   [-pin_level pinLevel]
```

where the arguments have the following meaning:

-from *collection*	Set of pins and ports whose fan-out cone is retrieved.
-only_cells	Return the cells instead of the pins and ports in the fan-out cone.
-flat	Hierarchical pins, that is, pins that do not exist in a flattened netlist, must not be returned.
-endpoints_only	Only return pins and ports that are legal timing end points. Driver pins, floating pins and pins connected through disabled timing arcs are not included.
[-level *cellLevel*]	Include pins of all cells that are at a distance up to *cellLevel* cell arcs from the root pin.
[-pin_level *pinLevel*]	Include all pins that are at a distance up to *pinLevel* pin arcs from the root pin.

-pin_levels pin_count
The traversal will stop when reaching a depth of search of *pin_count* hops, where the counting is performed over the layers of pins of same distance from the source.

-step_into_hierarchy
This option may only be used in hierarchical mode and only has effect with either **-levels** or **-pin_levels**. Without the switch, a hierarchical block at the same level of hierarchy as the current sink is considered to be a cell; the output pins are considered a single level away from the related input pins, regardless of what is inside the block. With the switch enabled, the counting is performed as though the design were flat, and although pins inside the hierarchy are not returned, they determine the depth of the related input pins.

6

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790153	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0055601
all_instances Creates a collection of all instances of a specific design (or library cell) in the current design, relative to the current instance. You can assign the resulting collection of cells to a variable or pass it into another command.	**all_instances** Given a collection of library cells and modules, returns all instances of those cells, or given a name pattern, returns all instances of cells and modules whose name match the name pattern.
SYNTAX collection **all_instances** [-hierarchy] *object_spec*	**Syntax** all_instances *object* \ [-hierarchical] \ [-quiet]
ARGUMENTS -hierarchy Searches for instances in all levels of instance hierarchy below the current instance. By default, only instances from the current level of hierarchy are considered. object_spec Specifies the target design or library cell. This can be a design collection, a lib_cell collection, or a name.	where the arguments have the following meaning: *object* — Collection of library cells, modules, or names of cells and modules for which to return all instances. [-hierarchical] — Match cells at any level in the hierarchy. [-quiet] — Suppress the messages.

7

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790161

append_to_collection

Add object(s) to a collection. Modifies variable.

SYNTAX

```
collection add_to_collection
var_name
object_spec
[-unique]
collection var_name
list     object_spec
```

ARGUMENTS

var_name
: Specifies a variable name. The objects matching *object_spec* are added into the collection referenced by this variable.

object_spec
: Specifies a list of named objects or collections to add.

-unique
: Indicates that duplicate objects are to be removed from the resulting collection. By default, duplicate objects are not removed.

Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0055614

append_to_collection

Appends the specified collection *object_collection* to the collection held by the collection variable *base_collection*. If this variable does not exist, this command creates it. You can choose to remove the duplicate objects from the resulting collection.

Syntax

```
append_to_collection base_collection \
   object_collection \
   [-unique]
```

where the arguments have the following meaning:

base_collection	The variable that holds the original collection. Is created if needed.
object_collection	A collection of objects to add to the original collection.
[-unique]	Removes duplicate objects from the resulting collection stored in the variable *base_collection*.

8

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790169	Pl's Ex. 652 (AP 13.11.rel.1 Executable) ZZATopTech 0000035-44

characterize_context

Captures the timing context of a list of instances.

Command: characterize_context <db:module_inst_list>
  Derive timing constraints for a set of cell instances.

SYNTAX

string **characterize_context** [-timing] [-environment]
[-design_rules]
[-constant_inputs]
[-no_boundary_annotations]
cell_list

list *cell_list*

option:

-timing	not supported yet
-environment	not supported yet
-design_rules	not supported yet
-constant_inputs	not supported yet
-no_boundary_annotations	not supported yet
-output_dir string(.)	directory to write derived sdc
--get_option arg<1>	get option value
--set_option ...	set option value
--get_default arg<1>	get default value
--set_default ...	set default value
--system_default	use system default values
--list_options	list current option values
--load_options ...	load current option values
--license	list required licenses
--help	display command help

ARGUMENTS

-timing
Characterizes timing information; for example, clocks, input and output delays, and timing exceptions.

-environment
Characterizes environment-related information; for example, operating conditions (process, temperature, and voltage), wire load model, capacitive loads on input and output pins, and driving cell information on input pins.

-design_rules
Characterizes design rules; for example, max_capacitance, max_transition, and max_fanout.

-constant_inputs
Characterizes logic constants propagated to input pins of the instance being characterized by the case analysis capability of PrimeTime.

-no_boundary_annotations
Disables characterization of annotated capacitance on boundary nets as annotated capacitance in the characterized instance. Instead, the port wire capacitance is adjusted to account for any difference between the estimated and annotated values. By default, PrimeTime characterizes annotated capacitance on boundary nets as annotated capacitance in the characterized instance.

cell_list
Specifies a list of instances to characterize.

9

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790179

check_timing

Shows possible timing problems for design.

SYNTAX

```
string check_timing [-verbose]
[-significant_digits digits]
[-ms_min_separation delta]
[-override_defaults check_list]
[-include check_list]
[-exclude check_list]

float delta
int   digits
list    check_list
```

ARGUMENTS

-verbose
Shows detailed information about potential problems.

-significant_digits *digits*
Specifies the number of digits of precision to be displayed by warnings that show floating point numbers. Allowed values are 0-13; the default is determined by the **report_default_significant_digits** variable, whose default value is 2. Use this option if you want to override the default.

-ms_min_separation *delta*
Minimum separation value between master and slave clocks. The default minimum separation is 0.0.

-override_defaults *check_list*
Overrides the checks in **timing_check_defaults** using *check_list*. See the man page of **timing_check_defaults** for its default value.

-include *check_list*
Adds the checks listed in *check_list* to the checks in **timing_check_defaults**.

-exclude *check_list*
Subtracts the checks listed in *check_list* from the checks in **timing_check_defaults**.

check_list
Gives the list of checks to be performed. Each element in this list is one of the following strings: clock_crossing, data_check_multiple_clock, data_check_no_clock, generated_clocks, generic, latch_fanout, latency_override, loops, ms_separation, multiple_clock, no_clock, no_input_delay, retain, signal_level, unconstrained_endpoints.

Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0055801

check_timing

Reports timing problems in the design. The following timing violations are reported:

- latch_fanout—A latch fans out to itself or to another latch connected to the same clock.
- no_clock—No clock reaches a sequential cell's clock pin.
- no_driving_cell—A port does not have a driving cell constraint.
- no_input_delay—An input port does not have an input delay constraint.
- unconstrained_endpoints— An endpoint of a timing path, such as an output port or data pin of a sequential cell, does not have a timing constraint set.

Syntax

```
check_timing
    [-verbose] \
    [-extra] \
    [-scenario name]
```

where the arguments have the following meaning:

[-verbose]	In the report, include the names of the pins and ports with timing constraint problems.
[-extra]	In addition to the regular timing checks, also check the completeness of the timing arcs in the Liberty files.
[-scenario *name*]	Specifies the scenario file to use. The scenario file contains the conditions under which to analyze the design such as Process-Voltage-Temperature corners. If you do not provide a scenario name, the timing checks are based on the SDC and Liberty file set for the current design.

10

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790192	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0055829
compare_collections Compares the contents of two collections. If the same objects are in both collections, the result is "0" (like string compare). If they are different, the result is nonzero. The order of the objects can optionally be considered.	**compare_collections** Compares collections. If both collections contain the same objects, the command returns *0*; otherwise, it returns *-1*. You can control whether the order of objects should be taken into account.

SYNTAX
 	
`int` **`compare_collections`** `[-order_dependent]` *`collection1 collection2`*
`collection` *`collection1`*
`collection` *`collection2`*
	Syntax
	
	`compare_collections` *`collection1 collection2`* `\`
	`  [-order_dependent]`

ARGUMENTS
 	
`-order_dependent`
Indicates that the order of the objects is to be considered; that is, the collections are considered to be different if the objects are ordered differently.
 	
`collection1`
Specifies the base collection for the comparison. The empty string (the empty collection) is a legal value for the *collection1* argument.
 	
`collection2`
Specifies the collection with which to compare to *collection1*. The empty string (the empty collection) is a legal value for the *collection2* argument.
	where the arguments have the following meaning:
	
	`collection1` `and` *`collection2`* — Collections to compare.
	
	`[-order_dependent]` — Indicates that collections are only considered identical if the order of the objects in the collections is the same.

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790201	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0055846
connect_net Connects a net to specified pins or ports.	**connect** Connects port and pins to a net. If the port and pins are already connected to other nets, you must explicitly specify that you want to reconnect them; otherwise, you get an error. If you want to connect a port to a net that is already connected a port, you must have marked these ports and the net as a feedthrough upon creation, and you must mark this connection as a feedthrough as well. By default, the *connect* Tcl command honors the *dont_touch* attribute set on a net and issues the *ChgDontTouch* error when you try to connect to such a net. You can however force a connection.
SYNTAX int **connect_net** *net object_spec* string*net* list *object_spec*	**Syntax** connect -net *net_name* \ *pins_and_ports* \ [-reconnect]\ [-feedthru] \ [-force]
ARGUMENTS net Specifies the name of the net to which the pins and ports are to be connected. object_spec Specifies a list of pins or ports to connect to *net*.	where the arguments have the following meaning: -net *net_name* — Name of the net to which you want to connect the pins and ports. *pins_and_ports* — Names of the pins and ports to connect to the net. By default, only pins and ports that are not yet connected to a net are allowed to be connected. Use the *-reconnect* argument to allow all pins and ports to be connected to this net. [-reconnect] — Allows you to specify pins and ports that are already connected to nets. Those pins and ports are disconnected from their original net and reconnected to the specified net.

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790204	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0055898
copy_collection Duplicates the contents of a collection, resulting in a new collection. The base collection remains unchanged.	**copy_collection** Returns a new collection that contains the same objects as a specified collection. Note that the objects are not copied, only the collection of those objects.
SYNTAX collection **copy_collection** *collection1* collection *collection1*	**Syntax** copy_collection *collection* where *collection* are the objects you want to copy.
ARGUMENTS collection1 Specifies the collection to be copied. If the empty string is used for the *collection1* argument, the command returns the empty string (a copy of the empty collection is the empty collection).	

13

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790213	Pl's Ex. 602 (AP 13.11.rel.4 Executable) at ATopTech 0057938
create_configuration Creates a configuration for multi-scenario analysis.	Command: create_configuration Don't support.
SYNTAX **create_configuration** -global_data *file_list* list *file_list*	-global_data { <string> ... } order dependent setup files (require) --get_option arg<1> get option value --set_option ... set option value --get_default arg<1> get default value --set_default ... set default value --system_default use system default values --list_options list current option values --load_options ... load current option values --license list required licenses --help display command help
ARGUMENTS -global_data An order dependent list of files containing everything that is common across the entire analysis. These files must at the very minimum contain the commands needed to read the design netlist.	

14

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790240-41	Pl's Ex. 602 (AP 13.11.rel.4 Executable) at ATopTech 0057938
create_operating_conditions Creates a new set of operating conditions in a library.	Command: create_operating_conditions standard SDC command
SYNTAX int **create_operating_conditions** -name *name* -library *library_name* -process *process_value* -temperature *temperature_value* -voltage *voltage_value* [-tree_type *tree_type*] [-calc_mode *calc_mode*] [-rail_voltages *rail_value_pairs*] string *name* string *library_name* float *process_value* float *temperature_value* float *voltage_value* string *tree_type* string *calc_mode* Tcl list *rail_value_pairs*	option: -name string — name of operating condition (require) -library string — name of library (require) -process double(0.000) — process scaling factor (require) -temperature double(0.000) — temperature value (require) -voltage double(0.000) — voltage value (require) -tree_type tree_type(balanced_tree) — tree type tree_type = balanced_tree | best_case_tree | worst_case_tree -calc_mode * — not supported yet -rail_voltage * — not supported yet --get_option arg<1> — get option value --set_option ... — set option value --get_default arg<1> — get default value --set_default ... — set default value --system_default — use system default values --list_options — list current option values --load_options ... — load current option values --license — list required licenses --help — display command help

15

ARGUMENTS

-name *name*
Specifies the name of the new set of operating conditions.

-library *library_name*
Specifies the name of the library for the new operating conditions.

-process *process_value*
Specifies the process scaling factor for the operating conditions. Allowed values are 0.0 through 100.0.

-temperature *temperature_value*
Specifies the temperature value, in degrees Celsius, for the operating conditions. Allowed values are -300.0 through +500.0.

-voltage *voltage_value*
Specifies the voltage value, in volts, for the operating conditions. Allowed values are 0.0 through 1000.0.

-tree_type *tree_type*
Specifies the tree type for the operating conditions. Allowed values are *best_case_tree*, *balanced_tree* (the default), or *worst_case_tree*. The tree type is used to estimate interconnect delays by providing a model of the RC tree.

-calc_mode *calc_mode*
For use only with DPCM libraries. Specifies the DPCM delay calculator mode for the operating conditions; analogous to the *process* used in Synopsys libraries. Allowed values are *unknown* (the default), *best_case*, *nominal*, or *worst_case*. The default behavior (*unknown*) is to use worst case values during

analysis similarly to worst_case. If **-rail_voltages** are specified, the command sets all (worst_case, nominal, and best_case) voltage values.

-rail_voltages *rail_value_pairs*
Specifies a list of name-value pairs that defines the voltage for each specified rail. The name is one of the rail names defined in the library; the value is the voltage to be assigned to that rail. By default, rail voltages are as defined in the library; use this option to override the default voltages for specified rails.

description:
This command is the same as standard SDC command.

16

PI's Ex. 18 (PrimeTime 2006.12 Executable) at SNPS_EXE_11	PI's Ex. 266 (AP 13.11.rel.4) at ATopTech 0055996-97
define_proc_attributes # Add extensions to a procedure	**define_proc_attributes** Describes the help text of a specified Tcl procedure and describes the attributes of its arguments. This Tcl command allows you to use Aprisa's *parse_proc_arguments* Tcl procedure in your procedure to parse its arguments. By using both the *define_proc_attributes* and *parse_proc_arguments* Tcl commands, you integrate your procedure in the Aprisa environment. The meta arguments, such as *--h*, are enabled and the *info* and *help* commands also work for your procedure.
[-info info_text] (Help string for the procedure) [-define_args arg_defs] (Procedure argument definitions for verbose help) [-command_group group_name] (Command group for procedure. Default: Procedures) [-permanent] (Procedure cannot be overwritten) [-hide_body] (Body cannot be viewed with 'info body') [-hidden] (Procedure does not show up in help or info) [-dont_abbrev] (Procedure can never be abbreviated) name (Procedure name)	**Syntax** define_proc_attributes \ *procedureName* \ -info *string* \ -define_args * \
	where the arguments have the following meaning: *procedureName* — Name of the procedure. -info string — One-line help string for the procedure. -define_args * — Arguments and options of the procedure. This is a collection of argument definitions. Each argument definition has the following format: {*arg_name option_help value_help data_type attributes*} where: *arg_name* — Name of the argument. If the name starts with a '-', it indicates a named argument. Otherwise, it is a positional argument. *option_help* — Help string describing the argument. *value_help* — Help string describing the acceptable values for the argument *data_type* — Type of value expected for this argument, such as *float*, *string*, *boolean*, *one_of_string* *attributes* — Additional attributes, such as *required*, *optional*, and for types *one_of_string*, the list of *values*.

17

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790289	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0055999
define_user_attribute Defines a new user-defined attribute.	**define_user_attribute** Defines a user attribute. You must define an attribute before using it. User attributes, similar to Aprisa attributes, have a name, a type, and can only be attached to objects of the specified type. You can create, modify, and delete user attributes. Attribute names must be unique within the class for which the attribute is defined, and the value of the attribute can only be of one data type.
SYNTAX string **define_user_attribute** -type *data_type* -classes *class_list* [-range_min *min*] [-range_max *max*] [-one_of *values*] [-import] [-quiet] *attr_name* string *data_type* list *class_list* double *min* double *max* list *values* string *attr_name*	**Syntax** define_user_attribute *attr_name* \ -type int | float | string | point | bool \ -class *class*
ARGUMENTS -type *data_type* Specifies the data type of the attribute. The supported data types are string, int, float, double, and boolean. -classes *class_list* Defines the attribute for one or more of the classes. The valid object classes are design, port, cell, pin, net, lib, lib_cell or lib_pin. -range_min *min* Specifies *min* value for numeric ranges. This is only valid when the *data_type* is int or double. Specifying a minimum constraint without a maximum constraint creates an attribute which accepts a value = *min*. -range_max *max* Specifies *max* value for numeric ranges. This is only valid when the *data_type* is int or double. Specifying a maximum constraint without a minimum constraint creates an attribute which accepts a value = *max*. -one_of *values* Provides a list of allowable strings. This is only valid when the data type is string. -import Import this attribute from a design or library database. -quiet Does not report any messages. attr_name Specifies the name of the attribute.	where the arguments have the following meaning: *attr_name* — Attribute name. -type int | float | string | point | bool — Data type of its value. -class *class* — Class name of object for which it is defined.

18

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790292	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056041
derive_clocks Creates clocks on source pins in design.	**derive_clocks** Creates clock definitions for all missing clocks so that design registers are constrained. These generated clocks are specified by a clock period and a set of times when clock edges occur, similar to regular clocks. The main difference is that the missing clocks are generated by Aprisa as opposed to being defined as part of the SDC constraints.

SYNTAX
 	
`string` **`derive_clocks`** `-period` *`period_value`* `[-waveform` *`edge_list`*`]`
`float` *`period_value`*
`list` *`edge_list`*
	Syntax
	
	`derive_clocks \`
	`  -period` *`period`* `\`
	`  -waveform` *`times`*

ARGUMENTS `-period` *`period_value`* Specifies the clock period of the automatically derived clocks. The clock period has a value greater than or equal to zero (value = 0). `-waveform` *`edge_list`* Specifies the rise and fall edge times of the clock, in library time units, over an entire clock period. It defines the clock edge specification. The first time that is listed is a rising transition; typically the first rising transition after time zero. There must be an even number of increasing times and alternating rise and fall times. If you do not specify an *`edge_list`* value, the command assumes a default waveform that has a rise edge of **`0.0`** and a fall edge of *`period_value`*`/2`.	where the arguments have the following meaning: `-period` *`period`* — Clock period for all missing clocks. `-waveform` *`times`* — Clock waveform, specified as a collection of times at which clock edges occur, starting with a rising edge.

19

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790302	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056151
filter The **filter** command, a synonym for the **filter_collection** command, is a DC Emulation command provided for compatibility with Design Compiler.	**filter** This command is aliased to filter_collection.

20

filter_collection

```
Filters an existing collection, resulting in a new collection. The base collection
remains unchanged.
```

filter_collection

Retains from a given collection of objects only those objects that meet the specified criteria. Criteria are formulated as logic and pattern-matching expressions of attributes and values.

The following operators are supported:

```
==   : equal
!=   : not equal
=~   : match pattern
!~   : not match pattern
<    : less than
<=   : less or equal
>    : greater than
>=   : greater or equal
&&   : Logic AND
||   : Logic OR
```

The pattern matching syntax can be the Tcl regular expression or the Tcl globbing (also known as wildchar) syntax.

You can use both Aprisa and user-defined attributes in filter expressions. Before using user-defined attributes, you must have defined them using the *define_user_attribute* Tcl command.

Moreover, you can also filter objects based on the existance of user attributes on these objects.

For convenience, many Aprisa commands, such as all the *get_** Tcl commands, have a *-filter* argument you can use to filter objects based on a Tcl regular or globbing expression.

SYNTAX

collection **filter_collection**
base_collection expression
[-regexp]
[-nocase]

collection*base_collection*
string *expression*

Syntax

filter_collection *collection expression* \
[{ *attributename* -exists }] \
[-regexp] \
[-nocase]

21

ARGUMENTS

base_collection
: Specifies the base collection to be filtered. This collection is copied to the result collection. Objects are removed from the result collection if they are evaluated as **false** by the conditional *expression* value. Substitute the collection you want for *base_collection*.

expression
: Specifies an expression with which to filter *base_collection*. Substitute the string you want for *expression*.

-regexp
: Specifies that the =~ and !~ filter operators will use real regular expressions. By default, the =~ and !~ filter operators use simple wildcard pattern matching with the * and ? wildcards.

-nocase
: Makes the pattern match case-insensitive. When you specify this option, you must also specify the **-regexp** option.

where the arguments have the following meaning:

collection	Collection on which the filter criteria is applied. Only objects that meet these criteria are returned by the filter.
expression	Expression using constants, object attributes, and the operators listed above.
[{ *attributename* -exists }]	Only retain objects that have an attribute of the specified name.
[-regexp]	Use Tcl regular expression syntax for the pattern. The default is Tcl globbing syntax.
[-nocase]	Expression is case insensitive.

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790307	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056171
foreach_in_collection Iterates over the elements of a collection.	**foreach_in_collection** Executes a set of Tcl commands on each object from a given Aprisa collection. This command is equivalent to the *foreach* Tcl command but has the advantage that it operates directly on an Aprisa collection, which is much more efficient than a Tcl list.
SYNTAX string **foreach_in_collection** *itr_var collections body* string*itr_var* list *collections* string*body*	**Syntax** foreach_in_collection *object collection* { body }
ARGUMENTS itr_var Specifies the name of the iterator variable. collections Specifies a list of collections over which to iterate. body Specifies a script to execute per iteration.	where the arguments have the following meaning: *object* — Tcl variable containing the object on which the body of Tcl commands is operated. *collection* — Collection of objects on which the body of Tcl commands is operated. This can also be a Tcl expression returning a collection. { *body* } — Set of Tcl commands that is executed on each object in *collection*, one at a time.

23

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790312	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056210

get_attribute

Retrieves the value of an attribute on an object.

SYNTAX

```
string get_attribute [-class class_name] [-quiet] object_spec attr_name
stringclass_name
stringobject_spec or
collection object_spec
stringattr_name
```

ARGUMENTS

-class *class_name*
Specifies the class name of *object_spec*, if *object_spec* is a name. Valid values for *object_spec* are *design*, *port*, *cell*, *pin*, *net*, *lib*, *lib_cell*, *lib_pin*, *clock*, *timing_path*, and *timing_point*. You must use this option if *object_spec* is a name.

-quiet
Indicates that any error and warning messages are not to be reported.

object_spec
Specifies a single object from which to get the attribute value. *object_spec* must be is either a collection of exactly one object, or a name which is combined with the *class_name* to find the object. If *object_spec* is a name, you must also use the **-class** option.

attr_name
Specifies the name of the attribute whose value is to be retrieved.

get_attribute

Returns the value of the specified attribute of an object or a collection of objects. If you pass a collection of objects, then a collection of attribute values is returned.

The command can also be used to check whether the attribute exists on that object.

Syntax

```
get_attribute object_or_collection attr_name \
   [-class cell | net | port | pin | lib_cell | lib_pin] \
   [-exist] \
   [-quiet]
```

where the arguments have the following meaning:

object_or_collection	Object or collection of objects whose attribute value you want to retrieve.
attr_name	Name of the attribute to retrieve.
[-class cell | net | port | pin | lib_cell | lib_pin]	Only examine objects of the specified type.
[-exist]	Check whether the object attribute exists. If you specify a collection of objects, then the function only returns true provided all objects in the collection have the attribute.
[-quiet]	Prevents an error to be issued if an attribute is not found.

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790326

get_generated_clocks

Creates a collection of generated clocks.

SYNTAX

```
collection get_generated_clocks [-quiet] [-regexp] [-nocase] [-filter expression]
patterns
stringexpression
list  patterns
```

ARGUMENTS

-quiet
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp
Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase
When combined with **-regexp**, makes matches case-insensitive. You can use **-nocase** only when you also use **-regexp**.

-filter *expression*
Filters the collection with *expression*. For any generated clocks that match *patterns*, the expression is evaluated based on the generated clock's attributes. If the expression evaluates to true, the generated clock is included in the result.

patterns
Matches generated clock names against patterns. Patterns can include the wildcard characters "*" and "?".

Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056233

get_generated_clocks

Returns a collection of generated clock objects. The returned set of generated clocks may be selected by name, by the Aprisa attribute-based object filter, or by a combination of these.

Syntax

```
get_generated_clocks clockpattern \
   [-filter attribute_constraint] \
   [-regexp] \
   [-nocase]
```

where the arguments have the following meaning:

clockpattern	Only return clocks whose names match the pattern. By default, Tcl globbing syntax is assumed.
[-filter *attribute_constraint*]	Only return generated clocks whose attributes meet the constraints defined in *attribute_constraint*. For more information on the syntax of the attribute constraints, see the *filter_collection* Tcl command.
[-regexp]	Treat the name patterns in *attribute_constraint* as a regular expression. By default, Tcl globbing syntax is assumed.
[-nocase]	Ignore case when performing name matches in *attribute_constraint*.

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790350	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056275
get_object_name Gets the name of the object in a collection of exactly one object.	**get_object_name** Returns the name of the specified object, or returns a collection of names of a specified collection of objects.
SYNTAX string **get_object_name** *collection* string*collection*	**Syntax** get_object_name *object* | -multiple *objects*
ARGUMENTS collection Specifies the collection. This must be a collection of exactly one object.	where the arguments have the following meaning: *object* — Object whose name you want to retrieve. -multiple — Controls whether the command expects a single object or a collection of objects. *objects* — Collection of objects whose name you want to retrieve.

26

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790351	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056279

get_path_groups

Creates a collection of path groups from the current design. You can assign these path groups to a variable or pass them into another command.

SYNTAX

```
collection get_path_groups [-quiet] [-regexp] [-nocase] [-filter expression]
patterns
stringexpression
list  patterns
```

ARGUMENTS

-quiet
Suppresses warning and error messages if no objects match. Syntax error messages are not suppressed.

-regexp
Views the *patterns* argument as real regular expressions rather than simple wildcard patterns. Also, modifies the behavior of the =~ and !~ filter operators to compare with real regular expressions rather than simple wildcard patterns.

-nocase
When combined with **-regexp**, makes matches case-insensitive. You can use **-nocase** only when you also use **-regexp**.

-filter *expression*
Filters the collection with *expression*. For any path groups that match *patterns*, the expression is evaluated based on the path group's attributes. If the expression evaluates to true, the path group is included in the result.

patterns
Matches path group names against patterns. Patterns can include the wildcard characters "*" and "?".

get_path_groups

Returns the list of names of all path groups given a glob-style pattern or a regular expression.

Syntax

```
get_path_groups clock_names \
   [-quiet] \
   [-regexp] \
   [-nocase]
```

where the values have the following meaning:

clock_names	Name or regular expression. All path groups with a launching clock or a capturing clock that have a name matching the regular expression are returned.
[-quiet]	Do not report errors or warnings.
[-regexp]	Treat *clock_names* as a regular expression.
[-nocase]	Do not consider case when matching clock names to the regular expression *clock_names*.

NOTE: This command returns a list of strings and not a collection of objects as most other *get** Tcl commands do. For this reason, no *-filter* argument is supported.

27

get_timing_paths

Creates a collection of timing paths for custom reporting and other processing. You can assign these timing paths to a variable or pass them into another command.

SYNTAX

```
string get_timing_paths
[-from from_list
   | -rise_from rise_from_list
   | -fall_from fall_from_list]
[-to to_list
   | -rise_to rise_to_list
   | -fall_to fall_to_list]
[-exclude exclude_list
   | -rise_exclude rise_exclude_list
   | -fall_exclude fall_exclude_list]
[-through through_list]
[-rise_through rise_through_list]
[-fall_through fall_through_list]
[-delay_type delay_type]
[-nworst paths_per_endpoint]
[-max_paths max_path_count]
[-group group_name]
[-true]
[-unique_pins]
[-true_threshold path_delay]
[-slack_greater_than greater_slack_limit]
[-slack_lesser_than lesser_slack_limit]
[-ignore_register_feedback feedback_slack_cutoff]
[-aocvm]
[-include_hierarchical_pins]
[-justify]
[-trace_latch_borrow]
[-recalculate]
[-start_end_pair]
[-dont_merge_duplicates]
[-pre_commands pre_command_string]
[-post_commands post_command_string]
[-path_type format]
```


get_timing_paths

Returns a collection of timing paths for custom processing.

A timing path is of object type *timing_path*. You can list all the attribute names of the object using the *list_attribute* Tcl command with the *-class timing_path* argument.

To list one or all attribute values, use the *get_attribute* or *report_attribute* Tcl command.

To iterate timing paths in a collection, use the *foreach_in_collection* Tcl command.

One attribute of a *timing_path* object is the *points* collection, which consists of multiple timing points. A timing point is of object type *timing_point*. You can list, report, and get attributes of a *timing_point* object using the *list_attribute*, *report_attribute*, and *get_attribute* Tcl commands, respectively.

Syntax

```
get_timing_paths objects\
   [-from pinlist] \
   [-rise_from pinlist] \
   [-fall_from pinlist] \
   [-to pinlist] \
   [-rise_to pinlist] \
   [-fall_to pinlist] \
   [-through pinlist] \
   [-rise_through pinlist] \
   [-fall_through pinlist] \
   [-delay_type max | min | min_max | max_rise | max_fall | \
      min_rise | min_fall] \
   [-nworst integer] \
   [-max_paths integer] \
   [-path_type full | full_clock | full_clock_expanded] \
   [-slack_greater_than double] \
   [-slack_less_than double] \
   [-groups {string [string]...}] \
   [-scenario string] \
   [-no_hierarchical_pins] \
```

```
list  from_list
list  rise_from_list
list  fall_from_list
list  to_list
list    rise_to_list
list    fall_to_list
list    exclude_list
list    rise_exclude_list
list    fall_exclude_list
list  through_list
list  rise_through_list
list  fall_through_list
stringdelay_type
stringformat
int   paths_per_endpoint
int   max_path_count
list  group_name
float   path_delay
float   greater_slack_limit
float   lesser_slack_limit
float   feedback_slack_cutoff
```

ARGUMENTS

-from *from_list*
Specifies a list of from pins, ports, nets, or clocks to be reported. Path startpoints are typically the input ports or clock pins of registers. If you specify a clock, all startpoints are considered if they are clocked by the clock.

-rise_from *rise_from_list*
Same as the **-from** option, except that the path must rise from the objects specified. If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by rising edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-fall_from *fall_from_list*
Same as the **-from** option, except that the path must fall from the objects specified.If a clock object is specified, this option selects startpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-to *to_list*
Specifies a list of to pins, ports, nets, or clocks to be reported. Path endpoints are typically the output ports or data pins of registers. If you specify a clock, all endpoints are considered if they are constrained by the clock.

-rise_to *rise_to_list*
Same as the **-to** option, but applies only to paths rising at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths captured by rising edge of the clock at clock source, taking into account any logical inversions along the clock path.

-fall_to *fall_to_list*
Same as the **-to** option,but applies only to paths falling at the endpoint. If a clock object is specified, this option selects endpoints clocked by the named clock, but only the paths launched by falling edge of the clock at the clock source, taking into account any logical inversions along the clock path.

-exclude *exclude_list*
Specifies that only paths not including the named pins, ports, nets, cell instances are to be reported. Reporting will exclude all data paths from/through/to the named pins, ports, nets and cell instances. If a cell instance

where the arguments have the following meaning:

objects	Only select paths that contain one of the specified objects of type pin, port, or clocks.
[-from *pinlist*]	Only select paths that start from one of the specified list of pins.
[-rise_from *pinlist*]	Only select paths that start with a rising signal from one of the specified list of pins.
[-fall_from *pinlist*]	Only select paths that start with a falling signal from one of the specified list of pins.
[-to *pinlist*]	Only select paths that end in one the specified list of pins.
[-rise_to *pinlist*]	Only select paths that end with a rising signal in one the specified list of pins.
[-fall_to *pinlist*]	Only select paths that end with a falling signal in one the specified list of pins.
[-through *pinlist*]	Only select paths that go through at least one of the specified pins.
[-rise_through *pinlist*]	Only select paths that go through at least one of the specified pins with a rising signal.
[-fall_through *pinlist*]	Only select paths that go through at least one of the specified pins with a falling signal.

29

is specified, all pins of the cell are excluded. **-exclude** has higher precedence than **-from/-through/-to**. **-exclude** does not work with -true ption. **-exclude** is exclusive with **-rise_exclude** or **-fall_exclude**. **-exclude** does not apply to borrowing path from **-trace_latch_borrow** option or clock path from **-path full_clock/full_clock_expanded** options.

-rise_exclude *rise_exclude_list*
Same as the **-exclude** option, but applies only to paths rising at the named pins, ports, nets, cell instances.

-fall_exclude *fall_exclude_list*
Same as the **-exclude** option, but applies only to paths falling at the named pins, ports, nets, cell instances.

-through *through_list*
Specifies a list of through pins, ports, or nets to be reported. Only paths through the named pins are considered.
You can specify many *through_list* groups by using multiple **-through** options. The objects specified within one **-through** option are assumed to be in OR mode. The group of objects specified with multiple **-through** options is assumed to be in AND mode.
If you specify **-through** only once, PrimeTime reports only the paths that travel through one or more of the objects in the list.
If you specify multiple **-through** options, PrimeTime reports only the paths that travel through one or more of the objects in each list. PrimeTime uses the exact order in which the **-through** options are listed; so to obtain correct results, you must ensure that this order is the same as that followed by the actual paths in the circuit.

-rise_through *rise_through_list*
Specifies the same as the **-through** option, except that the path must rise through the objects specified.

-fall_through *fall_through_list*
Specifies the same as the **-through** option, except that the path must fall through the objects specified.

-delay_type *delay_type*
Specifies the type of path delay. Valid values are **max** (the default), **min**, **min_max**, **max_rise**, **max_fall**, **min_rise**, or **min_fall**. The "rise" or "fall" in the *delay_type* refers to a rising or falling transition at the path endpoint.

-nworst *paths_per_endpoint*
Gets *n* worst paths to endpoint, where *paths_per_endpoint* is = 1. The default is 1, meaning that only the worst path to an endpoint is considered.
Specifying larger values of *paths_per_endpoint* increases run time.

-max_paths *max_path_count*
Specifies the maximum number of paths to get per path group, where *max_path_count* = 1. The default is 1.

-path_type *format*
Specifies the format of the path report and how the timing path is displayed. The allowed value is **full_clock_expanded**, which displays full clock paths between a primary clock and a related generated clock in addition to the full_clock timing path.

-group *group_name*
Restricts the collection to paths in this *group_name*. Paths are grouped by using the **group_path** or **create_clock** command.

-true
Specifies that the longest (least-slack) true paths in the design are to be reported. This option can require long runtimes for certain designs that have many false paths. The **true_delay_prove_true_backtrack_limit** and **true_delay_prove_false_backtrack_limit** variables are used to limit the amount of backtracking during the operation of the **report_timing** command with the **-true** option. The **set_case_analysis** command is used to specify a partial input vector to be considered for **-true** analysis. **-true** cannot be combined with **-max_paths**(1), **-nworst**(1), **-delay_type** (path type other than **max**), **-unique**, **-rise_through**, **fall_through** and **-rise_from** and fall_from options: **-true** is mutually exclusive with them.

-unique_pins
Specifies that only paths through a unique set of pins are to be reported. This option can require longer runtimes when used in combination with the **-nworst** option with a large number of paths targeted for reporting.

[-delay_type max | min | min_max | max_rise | max_fall | min_rise | min_fall]	Only select paths based of the specified delay type. The following types of delay are supported: *max*—Paths with a maximum delay. *min*—Paths with a minimum delay. *min_max*—Paths with a minimum or maximum delay. *max_rise*—Paths with a maximum delay for a rising signal. *max_fall*—Paths with a maximum delay for a falling signal. *min_rise*—Paths with a minimum delay for a rising signal. *min_fall*—Paths with a minimum delay for a falling signal.
[-nworst *integer*]	Number of worst paths to retain per path group. The default value is *1*.
[-max_paths *integer*]	Maximum number of paths to retain per path group. The default value is *1*.
[-path_type full | full_clock | full_clock_expanded]	Selects paths based on their timing. The following are valid values: *full*—Retain all pins along the *-nworst* paths of each path group from launch pin to capture pin. This is the default report. *full_clock*—Retain the *-nworst* paths for each path group and include the timing of the clock from the clock root to the clock of the state element that launches the signal and the clock to the state element that captures the signal. *full_clock_expanded*—Same as *full_clock*, but also include the clock path from the original clock to the generated clock.
[-slack_greater_than *double*]	Only select paths whose slack is greater than the specified value. The default value is *-10000000000000000000*.
[-slack_less_than *double*]	Only select paths whose slack is smaller than the specified value. The default value is *10000000000000000000*.
[-groups {*string* [*string*]...}]	Only select paths belonging to one the specified groups.
[-scenario *string*]	Only select paths from the specified scenario.
[-no_hierarchical_pins]	Do not report hierarchical pins.

30

-start_end_pair
Indicates that paths are reported for each pair of startpoint and endpoint based on connectivity. This option can lead to long runtime with large memory usage and can lead to generating a huge number of paths depending on the design. By default this option will only search for paths which are violating. This default value can be changed by having an explicit **-slack_lesser_than** option. The options that do not work with this option are **-nworst**, **-max_paths**, **-unique_pins**, **-true**, **-justify**, **-slack_greater_than**, **-ignore_register_feedback**. Unlike with other options of get_timing_paths, this option causes the paths returned to no longer be sorted based on slack, instead, paths are arranged based on the endpoint with those sharing the same endpoint appearing next to one another. The maximum number of paths returned is limited to 2000000. In order to avoid the potential of returning duplicate paths, this option works as though the variable timing_report_always_use_valid_start_end_points was set to true.

-true_threshold *path_delay*
Used with the **-true** option. Specifies a threshold path delay value, in library time units, to be used by the **-true** option to speed up searching. If this option is specified, the **get_timing_paths** command with the **-true** option returns the first path it finds greater than or equal to *path_delay*, rather than continuing to search for a longer one.

-slack_greater_than *greater_slack_limit*
Specifies that only those paths with a slack greater than *greater_slack_limit* are to be reported. This option can be combined with **-slack_lesser_than** to report only those paths inside or outside a given slack range.

-slack_lesser_than *lesser_slack_limit*
Specifies that only those paths with a slack less than *lesser_slack_limit* are to be reported. This option can be combined with **-slack_greater_than** to report only those paths inside or outside a given slack range.

-ignore_register_feedback *feedback_slack_cutoff*
Specifies that timing paths are to be ignored if they start and end at the same register that holds a value. This option applies to min delay as well as max delay reports. Paths are ignored only if the slack is less than the specified *feedback_slack_cutoff* value. This option is applied as a filter to the paths after they are generated. Therefore, the number of paths generated may be less than the number specified with the -nworst and -max_paths options.

-aocvm
Specifies that the returned timing paths are to be adjusted using AOCVM information. The order in which the paths are returned matches the order in which the paths would have been returned had this option not been specified. This option automatically sets **-path_type full_clock_expanded**.

-include_hierarchical_pins
Specifies that the returned timing paths contain points for each hierarchical pin crossed, as well as all leaf pins in the path.

-justify
Specifies to find and report an input vector that sensitizes the reported paths or to report the path as false if no input vector is found. Use the **set_case_analysis** command to specify a partial input vector to be considered for **-justify** analysis.

-trace_latch_borrow
This option controls the type of report generated for a path that starts at a transparent latch. If the path startpoint borrows from the previous stage, using this option causes the report to show the entire set of borrowing paths that lead up to the borrowing latch, starting with a nonborrowing path or a noninverting sequential loop.

-recalculate
Indicates that path recalculation should be applied during the search. The worst recalculated paths meeting the path requirements are returned. This option can result in long runtimes due to the path searching required. This option does not work with **-justify**, **-true**, **-slack_greater_than** and other multi scenario options, including **-pre_commands**, **-post_commands**, **-dont_merge_duplicates** and **-attributes**.

-pre_commands *pre_command_string*
This option is available only if the user invokes PrimeTime with the -multi_scenario option. This option allows users to specify a string of commands to be executed in the slave context before the execution of the merged_reporting command. Commands must be grouped using the ";" character. The maximum size of a command is 1000 chars.

where the arguments have the following meaning:

wns | tns	Specifies whether to return worst negative slack or total negative slack.
[-setup | -hold]	Specifies whether to return the setup or hold timing result. The default is setup timing.
[-scenario *scenario*]	Specifies for which timing scenario to return the timing QoR metric.

31

-post_commands *post_command_string*
This option is available only if the user invokes PrimeTime with the -multi_scenario option. This option allows users to specify a string of commands to be executed in the slave context after the execution of the merged_reporting commands. Commands are grouped using the ";" character. The maximum size of a command is 1000 chars.

-dont_merge_duplicates
This option is available only if the user invokes PrimeTime with the -multi_scenario option. It turns OFF a main capability in merged reporting that is ON by default. The option affects the manner in which paths from multiple scenarios are merged. By default, when the same path is reported from more than one scenario, PrimeTime reports only the single most critical instance of that path in the merged report and shows its associated scenario. By using this option, PrimeTime will not merge duplicate instances of the same path into a single instance, but instead shows all critical instances of the path from all scenarios. Since the number of paths reported is limited by the -nworst, -max_paths and other options of this command, the resulting merged report, when this option is used, may not be evenly spread out across the design, but instead may be focussed on the portion of the design that is critical in each scenario.

-attributes *attribute_list*
This option is available only if the user invokes PrimeTime with the -multi_scenario option.A list of attributes to be retrieved from a slave collection. If this option is not specified then only the full_name,scenario_name and object_class attributes are retrieved. This option should be used in conjunction with set_distributed_parameter -collection_levels commands to control the amount of data retrieved from the

32

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790407	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056367
index_collection Creates a single element collection. I.e. Given a collection and an index into it, if the index is in range, extracts the object at that index and creates a new collection containing only that object. The base collection remains unchanged.	**index_collection** Retrieves an object at a specified position from a collection.
SYNTAX collection **index_collection** *collection1 index* collection *collection1* int *index*	**Syntax** index_collection *collection index*
ARGUMENTS collection1 Specifies the collection to be searched. index Specifies the index into the collection. Allowed values are integers from 0 to **sizeof_collection** - 1.	where the arguments have the following meaning: *collection* — Collection from which to retrieve the object. *index* — An integer indicating the position in the collection. The first object is at index 0.

33

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790408	Pl's Ex. 602 (AP 13.11.rel.4 Executable) at ATopTech 0057938
insert_buffer Inserts a buffer at one or more pins.	Command: insert_buffer --interactive internal development utility
SYNTAX string **insert_buffer** [-libraries *lib_spec*] [-inverter_pair] [-new_net_names *new_net_names*] [-new_cell_names *new_cell_names*] *pin_or_port_list lib_cell* list *new_net_names* list *new_cell_names* list *pin_or_port_list* string *lib_cell*	option: -net collection — the net to be buffered (require) -buffer_cell collection — specify buffer library cell -candidate_location point — buffer/inverters candidate location (require) -skip_legalize — skip incremental placement legalization -no_worse_timing — do not commit if timing does not improve -inverter_pair — use inverter pair in stead of buffer -connected_fanout collection — fanouts connected with added buffer. -module collection — buffer/inverters module -new_net_name string — specify the name of new net -new_buf_name string — specify the name of new buffer
ARGUMENTS -libraries *lib_spec* If this option is specified, then PrimeTime resolves *lib_cellP from the libraries contained in the lib_spec* only. Libraries are searched in the order in which they appear in *lib_spec*. *lib_spec* can be a list of library names, or collections of libraries loaded into PrimeTime; the latter can be obtained using the **get_libs** command. You cannot specify this option if a full library cell name has been specified. -inverter_pair Indicates that a pair of inverting library cells is to be inserted instead of a single non-inverting library cell. -new_net_names *new_net_names* Specifies the net name to be given to the new net that PrimeTime inserts. This option can only be used if only one buffer or an inverter pair is being inserted. If one buffer is being inserted, you have to pass only one net name. If an inverter pair is being inserted, you have to pass two net names. These names can be any valid net names, but must be the leaf names i.e. not the hierarchical names. The new names must not contain embedded hierarchical separators. The new names must be unique in the current context (as specified by current_instance). If you use this option, you have to also use the -**new_cell_names** option.	description: This command is for AtopTech internal use only.

34

-new_cell_names *new_cell_names*
Specifies the cell name to be given to the new cell that PrimeTime inserts. This option can only be used if only one buffer or an inverter pair is being inserted. If one buffer is being inserted, you have to pass only one cell name. If an inverter pair is being inserted, you have to pass two cell names. These names can be any valid cell names, but must be the leaf names i.e. not the hierarchical names. The new names must not contain embedded hierarchical separators. The new names must be unique in the current context (as specified by current_instance). If you use this option, you have to also use the **-new_net_names** option.

pin_or_port_list
Specifies a list of pins or ports to buffer.

35

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790418

link_design

```
Resolves references in a design.
```

Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056420-24

link_design

Builds the complete design by resolving references from instances to cells. This command performs the following functions:

- The different LEF, GDS, and Liberty libraries are combined to build an internal project library with cells that have all views needed by Aprisa (timing view, layout view, abstract view, and so on).
- The references in the imported design are replaced by references to cells from this project library.
- In a hierarchical design, this step also resolves the references from blocks in the design to cells representing hard blocks.

If no design has been set yet with the *current_design* or *current_module* Tcl command, then Aprisa sets the current design to the first found module without a parent.

To build the design, the *link_design* Tcl command uses the following information:

- Verilog netlist (loaded with the *read_verilog* Tcl command)
- Logic/timing library (loaded with the *read_liberty* Tcl command)
- Physical library (loaded with the *read_lef*, *read_milkyway_fram*, *load_library* Tcl commands)
- PR_LIB or GDS abstract libraries stored with the project, when available. These need not be loaded explicitly. They allow you to examine the abstract in the context of the complete design.

For references to Liberty models that are not yet loaded, Aprisa uses the search path as set by *set_link_path* Tcl command. Only Liberty libraries can be loaded on demand. All physical libraries must be loaded explicitly for the *link_design* Tcl command to add them to the project library. When you load several libraries containing cells with identical names, Aprisa issues a warning and uses the first cell found. It is, however, recommended to ensure that all cells have unique names.

Re-executing the *link_design* Tcl command on a design causes Aprisa to rebuild the internal project library using the current settings of the search path and library variables, and re-establishes the binding of instances in the design to cells in the project library.

When saving a design, you have the option to save a local copy of the physical cells of the internal project library with the design. The *gdslib* library contains the full layout in GDS format as loaded from GDS or OASIS. The *prlib* library contains the routing abstract in PR_LIB format, as loaded from LEF, Milkyway FRAM, or PR_LIB libraries. Bot the *prlib* and *gdslib* libraries are stored with the project. Once these project libraries exist, they are always loaded when the design is loaded. By default, the *link_design* Tcl command links these libraries after external libraries that were loaded using *read_lef*, *read_gds*, *read_oasis* or *load_library* Tcl commands with the *-link_first* argument, that is, cells of these external libraries will overrule cells saved in the project libraries. Using the *db* parameter, *use_own_lib_before_link_first,* the *prlib* and *gdslib* project libraries are linked first, that is, only cells that do not yet exist in the project libraries are picked up from external libraries, even if the *-link_first* argument was used. When the *use_own_lib_before_link_first* parameter is set, the *link_first* argument only affects the order of libraries linked after the project libraries.

You can control how Aprisa deals with name conflicts when importing GDS cells. When third-party macros are imported, typically they come with their own GDS libraries. Enable the persistent *db* parameter, *auto_uniquify_gds_cells,* to automatically generate unique names for cells with identical names but different content. Cell content is considered different if any shape on any layer differs, or if they contain cross-references of different cells. Only GDS cells who have the cell attribute *uniquify* set are eligible for uniquification.

Note that, even though they contain the same information, Aprisa makes a distinction between empty modules and missing modules. An empty module has an (empty) Verilog definition; A missing module is a module who was not found and whose Verilog definition was inferred. You control whether empty modules should be considered errors. Missing modules are always considered errors.

With regards to busses, Aprisa expects bus nets to be connected to bus pins.A *LibPinNotBus* error is issued when the Verilog netlist contains a connection of a bus to a pin that is not defined as a bus pin.

You can specify that a pin is a bus in one of the following two ways:

- Load a .lib with the correct library pin bus definitions.
- Use a Verilog stub to describe the library cell with its bus pins.

There are two cases when a bus pin is inferred from the bus net it is connected to:

- A cell is a proto-lib_cell or a proto-module.
- The *db* parameter, *default_bus_lib_pin_order*, specifies a default bit order. This approach allows Aprisa to infer that a pin is a bus pin and to infer how the bus pins are connected to the net bits. Note that this method is not recommended.

SYNTAX

```
string link_design [-verbose] [-remove_sub_designs] [-keep_sub_designs]
[design_name]
stringdesign_name
```

Syntax

```
link_design [-proto] \
   [-replace_own_pr_lib_with { string [string]...}] \
   [-replace_own_gds_lib_with { string [string]...}] \
   [-reload_lg_lib_with_diff] \
   [-no_proto_lib_cell] \
   [-strict] \
   [-max_ref_count_for_proto_module integer] \
   [-min_pin_count_for_proto_module integer] \
   [-allow_defined_empty_modules] \
   [-bind_lib_cell_only_to_empty_module]
```

ARGUMENTS

-verbose
: Indicates that the linker is to display verbose messages.

-remove_sub_designs
: Indicates that subdesigns are to be removed after linking. By default, subdesigns are removed. Use this option to free up memory and improve performance. For more information, see the section entitled "Performance Considerations."

-keep_sub_designs
: Indicates that subdesigns are to be kept after linking. By default, subdesigns are removed. Use this option to keep the sub-designs around so that current_design can be changed to other designs later.

design_name
: Specifies the name of the design to be linked; the default is the current design.

where the arguments have the following meaning:

[-proto]	Create prototype modules and library cells for all cells referenced in the netlist for which physical models are missing.
[-replace_own_pr_lib_with { *string* [*string*]...}]	In the project library, replace all cells whose names are in the specified list with cells that have the same name from one of the loaded libraries. The project library can be saved on disk when the project is saved, and contains a copy of all cells with an Aprisa layout view that are used in the design.
[-replace_own_gds_lib_with { *string* [*string*]...}]	Replace all cells in the projects's own *gdslib* whose name is in the specified list of cell names with cells that have the same name from one of the loaded libraries. The project's own *gdslib* is created when the project is saved and contains a copy of all cells with a GDS layout view that are used in the design.

37

`[-reload_lg_lib_with_diff]`	Reload Liberty files from different Linux paths based on the current *search_path* or *link_path* settings. By default, a Liberty file is not loaded if another Liberty file with that same file name and same internal library name is already loaded, even if the path name is different or the file on disk (that is, its timestamp) changed. Use this argument if you want to replace the Liberty file with a new version that is in a different path but has the same file name and internal library name. After using this argument, the old version of the Liberty file can be removed from memory using the *remove_library -all_unused* Tcl command.
`[-no_proto_lib_cell]`	Do not create a dummy library cell if no valid cell is found.
`[-strict]`	Do not link unless all logical and physical library cells are present.This is the default behavior. If the *-proto* argument is used, this argument is not applicable. If for a module no abstract is found, the abstract is generated automatically, and the warning *LnkNoAbs* is issued.
`[-check]`	Link but issue a warning when library cells are missing, physical (LEF) or logical (Liberty) models are missing, when the pins on instances do not match the ports defined on the library cells or when pins in physical models do not match the pins on logical models.
`[-max_ref_count_for_proto_module maxinst]`	Maximum number of references allowed for proto-modules. If more than *maxinst* instances of the missing cell exist, no proto-module is created. The default value is *10*.
`[-min_pin_count_for_proto_module minpin]`	Minimum number of pins required for proto-module. If a missing cell has less than *minpin* pins, no proto-module is created. The default value is *50*.
`[-allow_defined_empty_modules]`	Do not treat empty modules as errors.
`[-bind_lib_cell_only_to_empty_module]`	Only bind a module to a corresponding library cell if that module is empty.

38

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790422	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056426
list_attributes Lists currently defined attributes.	**list_attributes** Lists the attributes of an object type if the *-class* argument is specified. Otherwise, it lists attributes of all available object types. For a list of supported object types, see Aprisa Classes.
SYNTAX string **list_attributes** [-application] [-class *class_name*] string*class_name* **ARGUMENTS** -application Lists application attributes as well as user-defined attributes. -class *class_name* Limit the listing to attributes of a single class. Valid classes are design, port, cell, net, and so on.	**Syntax** list_attributes \ [-class *class*] \ [-pattern *string*] \ [-sort]
ARGUMENTS -application Lists application attributes as well as user-defined attributes. -class *class_name* Limit the listing to attributes of a single class. Valid classes are design, port, cell, net, and so on.	where the arguments have the following meaning:: [-class *class*] — Specifies the type of object for which to return the attributes. [-pattern *string*] — List the attributes or report on the attributes that match the specified name pattern. [-sort] — Sort the names of the reported attributes.

39

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790428	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056434-35
list_libraries Lists all libraries that are read into PrimeTime.	**list_libraries** Lists all libraries that are loaded into an Aprisa session. The command returns the full paths to libraries, or, if the project was saved with the *-all_libs*, *-prlib*, or *-gdslib* arguments, the paths are shown as *own_pr_lib, lg_lib*, or *gds_pr_lib*. For each library, four names are reported: • *file_name* is the UNIX file name. • *full_name* is the UNIX file name, except for a Liberty file, where the *full_name* is the *file_name* followed by the internal library name. • *full_path* is the absolute UNIX path with all symbolic links resolved. • *search_path* is the path name to the library as specified in the library search path. When you save a project, you control whether the *full_path* or the *orig_path* to files are stored with the *db* parameter *dont_expand_paths*. You can also obtain this information from the following read-only attributes on each library: • *full_name*—Full name of the library, including the name of the file and the internal name of the library. • *base_name* —Base name of the library. For Liberty files, this is the internal library name; For all other libraries, such as LEF and GDS, this is the name without suffix of the file containing the library. • *source_file_name*—Name of the Linux file from which the library was read. • *source_file_search_path_name*—Path name of the Linux file containing the library, using the path as specified in the search path. This path may contain symbolic links and may be relative. • *source_file_full_path_name*—Absolute path name of the Linux file containing the library, with symbolic links resolved. You can list the attributes, use the following Tcl command: `% list_attr -class lib`

SYNTAX
 	
`string` **`list_libraries`** `[-only_used]`
	Syntax
	
	`list_libraries [lib_names] \`
	`  [-only_used] \`
	`  [-detail] \`
	`  [-lib_cell cells]`

40

ARGUMENTS

```
-only_used
        Indicates only the list libraries in use. A library is in use if a linked
        design links to library cells from the library.
```

where the arguments have the following meaning:

[*lib_names*]	Only report on the specified libraries.
[-only_used]	Only list the libraries if they contain cells that are used in the current project.
[-detail]	List for each library all the cells that are used in the current project.
[-lib_cell *cells*]	Only report on cells from the specified list. The link path information and list of libraries are not reported. If this argument is used with the *-only_used* argument, only used cells from the list are reported.

41

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790431	Pl's Ex. 602 (AP 13.11.rel.4 Executable) at ATopTech 0057938
load_of Gets the capacitance of a library cell pin. It is a DC Emulation command provided for compatibility with Design Compiler.	Command: load_of <string:lib_pin> <lib_pin> lib pin Tcl procedure
SYNTAX float **load_of** *lib_pin* string*lib_pin*	option: --help display command help
ARGUMENTS lib_pin Specifies the name of the library cell pin, or a collection that contains the library cell pin, for which to get the capacitance.	description: return pin load

42

```
parse_proc_arguments # Parse arguments to a procedure
```

```
-args arg_list      (Argument list to be parsed)
   result_array     (Name of array to use to store parse results)
```

define_proc_attributes

Describes the help text of a specified Tcl procedure and describes the attributes of its arguments. This Tcl command allows you to use Aprisa's *parse_proc_arguments* Tcl procedure in your procedure to parse its arguments. By using both the *define_proc_attributes* and *parse_proc_arguments* Tcl commands, you integrate your procedure in the Aprisa environment. The meta arguments, such as ***--h***, are enabled and the *info* and *help* commands also work for your procedure.

Syntax

```
define_proc_attributes \
   procedureName \
   -info string \
   -define_args * \
```

where the arguments have the following meaning:

procedureName	Name of the procedure.
-info string	One-line help string for the procedure.
-define_args *	Arguments and options of the procedure. This is a collection of argument definitions. Each argument definition has the following format:

```
{arg_name option_help value_help data_type attributes}
```

where:

arg_name	Name of the argument. If the name starts with a '-', it indicates a named argument. Otherwise, it is a positional argument.
option_help	Help string describing the argument.
value_help	Help string describing the acceptable values for the argument
data_type	Type of value expected for this argument, such as *float*, *string*, *boolean*, *one_of_string*
attributes	Additional attributes, such as *required*, *optional*, and for types *one_of_string*, the list of *values*.

Pl's Ex. 26 (PrimeTime 2010.06) at ATopTech SYNPS-00005754	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056653-54
read_aocvm Reads advanced on-chip variation (OCV) derate factor tables.	**read_aocvm** Reads an advanced on-chip variation (OCV) derating model from a text file. This is the recommend method to build such a model. The *set_aocvm_component* Tcl command will be phased out eventually.
SYNTAX int **read_aocvm** *aocvm_file*	**Syntax** read_aocvm *file* [-use_db_distance_unit]
ARGUMENTS aocvm_file Specifies the name of the advanced OCV file.	where the arguments have the following meaning: *file* — Name of the advanced OCV model file to read. [-use_db_distance_unit] — Assume distances are in user-defined distance units instead of database units. The syntax of the file is as follows: version *version_number* object_type design | lib_cell | cell rf_type rise | fall | rise fall delay_type cell | net | cell net derate_type early | late object_spec *string* depth *set_of_M_floats* distance *set_of_N_floats* table *N_rows_M_colums* where the lines have the following meaning: object_type design | lib_cell | cell — Derating model holds for standard cell, a specific block, or for a complete design. rf_type rise | fall | rise fall — Derating factor holds for a rising event, a falling event, or both. By default, the setting holds for both. delay_type cell | net | cell net — Derating factor applies to cells, nets, or both.

`derate_type early | late`	Setting holds for an early path (signal path for hold analysis, or clock path for setup analysis) or for a late path (signal path for setup analysis, clock path for hold analysis).
`object_spec` *`string`*	This line is ignored for now. Will be implemented in future release.
`depth` *`set_of_M_floats`*	Different values of logic depth for which a column of derating factors is provided. Note that *M* can be zero, indicating that this is a one-dimensional model that has derating factors that are only a function of the distance.
`distance` *`set_of_N_floats`*	Different values of distance for which a row of derating factors is provided. Note that *N* can be zero, indicating that this is a one-dimensional model that has derating factors that are only a function of the depth.
`table` *`N_rows_M_columns`*	*N* rows, with in each row *M* values. Each row corresponds to a distance. Each column corresponds to a depth.

45

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790454	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056692
read_milkyway Reads in one linked design from milkyway database.	**read_milkyway_fram** Reads physical library data from a Milkyway FRAM library. The FRAM library contains the cell frame views, that is, for each cell the location of its pins and the blockages for the router on the various layers. In addition the route abstracts, also the *no_pg* and *no_signal* route guides are read. The reader fully supports the Milkyway 2008 standard.
SYNTAX int **read_milkyway** [-version version] [-netlist_only] [-library design_library] [-scenario scenario_name] *CEL_name* string *CEL_name* string *scenario_name* string *design_library*	**Syntax** read_milkyway_fram *mw_library_path* where *mw_library_path* is the path name of the Milkyway database to read.
ARGUMENTS -version *version* Specifies the version of the design to be read. For example, there are design files under the CEL view in the milkyway design library design_lib: 'design_lib/CEL/design1_pre_route1:1', 'design_lib/CEL/design1_post_route:2' etc. The 1 or 2 after the ':' is the version number of the design. The default is to read the most current version. -netlist_only Indicates that only the netlist is to be read; constraints are not read. The default is to read both netlist and constraints. -library *design_library* Specifies the absolute or relative path to the MW design library. This option can be left out if the variable **mw_design_library** specifies the path to the MW design library. -scenario *scenario_name* MW database is capable of storing multiple constraints that can correspond to various scenarios of running the design. This option specifies the name of the scenario for reading in constraints from MW database. The default is to not use a scenario. CEL_name Specifies the design filename to be read. For example, there are design files under the CEL view in the milkyway design library design_lib: 'design_lib/CEL/design1_pre_route1:1', 'design_lib/CEL/design1_post_route:2' etc. The design1_pre_route or design1_post_route are the CEL_name argument. Do not include version number in this argument.	

46

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790454

read_milkyway

Reads in one linked design from milkyway database.

SYNTAX

```
int read_milkyway [-version version] [-netlist_only] [-library design_library] [-
scenario scenario_name] CEL_name
string CEL_name
string scenario_name
string design_library
```

ARGUMENTS

-version *version*
: Specifies the version of the design to be read. For example, there are design files under the CEL view in the milkyway design library design_lib: 'design_lib/CEL/design1_pre_route1:1', 'design_lib/CEL/design1_post_route:2' etc. The 1 or 2 after the ':' is the version number of the design. The default is to read the most current version.

-netlist_only
: Indicates that only the netlist is to be read; constraints are not read. The default is to read both netlist and constraints.

-library *design_library*
: Specifies the absolute or relative path to the MW design library. This option can be left out if the variable **mw_design_library** specifies the path to the MW design library.

-scenario *scenario_name*
: MW database is capable of storing multiple constraints that can correspond to various scenarios of running the design. This option specifies the name of the scenario for reading in constraints from MW database. The default is to not use a scenario.

CEL_name
: Specifies the design filename to be read. For example, there are design files under the CEL view in the milkyway design library design_lib: 'design_lib/CEL/design1_pre_route1:1', 'design_lib/CEL/design1_post_route:2' etc. The design1_pre_route or design1_post_route are the CEL_name argument. Do not include version number in this argument.

Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056694

read_milkyway_tech

Imports the technology information from a Milkyway database.

Syntax

```
read_milkyway_tech filename \
   [-rlc_model rlc_model] \
   [-rlc_corner MIN | NOM | MAX] \
   [-routing_dir hv | vh]
```

where the arguments have the following meaning:

filename	Name of the Milkyway file to read.
[-rlc_model *rlc_model*]	Name of the RLC model that is created as part of the technology import. The default value is *MW*.
[-rlc_corner MIN | NOM | MAX]	Read the RLC data of the specified corner. The default value is *MAX*.
[-routing_dir hv | vh]	Routing direction for all layers. For the *hv* routing direction, which is the default value, metal1 is routed horizontally; metal2 is routed vertically; metal3 is routed horizontally, and so on. For the *vh* routing direction, metal1 is routed vertically; metal2 is routed horizontally; metal3 is routed vertically, and so on.

PI's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790456-58

read_parasitics

Reads net parasitics information from an SPEF, DSPF, RSPF, or binary parasitics file and uses it to annotate the currently linked design.

SYNTAX

```
Boolean read_parasitics
[-format file_fmt]
[-complete_with completion_type]
[-lumped_cap_only]
[-pin_cap_included] [-increment]
[-path prefix]
[-keep_capacitive_coupling]
[-coupling_reduction_factor factor]
[-triplet_type ttype]
[-quiet] [-syntax_only]
    [-eco]
    [-original_file_name file_name]
[-ilm_context]
[-keep_variations]
    [-create_default_variations]
file_names

string file_fmt
string completion_type
string path_name
string file_names
string ofname
float  factor
```

ARGUMENTS

-format *file_fmt*
Specifies the format of the parasitics file. Allowed values are SPEF, DSPF, RSPF and SBPF (Synopsys Binary Parasitics Format). If **-format** is not specified, the application can determine whether the file is SPEF, DSPF, RSPF, or a compressed version of those three ascii formats. However, to read a file in SBPF, you must specify **-format SBPF**.

-complete_with *completion_type*
This option does not apply to the RSPF format. Indicates that a net with partially annotated parasitics is to be completed by inserting capacitances and resistances according to *completion_type*. Allowed values are *zero*, which completes the net by inserting zero capacitances and resistances; and *wlm*, which completes the net by inserting capacitances and resistances derived from wire load models. This option is equivalent to reading the parasitics file and then using the command **complete_net_parasitics -complete_with**.
Note: complete_net_parasitics and **read_parasitics -complete_with** complete a net only if all missing segments are between two pins and the nets are partially annotated (nets are not affected if they are fully annotated or have no annotation at all). Also, the net must be hierarchical, so that if the parasitics for the block-level parts of a net are missing, those

PI's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056700-01

read_parasitics

Loads parasitic information extracted from a third-party tool onto the current design. By default, this command reads parasitic data in SPEF format.

The command accepts names with the special character ' (single quote) and insert an escape character. For example, nets that are set to *1'b0* and *1'b1* may result in SPEF net names such as the following:

```
cx_wrap/ucx/udpj_ctl/1\'b1
```

Syntax

```
read_parasitics filenames \
   [-format DSPF | SPEF] \
   [-lumped_cap_only] \
   [-pin_cap_included] \
   [-increment] \
   [-quiet] \
   [-syntax_only] \
   [-path path] \
   [-strip_path prefix] \
   [-merge_same_net_coupling] \
   [-condition { condition [condition]... }]
```

where the arguments have the following meaning:

filename	Name of files with parasitic information to load.
[-format DSPF | SPEF]	Format of the parasitic data. The default file format is *SPEF*.
[-lumped_cap_only]	Only annotate the total capacitance of the nets.
[-pin_cap_included]	RC networks already include the pin capacitances.
[-increment]	Add these parasitics to previously annotated parasitics instead of replacing them.
[-quiet]	Do not report the annotated parasitics in the log file.
[-syntax_only]	Do not load the parasitics but check if the SPEF syntax is valid. Note that this is only a syntax check. It

48

parasitics could exist in the top-level net. . If any of these conditions are not met, you must correct the SPEF or DSPF file manually.

-lumped_cap_only
This option does not apply to the SBPF format. Indicates that only the total capacitance of nets is to be annotated as a lumped capacitance on the annotated nets. The RC networks specified in the parasitics file are discarded. The annotated lumped capacitance is the capacitance specified when the net is declared in the parasitics file.

-keep_capacitive_coupling
Indicates that the cross capacitors are to be kept in the RC networks data structure. This facilitates the capacitive crosstalk analysis, but does not turn it on. This option disables the **-coupling_reduction_factor** option; the command will fail if both options are specified. All coupling capacitors are split to ground with a factor of 1.0 if crosstalk analysis is not activated. This option applies to both the SPEF and the SBPF format. This option requires a PrimeTime SI license.

-pin_cap_included
Indicates that the RC networks are to include the pin capacitances. By default, the RC network does not include pin capacitances. This option does not apply to the RSPF format. The RC pi model in RSPF format has to always include effect of pin capacitances.

-increment
Indicates that previously annotated parasitics on the nets listed in the parasitics file are not to be overwritten. Additionally, any incomplete annotations in the parasitics file are not to be rejected. By default, the RC annotation specified in the parasitics file overwrites the previous parasitics annotations of the nets listed in the parasitics file. Use this option for annotating hierarchical parasitics files.

-path *prefix*
Specifies a relative path from the current design to the hierarchical design name for which the parasitics file has been created. By default, absolute pathnames are used. Use this option if the parasitics file refers to an object (for example, *net*) in a hierarchy (for example, *hier*). Do not use this option if the parasitics file refers to an absolute path (for example, *hier/net*).

-coupling_reduction_factor *factor*
This option applies only to the SPEF format and the SBPF format. A positive floating point number that specifies the factor to apply when reducing coupling capacitances to grounded capacitances. The default value is 1.0. This option is disabled if the **-keep_capacitive_coupling** option is specified. The command will fail if both options are specified.

-triplet_type *ttype*
This option applies only to the SPEF and PARA formats. Several values in SPEF and PARA, such as capacitor and resistor values, can be specified as triplets - min:typ:max. By default, PrimeTime takes the max value. Using this option, the user can select the min or typ value. Allowed values are *max* (the default), *typ*, and *min*.

-quiet
Indicates that the **report_annotated_parasitics** report is not to be generated when the parasitics file has been read. By default, after reading the parasitics file, the **report_annotated_parasitics -check** command is executed. This command reports the number of annotated nets, verifies the completeness of annotated RC networks on nets, and checks that no RC elements dangle. It is recommended that you use the **-quiet** option when reading multiple parasitics files in incremental mode.

-syntax_only
Indicates that **read_parasitics** is to parse the file for syntax errors without performing any parasitic annotation. Use this option to troubleshoot your parasitics file and avoid generating error messages during the actual annotation. No design is required to use **-syntax_only**.

	does not check whether the parasitic file matches the netlist.
[-path *path*]	Relative path from the current design to the hierarchical design name for which the parasitic file has been created.
[-strip_path *prefix*]	Prefix of all SPEF or DSPF objects that needs to be stripped.
[-merge_same_net_coupling]	Merge coupled capacitances between same nets.
[-condition { *condition* [*condition*]... }]	List of SPEF parasitic conditions to load.

49

-ilm_context
Indicates that the annotation is being performed in the presence of Inteface Logic Models (ILMs). An original design parasitics can be used to annotate a design with ILMs using this option. This option does not issue error messages for missing nets, cells and pins.

-eco
Indicates that the files being currently annotated are ECO parasitics from Star-RCXT. PTSI can read ECO parasitics that are written out by Star-RCXT only. The ECO parasitics can be annotated only when there are some existing parasitics that are already annotated. ECO parasitic files contain re-extracted parasitics for just the ECO nets and their immediate coupling neighbours only and do not contain all the nets of the design. Incremental analysis can be performed after reading ECO parasitics.

-original_file_name *orig_file_name*
This option can only be used when *-eco* option is being used. If the original annotation is performed via multiple parasitic files into PTSI, then the ECO parasitic file corresponds to one of the original files (because it corresponds to one extracted database in Star-RCXT). PTSI will try to determine the corresponding original file but it is not always possible. You can use this option to specify which original parasitic file does the ECO file correspond to.

file_names
When the format is one of SPEF, DSPF, RSPF and SBPF, it specifies a list of files from which parasitics information is to be read.

-keep_variations
Indicates that the statistical parasitic information are to be kept in the RC networks data structure. This facilitates the variation aware timing analysis, but does not turn it on. This option applies only to SBPF format for now. Also, currently, this option does not work with either **-eco** option or **-increment** option. This option requires a PrimeTime VA license.

-create_default_variations
Specifies that default parasitic variations should be created for all the variation parameters. The default variations created are all assumed to be of normal distribution. The mean and sigma values are already present in the parasitic file.

50

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790477 -79

read_sdf

Reads leaf cell and net timing information from a file in Standard Delay Format (SDF) and uses that information to annotate the current design.

SYNTAX

```
string read_sdf [-load_delay net | cell]
[-analysis_type single | bc_wc | on_chip_variation]
[-min_file min_fname]
[-max_file max_fname]
[-path path_name]
[-type sdf_min | sdf_typ | sdf_max]
[-min_type sdf_min | sdf_typ | sdf_max]
[-max_type sdf_min | sdf_typ | sdf_max]
[-cond_use min | max | min_max] [-syntax_only]
[-strip_path strip_path_name]
[-quiet] [-worst ]
file_name

string path_name
string sdf_file_name
string min_sdf_file_name
string max_sdf_file_name
string strip_path_name
```

ARGUMENTS

-load_delay net | cell
: Indicates whether load delays are included in net delays or in cell delays in the timing file being read. The default is *cell*. The load delay is the portion of cell delay arising from the capacitive load of the net driven by the cell.

-analysis_type single | bc_wc | on_chip_variation
: Use this option only if you have not already set an analysis type with **set_operating_conditions -analysys_type**. If you are in min_max mode, the default is *bc_wc*. *single* indicates that only one operating condition is to be used. Specifying either *bc_wc* or *on_chip_variation* switches to min_max mode and causes both minimum and maximum delays to be read from the SDF file. Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max). By default, the **-analysis_type bc_wc** | **on_chip_variation** option reads the *sdf_min* and *sdf_max* delays, respectively. To change this, use the **-min_type** and **-max_type** options.

-min_file *min_sdf_file_name*
: Use this option only if the minimum and maximum delays are in two separate SDF files. Specifies the file from which minimum delay timing information is to be read. The timing file must be in SDF format version v1.0, v2.0, v2.1 or v3.0.

-max_file *max_sdf_file_name*
: Use this option only if the minimum and maximum delays are in two separate

Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056706-07

read_sdf

Reads timing data from a Standard Delay Format (SDF) file and back-annotates the design. Both regular text files and compressed gzip files are supported.

NOTE: The listed unsupported options will be supported in a future release.

Syntax

```
read_sdf file \
   [-load_delay load_delay] \
   [-analysis_type analysis_type] \
   [-min_file min_file] \
   [-max_file max_file] \
   [-path path] \
   [-min_type sdf_min | sdf_type | sdf_max] \
   [-max_type sdf_min | sdf_type | sdf_max] \
   [-type sdf_min | sdf_type | sdf_max] \
   [-cond_use cond_use] \
   [-strip_path strip_path] \
   [-syntax_only] \
   [-quiet]
```

where the arguments have the following meaning:

file	Name of the SDF file to read.
[-load_delay *load_delay*]	Not supported yet.
[-min_file *min_file*]	Not supported yet.
[-max_file *max_file*]	Not supported yet.
[-path *path*]	Not supported yet.
[-min_type sdf_min | sdf_typ | sdf_max]	Not supported yet.
[-max_type sdf_min | sdf_typ | sdf_max]	Not supported yet.
[-type sdf_min | sdf_type | sdf_max]	Not supported yet.
[-cond_use *cond_use*]	Not supported yet.

51

SDF files. Specifies the file from which maximum delay timing information is to be read. The timing file must be in SDF format version v1.0, v2.0, v2.1 or v3.0.

-path *path_name*
Specifies the path from the current design to the subdesign for which the timing file has been created.

-type sdf_min | sdf_typ | sdf_max
Indicates which of the SDF triplet delay values are to be read from the SDF file. Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max). By default, **read_sdf** reads the maximum delays *sdf_max*.
Note: If you use **-type** while in min/max mode (for example, if you use **-operating_conditions bc_bw | on_chip_variation**), a single value is annotated onto both min and max values of an arc.

-min_type sdf_min | sdf_typ | sdf_max
Specifies which of the SDF triplet delay values are to be read from the SDF file for minimum delay. Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max). By default, **read_sdf** reads the minimum delays *sdf_min*. Use this option only with option **-analysis_type bc_wc | on_chip_variation**.

-max_type sdf_min | sdf_typ | sdf_max
Specifies which of the SDF triplet delay values are to be read from the SDF file for maximum delay. Delays in SDF are represented in the form of triplets (sdf_min:sdf_typ:sdf_max). By default, **read_sdf** reads the maximum delays *sdf_max*. Use this option only with option **-analysis_type bc_wc | on_chip_variation**.

-cond_use min | max | min_max
Use this option only if the SDF file includes some conditional delays using the SDF construct COND, and if the Synopsys library in use does not specify conditional delays. *min* indicates that the minimum of all conditional delays is to be used to annotate the corresponding timing arc. *max* indicates to use the maximum; *min_max* indicates min_max operating conditions; the minimum of all conditional delays is to be used for the minimum operating condition, and the maximum of all conditional delays is to be used for the maximum operating condition, You cannot use *min_max* with a single operating condition; you must be in min_max mode.

-syntax_only
Indicates that no timing annotation is to be performed; syntax only is to be processed. Use this option to verify that your SDF syntax is correct and will not issue any error messages.

-strip_path *strip_path_name*
Specifies a prefix path that is to be stripped from all SDF objects. Such a prefix path is usually a result of generating an SDF file for a subdesign, and using this subdesign as the current design.

-quiet
Use this option to skip execution of **report_annotated_delay** and **report_annotated_check** after reading SDF.

-worst
Indicates that **read_sdf** is to annotate the current design only with delays worse than the current annotated delays; applies to annotated net and cell delays and annotated timing checks. The worst delay is defined as the most pessimistic delay. This means primetime annotates the min of minima, and max of maxima values.

sdf_file_name
Specifies the file from which timing information is to be read. The timing file must be in SDF format version v1.0, v2.0, v2.1 or v3.0.

[-strip_path *strip_path*] Not supported yet.

[-syntax_only] Not supported yet.

[-quiet] Not supported yet.

52

PI's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790481-82	PI's Ex. 647 (AP 10.06.rel.1 Executable)
read_vcd The read_vcd command specifies the switching activity information generated by simulation for use in power calculation. Internally, non-VCD format switching activity is converted to VCD.	Command: read_vcd <string:name>
SYNTAX int read_vcd [-path *prefix*] [-strip_path *prefix*] [-zero_delay] [-pipe_exec *command*] [-cells *cell_list*] [-time *time_list*] file_name string prefix string file_name string command list cell_list list time_list	option: --license list required licenses --help display command help license: AP Command: read_verilog <string:filenames> reads Verilog files for linking option: -no_check do not perform additional syntax/semantic checking --get_option arg<1> get option value --set_option ... set option value --get_default arg<1> get default value --set_default ... set default value --list_options list current option values --load_options ... load current option values --license list required licenses --help display command help
ARGUMENTS -path prefix Specifies a relative path from the current design to the hierarchical low-level design for which the VCD file has been created. By default, absolute path names are used. Use this option if the VCD file refers to an object in a hierarchy. Do not use this option if the VCD file refers to an absolute path. -strip_path strip Specifies a path prefix that is to be stripped from all the object names read from the VCD file. This option is applied to strip the testbench/instance path from the VCD file. -zero_delay Specifies the VCD file comes from a zero delay simulation. -pipe_exec command Specifies a shell command which is used to generate the VCD file *file_names*. This option will invoke the command and directly pipe the output VCD file to PrimeTime-PX. In another word, the simulation and power analysis are in parallel run. No VCD disk file is generated at all.	description: This command parses Verilog files and build a net-list ready for 'link_design'. The net-list is not a design database and it will be removed after the link_design is done.

53

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790481-82	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056719
-time time_window_value Specifies a time value in nanoseconds (ns). If the time window is specified, power will be calculated only for the events happening within this time window. The user can specify as many time periods as necessary; however, the user must specify the beginning time of simulation windows in increasing order.If the user does not know the end of simulation time, user can use a negative number in **-time** option to indicate the end of simulation. This option gives the user the flexibility to focus on the time window of interest. -cells cell_list Specifies the cell names or collections of cells for which power need to be calculated. Without this option, PrimeTime PX calculates power for the whole design. file_name Specifies the switching acitivity file name to be read. If file_name ends with gzipped file, a compressed file, a VCD+ file, and a FSDB file, respectively, otherwise it is assumed to be in the VCD format.	

54

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790484	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056720
read_verilog Reads in one or more Verilog files.	**read_verilog** Reads the hierarchy and connectivity information of a design from a set of Verilog files and builds a netlist ready for linking, that is, binding instances to modules and library cells using the *link_design* Tcl command. The actual design database is only created during the *link_design* step. If the Verilog netlist contains connections to pins that do not exist on the library cell, then these pins are automatically generated to the library cell and a warning is issued. This command automatically recognizes and reads Verilog files in GZIP format.
SYNTAX string **read_verilog** [-hdl_compiler] *file_names* list *file_names*	**Syntax** read_verilog *files* \ [-no_check] \
ARGUMENTS -hdl_compiler Indicates that the Verilog files are to be read using the PrimeTime external reader (ptxr) that uses HDL Compiler. Reading files in this way requires an HDL Compiler license while the read is in progress. HDL Compiler supports the complete Verilog language, but uses more CPU and memory than does the native PrimeTime Verilog reader. file_names Specifies names of one or more files to be read.	where the arguments have the following meaning: *files* — Names of the Verilog files to read. Aprisa automatically recognizes and reads Verilog files in GZIP format. [-no_check] — Do not perform additional syntax and semantic checking. This expedites the reading, but may cause fatal errors down the road. Use this argument only if you read in Verilog files that have been previously checked.

55

Pl's Ex. 18 (PrimeTime 2006.12 Executable) at SNPS_EXE_011	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056725-26
redirect # Redirect output of a command to a file	**redirect** Redirects the output of any Tcl command to a user-specified file or to a Tcl variable. The redirect command allows you to send the output to more than one destination, such as to the screen and a file using the *-tee* argument. If you do not want to tee the output, you can redirect the output via the standard Tcl method as follows: *command > file*
[-append] (Append output to the file) [-tee] (Tee output to the current output stream) [-file] (Output to a file (default)) [-variable] (Output to a variable)	**Syntax** redirect *target command* \ [-append] \ [-tee] \ [-variable] \ [-compress] Or, to pass arguments to *command*: redirect *target* {*command command_options*}\ [-append] \ [-tee] [-variable] \ [-compress]
target (Name of file/variable target for redirect) command_string (Command to redirect. Should be in braces {}.)	where the arguments have the following meaning: *target* — Name of the file to which to write the output, or, if the *-variable* argument is used, name of the Tcl variable in which to store the output. *command* — Name of the Tcl command whose output you want to redirect. *command_options* — Command options to *command*. [-append] — Append output to the file instead of overwriting the file. [-tee] — Copy the output to the screen. [-variable] — Redirect output to a Tcl variable instead of to a file. [-compress] — Compress the output file.

56

remove_annotated_delay

Removes annotated delays from the design, either on specific cells or nets, between specific pins, or all annotated delays in the design.

SYNTAX

```
string remove_annotated_delay
[-all]
[-from from_list]
[-to to_list]
[object_spec]

list  from_list
list  to_list
list  object_spec
```

ARGUMENTS

-all
: Indicates that all annotated delays in the design are to be removed. This option is exclusive of the **-from**, **-to**, and *object_spec* options.

-from *from_list*
: Specifies a list of pins or ports that are the startpoints of the timing arcs for which annotated delays are to be removed. You cannot combine this option with *object_spec*.

-to *to_list*
: Specifies a list of pins or ports that are the endpoints of the timing arcs for which annotated delays are to be removed. You cannot combine this option with *object_spec*.

object_spec
: Specifies a list of leaf cells or nets for which all annotated delays are to be removed. You cannot combine this option with **-from** and **-to**.

remove_annotated_delay

Removes the delay information on selected nets, ports, or pins that was loaded from an external timing tool. You may want to use this command to ensure only Aprisa-calculated delay information is used and no delays slipped in from an external source.

Syntax

```
remove_annotated_delay objects \
   [-from pin_or_port] \
   [-to pin_or_port] \
   [-all]
```

where the arguments have the following meaning:

objects	Names of objects for which to remove the delay.
[-from *pin_or_port*]	Remove delay from the specified pin or port.
[-to *pin_or_port*]	Remove delay to the specified pin or port.
[-all]	Remove all annotated delay.

57

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790501	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056734
remove_annotated_transition Removes previously-annotated transition times from pins or ports in the current design.	**remove_annotated_transition** Removes the transition time information from the specified pins that was loaded from an external timing tool. You may want to do this to ensure only Aprisa-calculated transition time information is used and no transition times slipped in from an external source.
SYNTAX int **remove_annotated_transition** -all | *pin_list* list *pin_list*	**Syntax** remove_annotated_transition *pin_list* \ [-all]
ARGUMENTS -all Indicates that all annotated transition times in the design are to be removed. **-all** and *pin_list* are mutually exclusive; you must use one of these, but not both. pin_list Specifies a list of pins or ports from which annotated transition times are to be removed. **-all** and *pin_list* are mutually exclusive; you must use one of these, but not both.	where the arguments have the following meaning: *pin_list* — Names of the pins for which to remove the annotated transition. [-all] — Remove all annotated pin transitions.

58

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790507	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056736
remove_capacitance Removes capacitance on nets or ports.	**remove_capacitance** Removes user-specified capacitances from ports and nets that were set using the *set_load* Tcl command. The real extracted capacitances will be used instead.
SYNTAX string **remove_capacitance** *net_or_port_lis* list *net_or_port_list*	**Syntax** remove_capacitance *net_or_port_list* where *net_or_port_list* is the names of the nets or ports for which to remove capacitances.
ARGUMENTS net_or_port_list Specifies a list of ports and nets in the current design, whose capacitances are removed.	

59

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790508	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056738
remove_case_analysis Removes the case analysis value on input.	**remove_case_analysis** Removes case analysis that was set using the *set_case_analysis* Tcl command. Case analysis allows you to specify constant values for selected nets that are propagated through the design.
SYNTAX string **remove_case_analysis** *port_or_pin_list* list *port_or_pin_list*	**Syntax** remove_case_analysis *objects* where *objects* is the list of objects for which to remove case analysis.
ARGUMENTS port_or_pin_list Lists ports or pins for which the case analysis entry is to be removed.	

60

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790512	Pl's Ex. 266 (AP 13.11.rel.4) at ATopTech 0056740
remove_clock Removes one or more clocks from the current design.	**remove_clock** Removes all or a selected set of clocks in the design.

SYNTAX
 	
`string remove_clock -all | clock_list`
`list clock_list`
	Syntax
	
	`remove_clock [clock_list] \`
	`[-all]`

ARGUMENTS
 	
`-all`
Specifies to remove all clocks in the current design.
 	
`clock_list`
Specifies a list of collections containing clocks or patterns matching the clock names.
	were the arguments have the following meaning:
	
	`[clock_list]` List of clocks to remove.
	
	`[-all]` Remove all clocks.

61

remove_clock_gating_check

Captures clock-gating checks.

SYNTAX

```
string remove_clock_gating_check [-setup] [-hold] [-rise] [-fall] [-high | -low]
[object_list]
list  object_list
```

ARGUMENTS

-setup

Indicates the removal of the clock-gating constraint on the setup time only. If you do not specify either the **-setup** or **-hold** option, both setup and hold constraints are removed.

-hold

Indicates the removal of the clock-gating constraint on the hold time only. If you do not specify either the **-setup** or **-hold** option, both setup and hold constraints are removed.

-rise

Indicates the removal of the clock-gating constaint on the rising delays only. If you do not specify either the **-rise** or **-fall** option, constraints on both rising and falling delays are removed.

-fall

Indicates the removal of the clock-gating constaint on the falling delays only. If you do not specify either the **-rise** nor **-fall** option, constraints on both rising and falling delays are removed.

-high

Remove the high specification from the obejct list, previously set up by set_clock_gating_check command. This option has to be either high or low..

remove_clock_gating_check

Disables all clock gating setup and hold timing checks related to the SDC constraints set with the set_clock_gating_check Tcl command. This removal is permanent and is typically done after all the clock trees are generated. The timing analyzer still takes into account the clock gater cells, but their setup and hold constraints are assumed zero.

Note that the *ta* parameter, *timing_disable_clock_gating_checks*, disables all timing checks involving clock gater cells.

For an MCMM design, if you specify the *-all_scenarios* argument, the command applies to all scenarios regardless of the scenarios that are in the current session.

Syntax

```
remove_clock_gating_check \
   -all | clock_or_clockPins \
   [-all_scenarios]
```

where the arguments have the following meaning:

clock_or_clockPins	Collection of clocks or clock pins identifying the clocks for which to remove clock gating checks.
-all	Remove all clock gating checks for all clocks.
-all_scenarios	Remove the constraints for all scenarios. Note that this argument is only applicable for an MCMM design when there is no work scenario set.

-low
Remove the low specification from the obejct list, previously set up by set_clock_gating_check command. This option has to be either high or low.

object_list
Specifies a list of objects in the current design for which to remove the clock gating check. The objects can be clocks, ports, pins, or cells. If you specify a cell, all input pins of that cell are affected. If you do not specify any objects, the clock-gating check is removed from the current design.

63

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790515	Pl's Ex. 602 (AP 13.11.rel.4 Executable) at ATopTech 0057938

remove_clock_groups

Removes specific exclusive or asynchronous clock groups from the current design.

```
Command: remove_clock_groups
  remove clock groups
```

SYNTAX

Boolean **remove_clock_groups**
-physically_exclusive | -exclusive | -asynchronous
-name *name_list* | -all

list *name_list*

```
option:
  -all                      remove all clock groups
  -name *                   clok group list
  -physically_exclusive     physically exclusive
  -logically_exclusive      logically exclusive
  -asynchronous             asynchronous
  --get_option arg<1>       get option value
  --set_option ...          set option value
  --get_default arg<1>      get default value
  --set_default ...         set default value
  --system_default          use system default values
  --list_options            list current option values
  --load_options ...        load current option values
  --license                 list required licenses
  --help                    display command help
```

ARGUMENTS

-physically_exclusive
Specifies that groups set for physically exclusive clocks are to be removed. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-logically_exclusive
Specifies that groups set for logically exclusive clocks are to be removed. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-asynchronous
Specifies that groups set for asynchronous clocks are to be removed. The **-physically_exclusive**, **-logically_exclusive** and **-asynchronous** options are mutually exclusive; you must choose only one.

-name *name_list*
Specifies a list of clock groups to be removed, which matches the groups in the given names. You should use the **set_clock_groups** command to predefine these names. Substitute the list you want for *name_list*. The **-name** and **-all** options are mutually exclusive.

-all
Specifies to remove all groups set for exclusive or asynchronous clocks in the current design. The **-name** and **-all** options are mutually exclusive.

```
description:
  %remove_clock_groups -asynchronous -all
```


64

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790517	Pl's Ex. 602 (AP 13.11.rel.4 Executable) at ATopTech 0057938
remove_clock_latency Removes clock latency information from specified objects.	Command: remove_clock_latency [db:object_list] standard SDC command

SYNTAX
 	
string **remove_clock_latency** [-source]
[-clock *clock_list*]
object_list
 	
list *clock_list*
list *object_list*
	option:
	  -source remove source latency
	  -all remove all latency offsets
	  -offset remove latency offset
	  -inc_delay remove incremental delay offset
	  -ocv remove ocv latency
	  --get_option arg<1> get option value
	  --set_option ... set option value
	  --get_default arg<1> get default value
	  --set_default ... set default value
	  --system_default use system default values
	  --list_options list current option values
	  --load_options ... load current option values
	  --license list required licenses
	  --help display command help

ARGUMENTS
 	
-source
  Specifies that clock source latency should be removed.
 	
-clock *clock_list*
  Removes any network latency defined on the pin/port objects in object_list which refers the clocks in clock_list from the design. If the **-clock** option is supplied when object_list refers to clock objects, a warning is issued that the option is not relevant in this case and execution of the command proceeds as if -clock was not given. This option does not remove a more general latency setting without any specific clock.
 	
object_list
  Provides a list of clocks, ports, or pins.
	description:
	  This command is the same as standard SDC command.

65

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790519	Pl's Ex. 602 (AP 13.11.rel.4 Executable) at ATopTech 0057938
remove_clock_sense Removes unateness information defined on pins.	Command: remove_clock_sense <db:object_list> standard SDC command
SYNTAX string **remove_clock_sense** [-all] [-clocks *clock_list*] *object_list* list *clock_list* list *object_list*	option: -all — remove all clock unateness from current design -clocks collection — constraint applied to specified clocks only --get_option arg<1> — get option value --set_option ... — set option value --get_default arg<1> — get default value --set_default ... — set default value --system_default — use system default values --list_options — list current option values --load_options ... — load current option values --license — list required licenses --help — display command help
ARGUMENTS -clocks *clock_list* Optionally specifies a list of clock objects to be associated with the given pin objects in object_list. If the **-clocks** option is specifed, only the unateness specified for that particular clock domain will be removed. Otherwise, unateness information for all clocks passing through the given pin objects will be removed. The **-clocks** option can only remove clock sense predefined by **set_clock_sense -clock**. It does not remove the default clock sense setting for this given pin. -all Remove all unateness information in current design. object_list Lists of pins with predefined unateness to remove.	description: This command is the same as standard SDC command.

66

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790522-23	Pl's Ex. 602 (AP 13.11.rel.4 Executable) at ATopTech 0057938

remove_clock_uncertainty

Removes clock uncertainty information previously set by the **set_clock_uncertainty** command.

```
Command: remove_clock_uncertainty [db:object_list]
  Remove clock uncertainty constraints.
```

SYNTAX

```
string remove_clock_uncertainty
[object_list |
 -from from_clock
   | -rise_from rise_from_clock
   | -fall_from fall_from_clock
 -to to_clock
   | -rise_to rise_to_clock
   | -fall_to fall_to_clock]
[-rise]
[-fall]
[-setup]
[-hold]
[object_list]

list  from_clock
list  rise_from_clock
list  fall_from_clock
list  rise_to_clock
list  fall_to_clock
list  to_clock
list  object_list
```

```
option:
  -all                        remove all uncertainty
  -append                     remove append uncertainty
  -end                        remove end uncertainty
  -setup                      for setup only
  -hold                       for hold only
  -from collection            from clock
  -to collection              to clock
  --get_option arg<1>         get option value
  --set_option ...            set option value
  --get_default arg<1>        get default value
  --set_default ...           set default value
  --system_default            use system default values
  --list_options              list current option values
  --load_options ...          load current option values
  --license                   list required licenses
  --help                      display command help
```

ARGUMENTS

-from *from_clock* -to *to_clock*
: These two options specify the source and destination clocks for interclock uncertainty. You must specify either the pair of **-from/-rise_from/-fall_from** and **-to/-rise_to/-fall_to**, or *object_list*; you cannot specify both.

-rise_from *rise_from_clock*
: Same as the **-from** option, but indicates that *uncertainty* applies only to rising edge of the source clock. You can use only one of the **-from**, **-rise_from**, or **-fall_from** options. Use **-rise_from** instead of the obsolete option **-from_edge rise**.

-fall_from *fall_from_clock*
: Same as the **-from** option, but indicates that *uncertainty* applies only to falling edge of the source clock. You can use only one of the **-from**, **-rise_from**, or **-fall_from** options. Use **-fall_from** instead of the obsolete option **-from_edge fall**.

-rise_to *rise_to_clock*
: Same as the **-to** option, but indicates that *uncertainty* applies only to rising edge of the destination clock. You can use only one of the **-to**, **-rise_to**, or **-fall_to** options. Use **-rise_to** instead of the obsolete option **-to_edge rise**.

```
description:
  Remove clock uncertainty constraints from design.
```


67

-fall_to *fall_to_clock*
Same as the **-to** option, but indicates that *uncertainty* applies only to falling edge of the destination clock. You can use only one of the **-to**, **-rise_to**, or **-fall_to** options. Use **-fall_to** instead of the obsolete option **-to_edge fall**.

object_list
Specifies a list of clocks, ports, pins, or cells from which uncertainty information is to be removed. You can use either the pair of **-from/-rise_from/ -fall_from** and **-to/-rise_to/-fall_to** options or the *object_list* option, but you cannot specify both; they are mutually exclusive.

-rise
Specifies that uncertainty is to be removed for only the rising clock edge. By default, uncertainty is removed for both rising and falling clock edges. This option is valid only for interclock uncertainty, and is now obsolete. Unless you need this option for backward-compatibility, use **-rise_to** instead.

-fall
Specifies that uncertainty is to be removed for only the falling clock edge. By default, uncertainty is removed for both rising and falling clock edges. This option is valid only for interclock uncertainty, and is now obsolete. Unless you need this option for backward-compatibility, use **-fall_to** instead.

-setup
Specifies that only setup check uncertainty is to be removed. By default, both setup and hold check uncertainties are removed.

-hold
Specifies that only hold check uncertainty is to be removed. By default, both setup and hold check uncertainties are removed.

68

Pl's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790526	Pl's Ex. 602 (AP 13.11.rel.4 Executable) at ATopTech 0057938
remove_configuration Removes a configuration for multi-scenario analysis.	Command: remove_configuration Don't support.
SYNTAX boolean **remove_configuration**	option: --license list required licenses --help display command help

69

PI's Ex. 335 (PrimeTime 2006.12) at ATopTech 1790535-36

remove_data_check

Removes specified data-to-data checks previously set by set_data_check.

SYNTAX

```
string remove_data_check
{-from from_object
   | -rise_from from_object
   | -fall_from from_object}
{-to to_object
   | -rise_to to_object
   | -fall_to to_object}
[-setup | -hold]
[-clock clock]


object  from_object
object  to_object
object  clock_object
```

ARGUMENTS

-from *from_object*
: Specifies a pin or port in the current design as the related pin of the data-to-data check to be removed. Both rising and falling checks are removed. You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-to *to_object*
: Specifies a pin or port in the current design as the constrained pin of the data-to-data check to be created. Both rising and falling checks are removed. You must specify one of **-to**, **-rise_to**, or **-fall_to**.

-rise_from *from_object*
: Similar to the **-from** option, but applies to only rising delays at the related pin. You must specify one of **-from**, **-rise_from**, or **-fall_from**.

-fall_from *from_object*
: Similar to the **-from** option, but applies to only falling delays at the related pin. You must specify one of **-from**, **-rise_from**, or **-fall_from**.

PI's Ex. 602 (AP 13.11.rel.4 Executable) at ATopTech 0057938

```
Command: remove_data_check
  remove all data check constraints.

option:
  -from collection            not supported yet
  -rise_from collection       not supported yet
  -fall_from collection       not supported yet
  -to collection              not supported yet
  -rise_to collection         not supported yet
  -fall_to collection         not supported yet
  -clock collection           not supported yet
  -setup                      not supported yet
  -hold                       not supported yet
  --get_option arg<1>         get option value
  --set_option ...            set option value
  --get_default arg<1>        get default value
  --set_default ...           set default value
  --system_default            use system default values
  --list_options              list current option values
  --load_options ...          load current option values
  --license                   list required licenses
  --help                      display command help

description:
  remove all data check constraints.
```


70