Tweaker™ User Guide

Version U-2022.12, December 2022

SYNOPSYS®

Copyright and Proprietary Information Notice

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at https://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Feedback

Contents

About This User Guide

This manual is written for novice Tweaker-T1 users, or if you have previous Tweaker-T1 experience but need more technical guidance. Chapters 1-3 are important for the first time users, while advanced users can start from chapter 5.

Chapters 1-3 show Tweaker-T1's capabilities, as well as situations that call for Tweaker-T1 application in timing fixes. Chapter 4 describes in-depth preparation before fixing; this preparation includes all the required input data and STA sessions of each timing scenario. Beginning with Chapter 5 and 6, there is an introduction to the structure of each ECO operation and how to output ECO data for third party tools. A few topics such as Filter Report, TWF role, and Low Power are listed as independent chapters to provide a deeper understanding of what goes on backstage in the Tweaker ECO Platform.

Referencing this manual should provide an idea of how to implement Tweaker-T1 into specific design flows, and allows you to perform timing ECO in a fast and groundbreaking manner.

This preface includes the following sections:

New in This Release

Information about new features, enhancements, and changes, known limitations, and resolved Synopsys Technical Action Requests (STARs) is available in the Tweaker Release Notes on the SolvNetPlus site.

Related Products, Publications, and Trademarks

For additional information about the Tweaker tool, see the documentation on the Synopsys SolvNetPlus support site at the following address:

https://solvnetplus.synopsys.com

For quick-start or general training with operating Tweaker's UI, contact your Tweaker administrator for the following documents:

- *Tweaker-T1-fundamental-training.pdf*
- *Tweaker-T1-gui-training.pdf*

For detailed input command usage, enter `help` with the desired command in the [CMD] UI command line to search for matching commands and then use `man` with the desired command to see its details. For example, enter `help *slk*` to search all commands related to "slk", then use `man` to see its detailed usage. For variables, use the `printvar` command with the variable name to search for matching variable names, and then use `man` to see read through its purpose and usage. For example, type `printvar *sizing*` to search all variables related to "sizing" (See Figure 1), then enter `man slk_auto_sizing_max_shift_distance` for a detailed description of this variable (See Figure 2). You can also use the `man` command to check blocking code for finer details. For example, Figure 3 shows the results for "man B004" to see the blocking code information, as well as related variable settings. Respectively, this command works alongside the Error code. For a specific error message's details, enter `man E00151` to find out the error's root cause (See Figure 4). Tweaker also supports the TAB function to list incomplete commands or variables in the case that you are unsure of the name of a command or variable.

*Figure 1 Command Results: printvar *sizing**

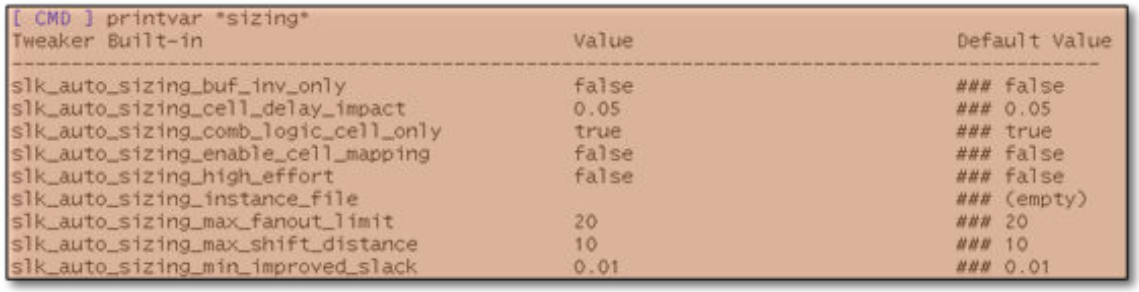
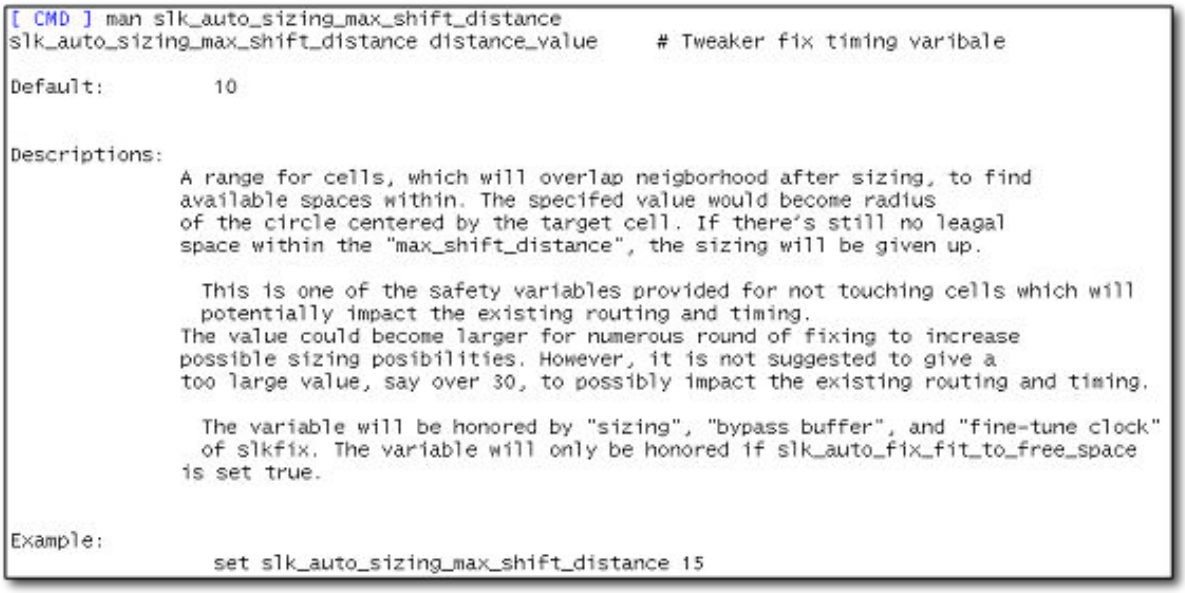


Figure 2 Command Results: man slk_auto_sizing_max_shift_distance



```
[ CMD ] man slk_auto_sizing_max_shift_distance
slk_auto_sizing_max_shift_distance distance_value     # Tweaker fix timing varibale

Default:         10


Descriptions:
               A range for cells, which will overlap neigborhood after sizing, to find
               available spaces within. The specifed value would become radius
               of the circle centered by the target cell. If there's still no leagal
               space within the "max_shift_distance", the sizing will be given up.

                 This is one of the safety variables provided for not touching cells which will
                 potentially impact the existing routing and timing.
               The value could become larger for numerous round of fixing to increase
               possible sizing posibilities. However, it is not suggested to give a
               too large value, say over 30, to possibly impact the existing routing and timing.

                 The variable will be honored by "sizing", "bypass buffer", and "fine-tune clock"
                 of slkfix. The variable will only be honored if slk_auto_fix_fit_to_free_space
               is set true.


Example:
                set slk_auto_sizing_max_shift_distance 15
```

Figure 3 Command Results: man B004

```
[ CMD ] man B004
B004    #       Blocked by Timing Window (Setup)

Descriptions

        Violation path doesn't have enough setup margin in timing window file during ECO.

Next step

        set slk_setup_target_slk 0.1
```

Figure 4 Command Results: man E00151

```
[ CMD ] man E00151
E00151  # Library cell is not defined($token)

Description

        This cell is not defined in the library.

Next step

        Please check if there is any cell missing in the library.

See also

        N/A
```

Feedback

For templates, reference the following directory, in which each sub-directory is related to the corresponding Tweaker-T1 operations:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1

You should review these templates before beginning the trial to understand how Tweaker supports Tcl scripts, and to better understand Tweaker-T1's timing fixing structure.

Conventions

The following conventions are used in Synopsys documentation.

Convention	Description
`Courier`	Indicates syntax, such as `write_file`.
`Courier italic`	Indicates a user-defined value in syntax, such as `write_file` *`design_list`*
`Courier bold`	Indicates user input—text you type verbatim—in examples, such as `prompt>` **`write_file top`**
`Purple`	• Within an example, indicates information of special interest. • Within a command-syntax section, indicates a default, such as `include_enclosing = true | false`
[ ]	Denotes optional arguments in syntax, such as `write_file [-format` *`fmt`*`]`
...	Indicates that arguments can be repeated as many times as needed, such as *`pin1 pin2 ... pinN`*.
|	Indicates a choice among alternatives, such as `low | medium | high`
\	Indicates a continuation of a command line.
/	Indicates levels of directory structure.
Bold	Indicates a graphical user interface (GUI) element that has an action associated with it.
Edit > Copy	Indicates a path to a menu command, such as opening the **Edit** menu and choosing **Copy**.
Ctrl+C	Indicates a keyboard combination, such as holding down the Ctrl key and pressing C.

Customer Support

Customer support is available through SolvNetPlus.

Accessing SolvNetPlus

The SolvNetPlus site includes a knowledge base of technical articles and answers to frequently asked questions about Synopsys tools. The SolvNetPlus site also gives you access to a wide range of Synopsys online services including software downloads, documentation, and technical support.

To access the SolvNetPlus site, go to the following address:

https://solvnetplus.synopsys.com

If prompted, enter your user name and password. If you do not have a Synopsys user name and password, follow the instructions to sign up for an account.

If you need help using the SolvNetPlus site, click REGISTRATION HELP in the top-right menu bar.

Contacting Customer Support

To contact Customer Support, go to https://solvnetplus.synopsys.com.

Statement on Inclusivity and Diversity

Synopsys is committed to creating an inclusive environment where every employee, customer, and partner feels welcomed. We are reviewing and removing exclusionary language from our products and supporting customer-facing collateral. Our effort also includes internal initiatives to remove biased language from our engineering and working environment, including terms that are embedded in our software and IPs. At the same time, we are working to ensure that our web content and software applications are usable to people of varying abilities. You may still find examples of non-inclusive language in our software or documentation as our IPs implement industry-standard specifications that are currently under review to remove exclusionary language.

1

Overview

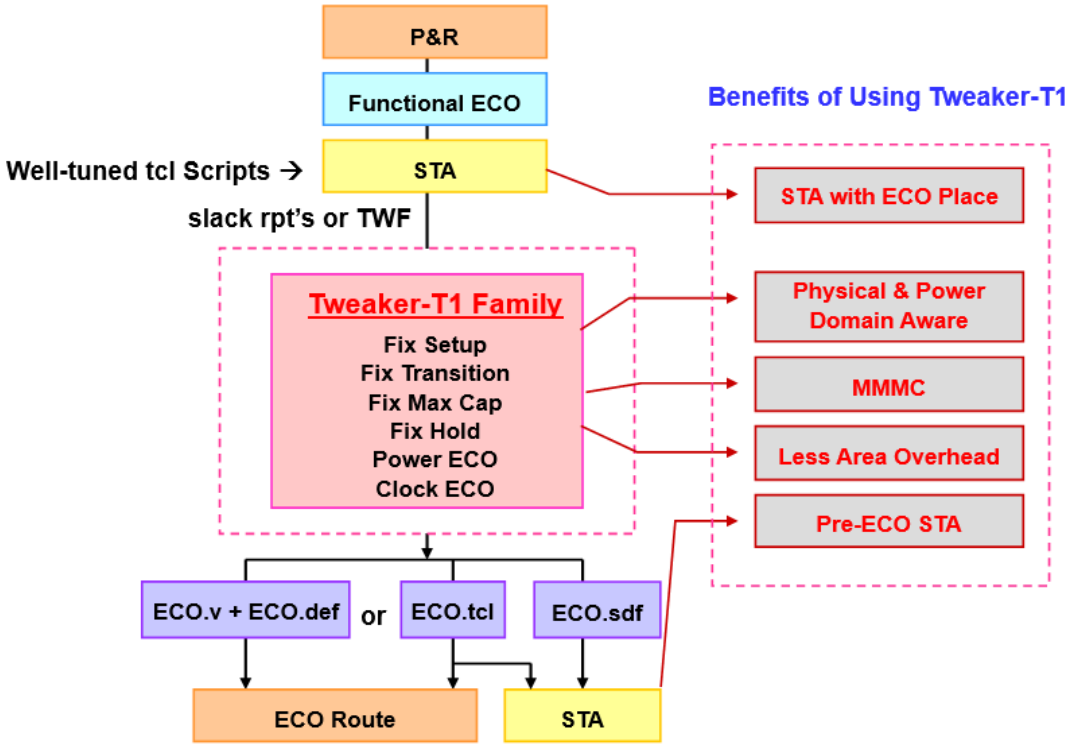
The following topics are:

The Tweaker-T1 Role at the ECO Phase

ECO (engineering change order) has long been considered “activities related to design change”. The scope of ECO is expanded to “all the incremental jobs in the ECO phase of design flow”, since today’s ECO jobs cover more than just design changes. During the ECO phase, the role of Tweaker-T1 is to help you solve a variety of timing issues that arise from functional ECO, or to assist P&R tools for multi-mode, multi-corner (MMMC) timing closure.

Feedback

Figure 5 Tweaker-T1 ECO Phase



To deliver high quality results with short runtime in the ECO Phase, the Tweaker-T1 ECO tool is built on a foundation of technology and architecture-specific ECO. The following topics provide a behind the scenes look into the tool's ECO thinking.

Incremental and Local Optimization

Typically, a design enters the Timing ECO Phase because its timing needs to be closed quickly while simultaneously preserving its previously achieved performance. The Timing ECO Phase requires all jobs to be completed INCREMENTALLY in a swift and reliable manner. To reach convergence without impacting a design's existing performance, Tweaker-T1's Funneling Safe ECO approach prioritizes the implementation of a relatively conservative local optimization over an aggressive whole scope optimization.

The Multi-Mode, Multi-Corner Data Structure

Tweaker-T1 holds a unique multi-mode, multi-corner (MMMC) data structure that helps you incorporate multiple timing scenarios simultaneously. The innovative MMMC data structure enables Tweaker-T1 to continuously make appropriate timing changes, based on each scenario, as soon as a cell is changed or moved. You can monitor the potential impacts on all scenarios, and decide what specific ECO action should be allowed. With assistance from the MMMC data structure, Tweaker-T1 is also capable of analyzing violation reports and focusing on the bottlenecks of violations spread throughout the design.

Figure 6 Example of MMMC Supported Tcl Scripts

```
set LIB {bc wc}
set PARA {cworst}
foreach lib $LIB {
 foreach para $PARA {
  begin_corner ${lib}_${para}
  set_group -lib -name $lib
  set_group -spf -name $para
  set_group -sdf -name ${lib}_${para}
  twfin -analysis_type on_chip_variation "${rpt_path}/${lib}.${para}.twf.gz"
  if { $lib == "bc" } {
       slackin -mode test -analysis_type on_chip_variation "${etos_path}/${lib}.${para}.scan.hold"
       slackin -mode test -analysis_type on_chip_variation "${lib}.${para}.scan.hold"
  } else {
       slackin -mode func -analysis_type on_chip_variation "${etos_path}/${lib}.${para}.func.setup
       slackin -mode trans -type sdf_max "/home/ael/etos/temp/run_tk_temp/wc.cworst.transition.rpt
    }
  end_corner ${lib}_${para}
 }
}
```

The following topics are covered:

- Physical Aware
- ECO Domain
- Signoff Data
- Hierarchical Design

Physical Aware

Besides MMMC, Tweaker-T1 is most celebrated for its physical awareness feature. The physical information supported by Tweaker includes floorplanning, placement, routing, blockage, placement rules, and FinFET rules. The physical awareness feature plays an important role for Tweaker-T1 to accurately estimate RC and timing. To be precise, this feature dramatically reduces the number of ECO iterations in comparison to other timing ECO solutions, without needing reference to the physical information. In the manual ECO

mode, this feature also helps in deciding the location of ECO cells, which allows for easier logic-to-physical team communication.

ECO Domain

Tweaker-T1 reads slack reports generated by mainstream signoff STA tools and builds its unique ECO Domain concept (see the ECO Domain topic) based on instances residing in the slack reports. By focusing only on the timing critical part of the design, the lightweight “ECO Domain” enables Tweaker-T1 to handle more than 100 signoff scenarios simultaneously for a large design. Based on the ECO Domain creation, you can easily monitor and instantly update timing results across different timing scenarios.

Signoff Data

By fully adopting the signoff data, Tweaker-T1 minimizes the amount of correlation issues. Tweaker-T1 reads the slack reports generated by signoff STA tools, and then performs timing fix jobs based on your specifications. For new wires, Tweaker-T1 estimates RC values based on the signoff RC numbers of the existing wires. This tool does not perform whole chip RC extraction using foundry technology files.

Hierarchical Design

Through just one run, Tweaker-T1 is capable of completing timing fixes on violation paths across all levels (TOP and all sub-designs) of a design’s hierarchy. Currently there are five classified types of hierarchical design that you might encounter, which are discussed in the Hierarchical Designs topic. Tweaker-T1 requires a minimal amount of engineering resources to achieve whole-chip timing optimization. When timing ECO is finished for the hierarchical design, Tweaker-T1 outputs a hierarchical netlist, partial DEF files, and block-by-clock Tcl commands for P&R tools; these facilitate each physical block owner in completing their ECO routing independently.

Overall, Tweaker-T1 provides a practical and user-friendly platform that operates in accordance with the physical awareness feature, as well as accurate timing prediction to efficiently deliver a high QoR ECO job. This bridges the gap between front-end and back-end users, allowing you to finish timing ECO jobs with minimal time and number of iterations.

Major Features of Tweaker-T1

Timing issues range across various categories: from transition, capacitance, fan-out, noise violations to setup and hold violation issues, from internal paths (register-to-register) to I/O paths, and from logic team to physical team. As a result of developing useful and flexible features, Tweaker-T1 is tailored to solve these timing issues. Unlike other whole-

Feedback

scope timing optimization tools, Tweaker-T1 strictly performs your job instructions on its designated area.

Tweaker-T1 features are introduced in the following topics.

- Fixing Hold Time Violations
- Fixing Setup Time Violations
- Fixing Maximum Transition, Capacitance, Fan-Out, and Noise Violations
- ECO Compare, ECO Place, and ECO Synthesis
- Hack SDF for Early Post-Layout Simulation
- Manual ECO With Tweaker-Lite
- Leakage Power ECO With Tweaker-P1
- Area ECO With Tweaker-A1
- Clock ECO With Tweaker-C1
- Metal ECO With Tweaker-M1
- Reliability ECO With Tweaker-R1
- Dynamic Power ECO With Tweaker-P2
- Multiple Level Latch Timing ECO
- Path Cloning for Manual ECO

Fixing Hold Time Violations

Tweaker-T1 eliminates hold time violations through all scenarios while simultaneously keeping setup time intact. To fix hold time, Tweaker-T1 provides VT swapping, sizing, add dummy load, high fan-out insertion, and delay insertion functions, as shown in Figure 7. This figure represents the standard hold time fixing flow for minimizing not only the area overhead, but also the leakage power overhead.

Figure 7 Tweaker-T1's Hold Time Fixing Flow

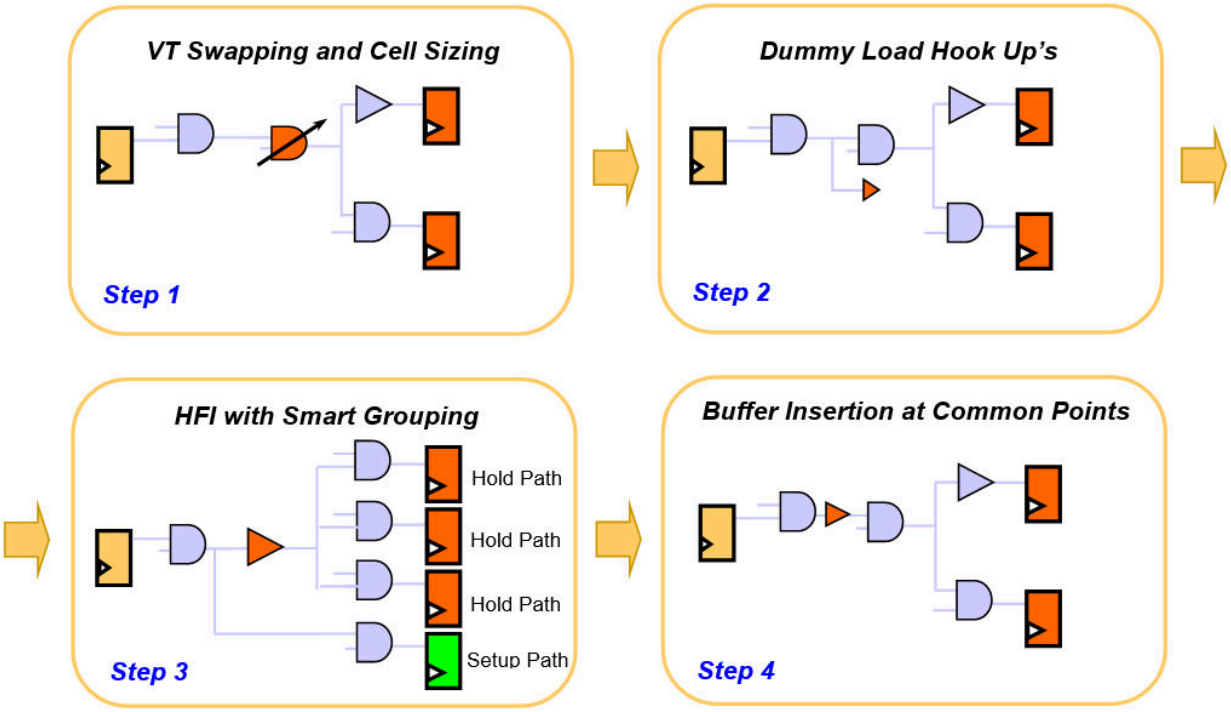
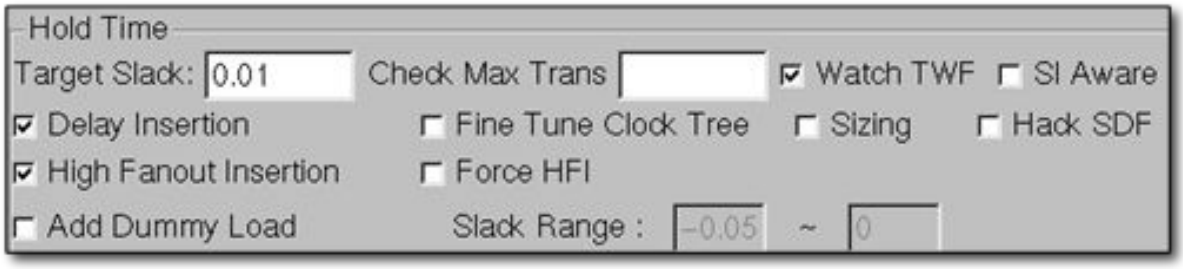


Figure 8 Hold Time Fix Settings



Tweaker-T1 provides three strategies that can fix these tiny violations with very little area and leakage power overhead: VT Swapping, Sizing, and Add Dummy Load. Typically, a histogram of hold time violations (number of violations vs. slack) will show a distribution, with larger quantities of violations near the zero slack value. Therefore, fixing these tiny violations using buffer insertion would be very costly in terms of area and power. Applying the three mentioned strategies effectively fixes the tiny hold time violations that are near zero slack.

VT Swapping will swap cells to different voltage thresholds based on specified mapping rules, while "sizing" will let Tweaker-T1 automatically choose the appropriate cell within

the same footprint. (Normally, Tweaker-T1 will choose slower cells for hold time violation fixing.)

Add Dummy Load will hook a load onto a net to slow its driver and, ultimately, fix hold time violations. The loading of the input capacitance and its wire will contribute to the required delay. As opposed to the traditional buffer insertion, hooking a small cell onto the net is recommended – this way there will be much less area overhead.

After executing the three strategies, remaining violations are solved by buffer insertion. However, Tweaker-T1 refrains from using the traditional buffer insertion method, and instead takes an advanced and refined approach.

Numerous efforts are spent in developing a high fan-out insertion algorithm to split the setup and hold paths, and also to insert buffers at drivers instead of endpoints. This algorithm shines in its ability fix a great portion of hold time violations using fewer buffers compared to the traditional method of inserting buffer chains at endpoints.

The constant shrinking of technology inevitably leads to serious signal integrity issues. As a solution, Tweaker features a smart buffer insertion method that decides at which design location a buffer can be inserted with minimal change to the routing pattern. Ultimately, this allows fixing to be done without developing signal integrity issues.

Sometimes, remaining hold violations cannot be fixed due to insufficient setup margin size or lack of free space for delay insertion. To solve the setup margins' shortcoming, Tweaker-T1 provides a feature that performs setup extraction, based on the hold violation paths, to gain more timing margin resources for the next delay insertion. As for shortages of free space for insertion, Tweaker-T1 provides not only an area recovery feature, but also a spare-cell-reusing feature, to resupply placement resources in design areas of high density.

All in all, Tweaker-T1 has various strategies to solve hold time violation issues with setup time retention and minimal area overhead. For more details, see the Start Fixing Hold Violations topic.

Fixing Setup Time Violations

Tweaker-T1 adopts a "90/10" strategy for fixing setup time, which assumes that 90% of the timing violations left unfixed by a P&R tool are a result of the correlation gap between P&R tools and signoff tools. These "easy-but-many" violations can be addressed by the automatic functions of Tweaker-T1. The remaining 10% the violations will be analyzed as really critical paths, which need to be addressed by human intelligence along with assistance of the numerous interactive functions, such as split load, split cell, and move cell, shown in Figure 9.

Feedback

Figure 9 Setup Time Fixing Settings

Most commonly used for fixing setup time violations, the sizing function applies automatic sizing up/down cells to improve timing. With each sized cell, the timing of its related paths will be updated accordingly and immediately. Besides the regular sizing function, the flexible VT swapping functions are also available options if the project adopts multiple voltage threshold cells.

To meet target slack, Tweaker-T1 can fairly judge when to bypass redundant buffers that exist in the setup paths, and size up its driver if needed. To split loads, Tweaker-T1 filters weak transition nets based on user-specified transition constraints and then insert buffers at non-critical fan-outs, which speeds up critical paths.

Other setup time fixing methods include Moving Cell and Pin Swap. Some critical setup violations arise from bad cell location, but Tweaker-T1 improves setup time by relocating cells based on critical timing; this is the Moving Cell function. The Pin Swap feature is derived from the Moving Cell function, and it only differs in terms of the A and B inputs in the library (for example, AND2), which might have different cell delays. To get better timing results, Tweaker-T1 can swap A and B's pin connections.

By default, Tweaker-T1 watches hold margins while performing setup ECO; however, some paths' hold margins are not enough, possibly preventing some setup violations from being fixed. To repair these remaining setup violations, you should first disable the hold-watching function before the setup ECO iteration, and later enable hold-watching again to continue smoothly onto the hold recovery flow.

In brief, the different techniques involved in fixing setup time include cell sizing, buffer bypassing, splitting load, cell moving, and pin swapping. Cell sizing allows you to control the amount of timing impact from a fix. These timing impact constraints help you to minimize worst negative slack (WNS) or reduce the violation count–total negative slack (TNS) as much as possible. In addition to cell sizing, Tweaker-T1 also provides buffer bypass, split load, move cell, or pin swap features; all of which help in optimizing WNS and TNS. For more detailed usage, see the Start Fixing Setup Violations topic.

Fixing Maximum Transition, Capacitance, Fan-Out, and Noise Violations

Compared to simply fixing setup or hold time, it is significantly more straightforward to fix pins and nets based on maximum transition, maximum capacitance, and maximum fan-out. In the traditional sense, max trans/max cap fixes become frustrating due to the

MMMC (Multi-Mode, Multi-Corner) aspect of advanced-process nodes – this causes constraint violations to not only reside in the slow scenarios, but also in the fast ones. However, Tweaker-T1's unique MMMC data structure takes a different approach in fixing all scenarios of constraint violations with extremely short runtime. All of this is possible with Tweaker-T1 capabilities such as VT swapping, Sizing, and HFS (High Fan-out Synthesis), and enable fixing while keeping other timing issues intact, especially for setup paths and clock tree paths.

Figure 10 Maximum Transition and Capacitance Fix Settings

After the initial steps of VT swapping and sizing-up, Tweaker-T1 performs HFS, an algorithm that intelligently groups fan-outs depending on the specified repeater buffer/inverter list, all without creating new module ports in the design. In the occurrence of huge fan-out nets, which typically cause max transition violations, Tweaker also provides a buffer tree-building command for efficiently optimizing max transition violations.

Since you might take different approaches to fixing constraint violations, Tweaker provides the option of fixing max transition using one of the 3 following conditions: slack value, actual value, or library value. An additional benefit provided by Tweaker-T1 is its capability to handle the cross-hierarchy violations within a hierarchical design.

Overall, the benefits of having Tweaker-T1 fix transition, capacitance, fan-out, and noise violations in the ECO phase are:

1. Tweaker always performs fixing exclusively based on the constraint violations reported by your signoff tool – no violations will be missed by your timing fix tool.
2. The MMMC structure enables Tweaker-T1 to handle all scenarios in the same run
3. The capability of handling cross-hierarchy violations while taking physical and power domain information into account.

For detailed usage, refer to the Start Fixing Maximum Transition, Capacitance, Fan-Out, and Noise topic.

Feedback

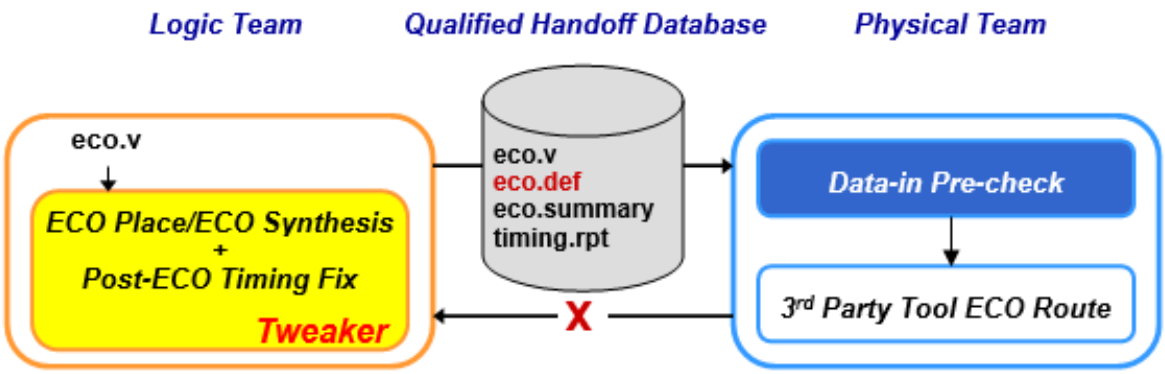
ECO Compare, ECO Place, and ECO Synthesis

In the ECO phase, functional ECO is one of the tedious issues a logic designer may encounter. Often, functional ECO jobs will induce timing ECO, and any solutions for completing a functional ECO job without damaging the existing timing would be far from perfect. When undergoing conventional functional ECO, an eco.v would be handed off to back-end teams either without timing checks or just with a rough timing check using wire load models. This lack of thorough checking gives rise to countless ECO iterations between each project's logic and physical teams.

To address this difficulty, Tweaker implements an improved flow that is developed to precisely predict the timing on functional ECO cells, as shown in Figure 11. Tweaker-T1 is a very creative tool in that it allows logic designers, unlike in the conventional sense with physical designers, to do ECO Place or ECO Synthesis. Along with this capability, logic designers can see accurate timing before handing off ECO data to their back-end team, and fix potential timing issues in advance.

Tweaker's ECO platform provides all design, physical, and timing information such that logical designer users can easily run the ECO job, and see whether this logical change is valid for implementation. As a result, these features leave the logical designer feeling more confident before delivering the ECO result, reduces the amount of unnecessary ECO iterations, and eases communication between different engineering teams.

Figure 11 Functional ECO Timing Prediction Flow



More specifically, ECO Compare performs a comparison between the ECO netlist and the original netlist to construct the ECO Domain, and ECO Place or ECO Synthesis then legalizes those ECO cells in the design. With placement and global routing, Tweaker-T1 provides more accurate timing prediction to those cells.

To reflect timing predictions, Tweaker-T1 outputs a partial – usually small – SPEF file which contains the timing information of only the ECO Domain itself. With this SPEF file

information, you can back-annotate the timing prediction to the signoff STA tool and see if there are any potential timing impacts caused by the functional ECO jobs. For detailed usage, refer to the ECO Compare, ECO Place, and ECO Synthesis topic.

Hack SDF for Early Post-Layout Simulation

The development of this feature targets generating a hacked SDF file for logic designers to start their post-simulation earlier, even while many timing violations (particularly hold time violations) are still left unresolved. The truth is that the hold time violations could paralyze the simulation and hinder you from addressing other potential issues in their design. To help designers do post-simulation as early as possible, Tweaker-T1 can “pretend” to solve timing violations based on TWF, or the violation report, by only hacking SDF and leave the netlist as is. The advantage of choosing TWF over slack reports is that the TWF downplays the “nworst” problem from generating slack reports, and guarantees a complete hold time clean SDF with setup time reduced as much as possible. The hacked value is carried to one of the min/max triplet columns in the SDF file, and designers can select one of them to get annotated at the post-simulation stage.

In summary, with the help of timing-violation-free SDF, designers can start simulating and surveying potential problems a lot earlier, while keeping the current design unchanged. For detailed usage, refer to the Hack SDF for Post-Sim topic.

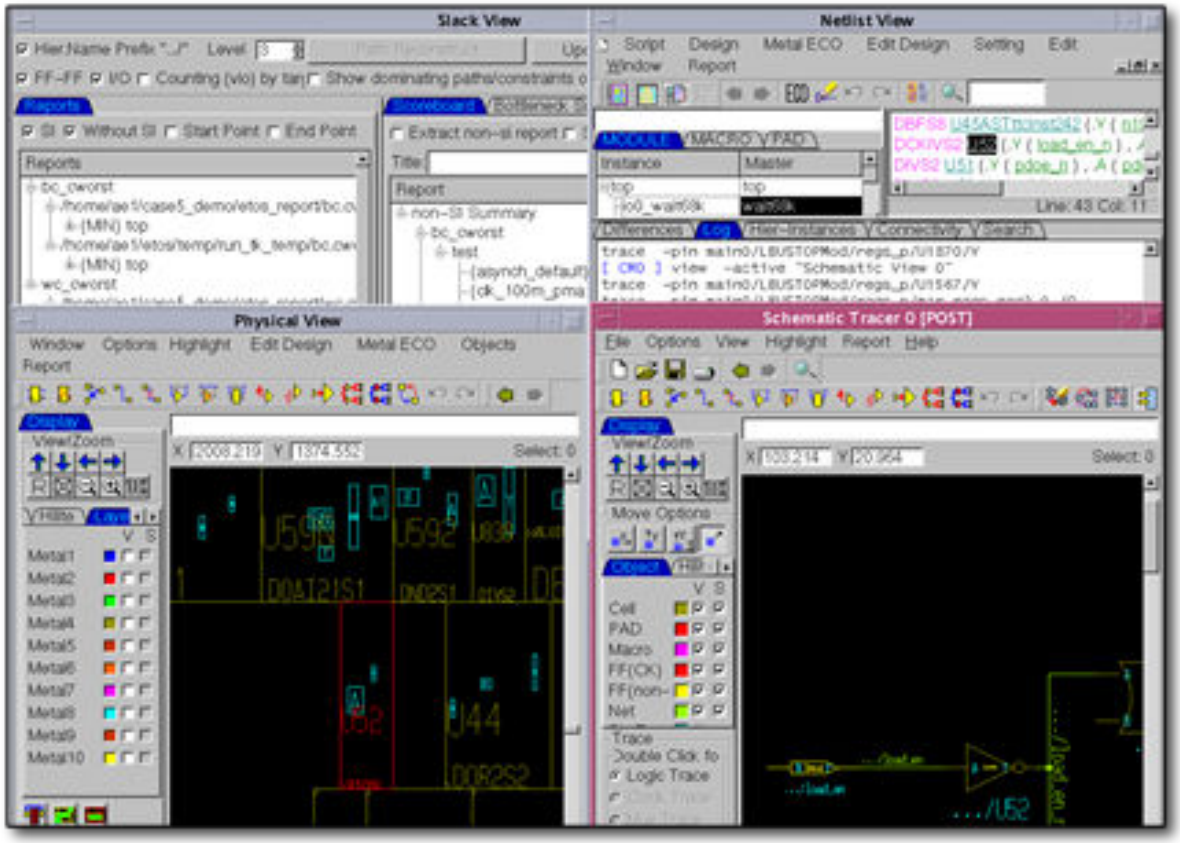
Manual ECO With Tweaker-Lite

The Tweaker-Lite Baseline Platform is developed to be convenient in applying manual ECO jobs on each of your designs. Tweaker’s platform consists of the Netlist View, Physical View, Schematic View, Slack View, and Path View. The Netlist View, the main window, includes a module browser, netlist editor, and much more. The Physical View displays cell placement based on DEF. The Schematic View lets you trace and edit your netlist according to your preferences. The Slack View collects all the slack reports under the MMMC structure and provides bottleneck analysis. The Path View shows individual real timing paths from startpoint to endpoint, in both launch and capture paths. Cross probing and editing capabilities are available in all views, and every view will be updated together after any edits are made. Moreover, the Tweaker platform supports numerous interactive timing fix functions, and timing is updated instantly when any cell is changed, or even moved.

Due to the wide range of different manual ECO jobs, a majority of these manual ECO jobs have corresponding functions built into the Tweaker-Lite GUI (See Figure 12). Some frequently used functions are set with a user-friendly semi-auto mode to help you easily complete manual ECO. The functions range from simple buffer insertion and cell movement, to more complicated functions like splitting load on a driver or cloning the cell to share loading. Most of these interactive functions are described in the *tweaker_baseline_training.pdf* training slides.

Feedback

Figure 12 Tweaker-Lite GUI



Since Tcl is supported in Tweaker, the Tweaker-Lite platform accepts batch editing through Tcl scripts and is therefore convenient when your ECO jobs include continuous editing.

Leakage Power ECO With Tweaker-P1

Tweaker-P1 is an independent operation under the Tweaker-T1 Family, which implements VT Swapping and Sizing Down algorithms. These processes minimize the leakage power of your design by removing excessive parts that were previously induced by synthesis and optimization. The two key assets of Power ECO include the leakage power-minimizing capability and the ability to keep original setup time intact. Similarly, the VT Swapping at the ECO phase results in the least design changes, so Tweaker-P1 decides to do cell swapping entirely from the signoff data. In accordance to this method, Tweaker-T1 can achieve near-exhaustive swapping without significant impact on the original timing.

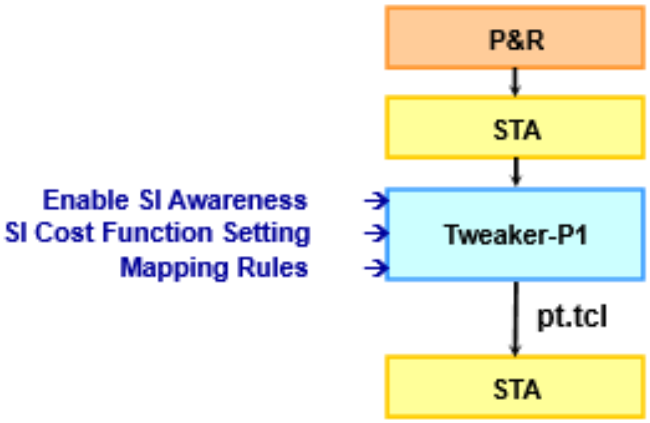
Another feature unique to P1 is its way of evaluating cell swaps. Tweaker-P1 does not always base cell swaps on the highest swap ratio; instead, they evaluate swap necessity completely on the leakage power information (in the .lib file). Moreover, the MMMC

Feedback

structure lets Tweaker-P1 assess leakage power at a specific scenario while watching the timing impact on another scenario. This process is essential for advanced process nodes such as 40 nm and below, where high VT cells sometimes leak more power than regular VT cells in slow scenarios. See Leakage Power ECO With Tweaker-P1 for more detailed usage.

However, swapping cells to become slower can induce extra delay, which is caused by coupling capacitances. Fortunately, Tweaker-P1 is equipped with signal integrity (SI) awareness options to avoid cell swapping at potentially risky candidates.

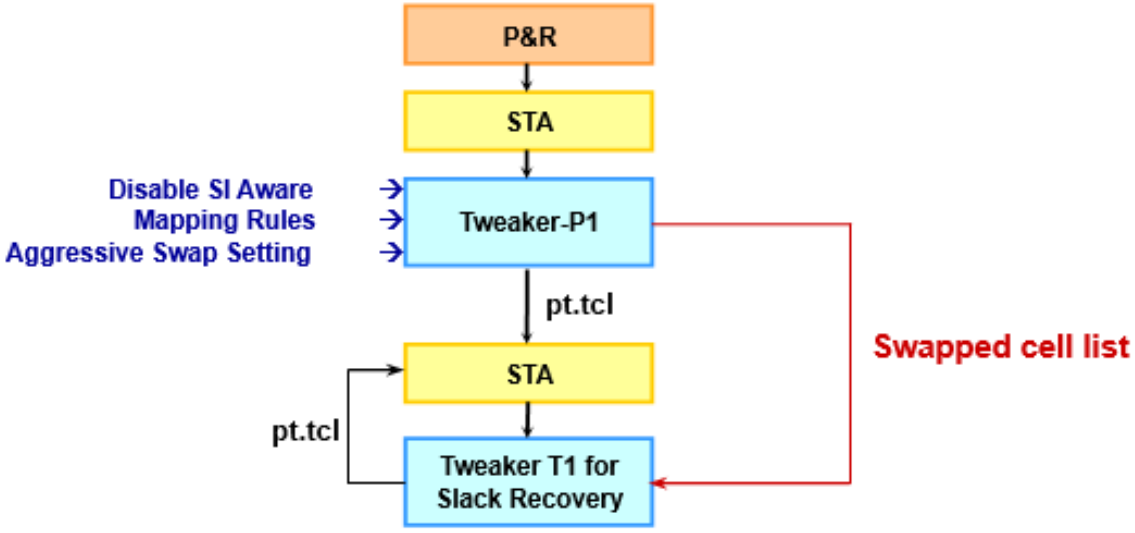
Figure 13 Power ECO With Tweaker-P1



To perform more aggressive swapping, particularly for CPU design, you can turn off the SI aware options and adjust the aggressive swap settings higher. Then, you can use Tweaker-T1's fix timing capability to "recover" those induced timing violations, especially the SI-induced violations. This approach only costs one more ECO iteration but gives almost exhaustive swapping. Additionally, timing convergence is guaranteed within one ECO iteration because the timing recovery is based on the previous cell-swap list output.

Feedback

Figure 14 Power ECO With Aggressive Swapping



Moreover, since Tweaker-P1 shares the same database as Tweaker-T1, MMMC is well-supported. An additional benefit of Tweaker-P1 is its physical awareness feature, which is especially useful if you intend to do swapping between cells of different sizes. Tweaker-P1 also supports VT Swapping and cell sizing, both of which prevent cell overlapping and placement rule violations. The proposed flow is: LVT > RVT > HVT > Sizing Down.

Area ECO With Tweaker-A1

Tweaker-A1's area recovery flow allows you to remove redundant buffers or inverter-pairs, and release more free area for future ECO operations. There are cases in which you can experience over-fixing on a high-density design through a third party tool; often, these third party tools optimize timing path redundancies due to over constraints in SDC, miscorrelation with signoff tools and APR tools, or design limitation cased on hierarchical structure. For these reasons, the third party tools can easily insert unnecessary buffers, but these insertions are difficult to remove later on.

In high density designs that lack enough free space, timing closure is extremely difficult to achieve in the next level of ECO. Fortunately, Tweaker lessens the severity of this issue using a unique area recovery flow, in which multiple critical setup and hold corners can be read in. Based on this timing information, Tweaker evaluates whether each buffer or inverter pair can be removed while simultaneously maintaining enough setup, hold, and max transition margin. Through this methodology, you are allowed another chance to release more free space and use it for the next timing optimization.

Clock ECO With Tweaker-C1

Tweaker-C1, an independent operation under the Tweaker-T1 Family, provides a clock fine-tuning feature that adjusts clock network components incrementally. This feature applies “useful clock skew” to either minimize the WNS for setup time or reduce the violation count for hold time. Unlike other skew tuning solutions, Tweaker-C1 relies on the signoff slack reports to do skew tuning. In this way, runtime is much shorter because the feature applies local and incremental skew tuning while watching impact on the whole clock network. And because this skew tuning technique relies heavily on the signoff data, accuracy is exceptionally improved.

Ultimately, with proper depiction on clock network through STA reports, Tweaker-C1 can activate its clock fine-tuning function to gain benefits from either minimizing WNS for setup time or reducing violation count for hold time. To verify the post-fixing result, you can output a partial SPEF for STA tools to back-annotate and see the STA’s what-if result before handing off the ECO data. For “sizing” only, an eco.tcl file for PrimeTime will be even more convenient for STA what-if analysis. The use simply needs to restore the PrimeTime session, source the eco.tcl file, then run the `update_timing` command. The result for this procedure will be very trustworthy, as the RC network will be almost unchanged. Refer to Clock ECO With Tweaker-C1 for more details.

Metal ECO With Tweaker-M1

Tweaker-M1 gives the flexibility on “Metal Layer Change”; remap and timing closure after tape-in or tape-out of base layers where only metal layer changes are allowed. This feature supports mixing resources between spare cells and gate-array cells when remapping and timing closure to get the best result with a minimum number of resources used. Tweaker-M1 can close timing with auto-mechanism by using the surrounding resources near violation paths. Featuring cell stealing methodology for timing closure, you can close timing even when there are not enough resources near the timing violation path.

You can perform manual ECO operation in “Metal Layer Change” with timing prediction in the Tweaker GUI. Instances are displayed in various colors to distinguish which cells are safe to steal, will fix the violation, or improve the violation without degrading the stolen instance’s path when using cell stealing to sizing up/down and insertion. You can implement or verify the post-fixing result of the ECO changes to the APR or STA tool by sourcing the Tweaker-generated ECO Tcl file.

Reliability ECO With Tweaker-R1

Reliability is the most important factor in differentiating the design, especially for the profit margin. A reliable design implementation helps improve the yield and enhance the operation of chips across differentiating operating conditions. PPA alone cannot

differentiate the chip, but PPA along with improved reliability and robustness is key for chip success.

With new design trends and challenges day-by-day, it is getting difficult to cope up with uncertainties that are encountered during chip design. Advance Process nodes adds much more complexity during the manufacturing process which requires to be handled very precisely in-order-to add predictability. It is necessary that we have a solution that is sustainable and gives a good yield of the chip after it manufactures.

These reliability solutions are even more significant for the chips manufactured for Automobiles and Aeronautics devices as the failing of such chips in even extreme conditions are very costly and expensive, so across various PVT and with extra guard band these devices cannot afford to fail.

Tweaker-R1 is a unique ECO solution available which provides reliability ECOs to make the silicon more robust and improves the yield across various technology. There are various challenges that can be solved in Tweaker to improve reliability of the chip. Some of them are VMIN ECO, IR ECO, IR Based Timing ECO, Peak Power reduction, and so on.

The Timing Reliability Solution

Timing is the first and foremost concern of any signoff cycle. The timing reliability solution helps to minimize large variation cells available in the design, especially on timing critical paths. This add extra predictability during design convergence.

The IR Reliability Solution

IR signoff is an important parameter to estimate the voltage drop; this becomes more significant when IR drop cells are present in the timing critical paths. It is important to generate IR and the timing map and optimize the path based on IR drop value. The optimizations are IR aware, so that it does not increase the additional IR drop. In many circumstances it helps to reduce the IR drop with several area recovery techniques by downsizing the cells and moving non-critical cells outside the IR critical region.

Dynamic Power ECO With Tweaker-P2

Tweaker-P2 is one of the features of the Tweaker-ECO family which can be integrated in the pre- and post-timing ECO stage for dynamic power optimization. As with lower technology nodes, the dynamic power contribution toward total power is more than 90%. So, it is significant to optimize this and reduce the contribution as much as possible. The Tweaker-P2 solution is completely physical and MMMC timing aware and optimized the power without sacrificing any timing or physical DRC.

With Tweaker-P2, Tweaker has the capability to reduce the dynamic power with toggle information or without toggle information. Ideally, Tweaker expects a fully propagated activity file from the PTPX, in case It is not available, Tweaker can use the default activity with its Vector-less flow. Tweaker easily supports both kinds of requirements.

Feedback

Apart from the regular Timing ECO inputs, Tweaker needs the following additional information to set up a flow for Tweaker-P2 (Dynamic Power):

- Libraries required which include internal power table
- A toggle file (supports a net-based/pin-based toggle file)

In the following some scenarios, you might prefer to use the vector-less flow:

- When you want quick and easy total power ECO without toggle files
- When the dynamic power analysis settings are not finalized (the lack of toggle data files)

Tweaker also provides flexibility to optimize the internal switching and leakage power in the same run. This method helps to save the total power optimization.

Tweaker also generates the ECO Tcl file for PrimeTime PX what-if and implementation tools. Post ECO, you can also do the what-if analysis with PrimeTime PX by reporting power (with the `report_power` command) and reporting switching activity (with the `report_switching_activity` command) pre- and post-ECO in the PrimeTime PX shell to see the results.

Multiple Level Latch Timing ECO

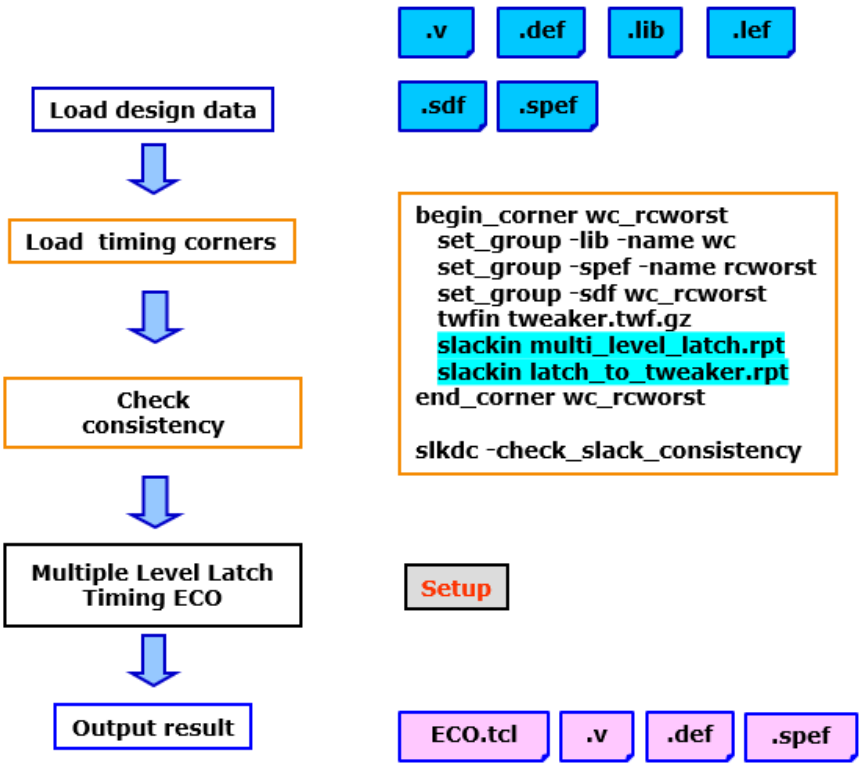
You can perform Multiple Level Latch Timing ECO to fix multiple level latch timing by recovering more time borrowing in earlier levels of the design. Use this feature to report input cone paths to create a latch-based TWF update engine file.

The following are the steps for the Multiple Level Latch Timing ECO flow:

1. Load the design data
2. Load the timing scenarios
    - Provide the multiple level latch report from the original report
    - Provide the latch_to_tweaker.rpt file from the latch database script
3. Check the consistency
4. Enable Multiple Level Latch Timing ECO
    - Return timing borrowing
    - Create Timing ECO
    - Update time borrowing
5. Output the results

Feedback

Figure 15 The Multiple Level Latch Timing ECO Flow

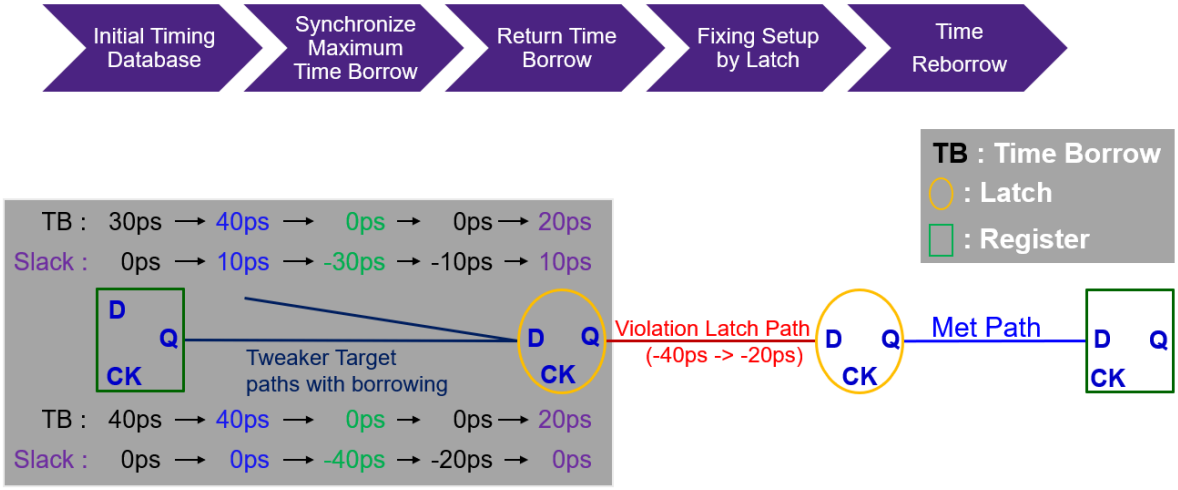


After you load the design with the latch input cone paths, you can create an additional ECO domain for the latch paths by following these steps:

1. Run the `slkfix -create_setup_domain_by_latch_report` command to create a domain
2. Run the `slkdb -check_slack_consistency` command to initialize the timing database
3. Run the `slkdb -update_twf_by_latch_report` command to synchronize the maximum time borrow, which picks the worst time borrow window for each endpoint
4. Run the `slkdb -adjust_time_borrow_for_setup` command to return the time borrow from the last stage to the previous stage
5. Run the `slkfix -setup -path_list $latch_corresponding_path_lists` command to fix timing in the previous stage
6. Run the `slkdb -update_time_borrow` command to update the time borrow

Feedback

Figure 16 Multiple Level Latch FSM



Path Cloning for Manual ECO

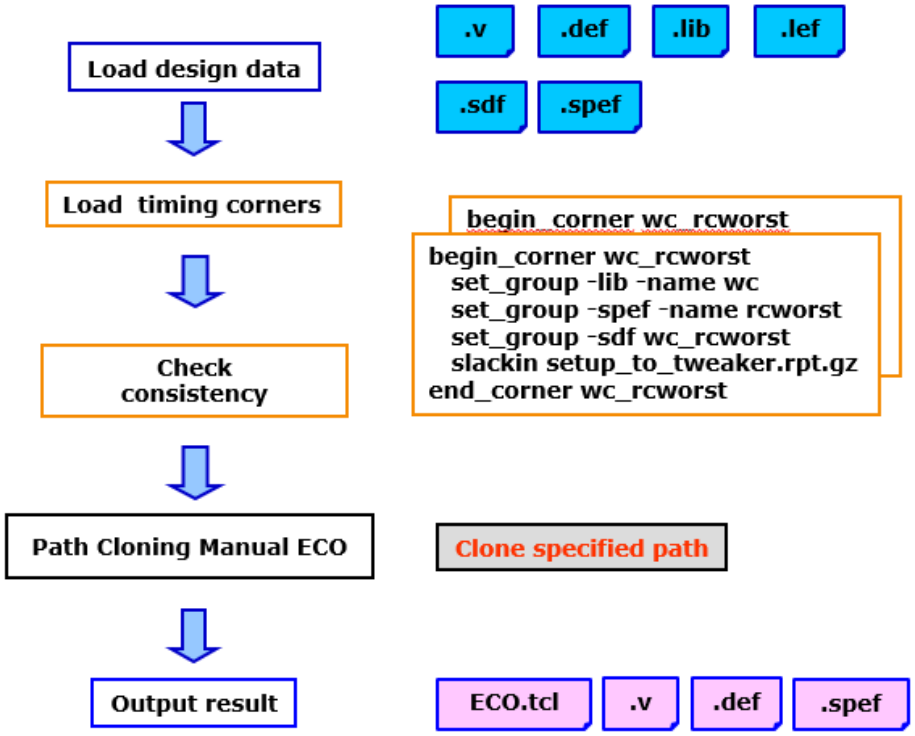
You can perform path cloning to fix critical timing paths using Path Cloning for Manual ECO. Path cloning gives you more flexibility on timing paths in the ECO.

The following are the steps for the Path Cloning for Manual ECO flow:

1. Load the design data
2. Load the timing scenarios
3. Check the consistency
4. Enable Path Cloning for Manual ECO
5. Output the results

Feedback

Figure 17 The Path Cloning for Manual ECO Flow



To perform path cloning for manual ECO from the command line use the following syntax. You must provide the start and end hierarchical pins. Use the `-from_pin` option to specify the start pin and the `-to_pin` option to specify the end pin of the cloned path.

```
[CMD] clone_path -from_pin start_output_pin -to_pin end_input_pin
```

For example,

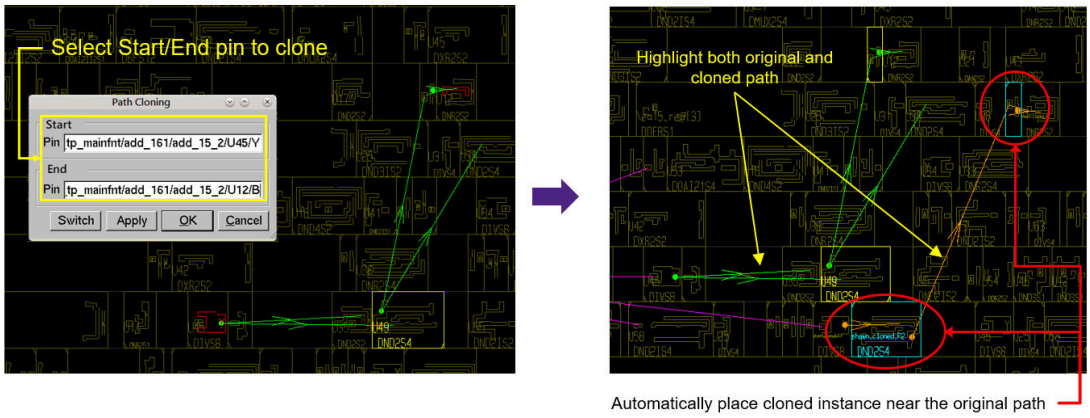
```
[CMD] clone_path -from_pin U_dsub/U_phydig/U25/A
                 -to_pin U_dsub/U_phydig/U42/Y
```

The tool automatically places the cloned path near the original path.
To adjust the maximum shift distance for the cloned path, set the `slk_path_cloning_max_shift_distance` variable. The default is 20. If there is no available free space in the search range, the tool allows overlap and places the cloned path near the original path.

You can also perform path cloning for manual ECO from the GUI. In the Physical View, right-click on the start pin you want to clone, then select **Path cloning** from the menu. The Path Cloning window appears, where you can specify the start and end pins.

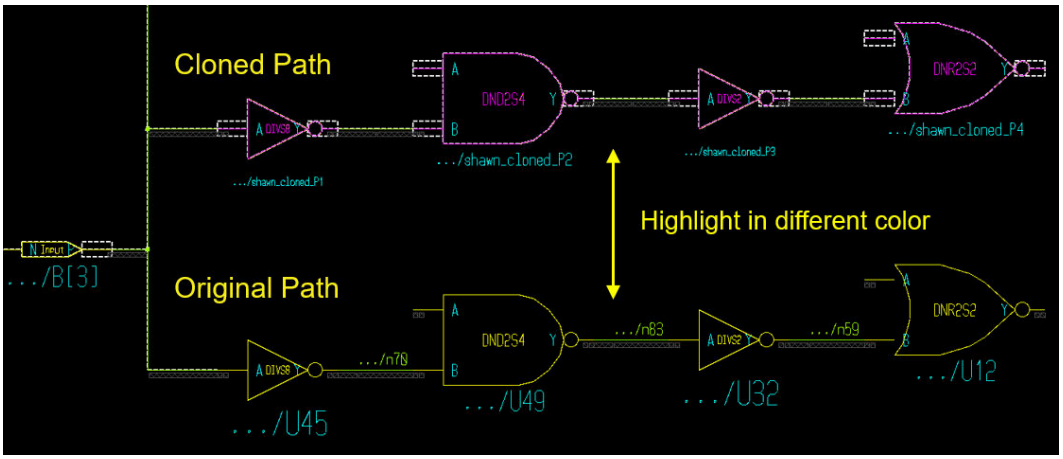
Feedback

Figure 18 Cloned Path in the Physical View



You can view the cloned path in the Schematic View.

Figure 19 Cloned Path in the Schematic View



2

Fix Timing Overview

The previous sections plot a picture of Tweaker-T1's role and capability at ECO phase, and how Tweaker-T1 deals with all kinds of timing issues. This topic further extends the discussion about the backbone and behind-the-scenes concepts of Tweaker-T1.

The following topics are covered:

- Timing Fix Based on STA Reports
- ECO Domain
- Instant Timing Update

Timing Fix Based on STA Reports

As an ECO tool, Tweaker-T1 performs local timing optimization instead of whole chip timing optimization. Local timing optimization serves as a strong benefit to timing fixes by keeping existent timing intact without making any detrimental impacts; this is an efficient way to meet timing closure. Tweaker-T1 instead reads signoff quality STA reports to construct the main focus: the local "ECO Domain". Normally the local domain–built based on violation paths in the STA reports–are considered ECO Domains, and these will be discussed further in the next topic.

Aside from using the report-based ECO Domain in normal timing fix procedures, you can also use the timing window file (TWF) to build the timing ECO domain if the "nworst" number fails to cover a majority of the timing violations. To generate TWF files from signoff STA tools, you must source the Tcl scripts. These TWF files carry each pin's slack/transition information, clock component list, and path-based analysis (PBA) timing information. With this information, you can easily find the negative slack and build setup violation paths. All in all, Tweaker-T1 is capable in setup fixing during the initial stage where you typically have difficulty in generating their length timing violation reports.

Some operations, such as Power ECO or Area ECO, require whole scope domain to complete its job. Through TWF parsing, Tweaker can use every pin's positive and negative values to construct the whole scope domain for leakage and area optimization.

Feedback

ECO Domain

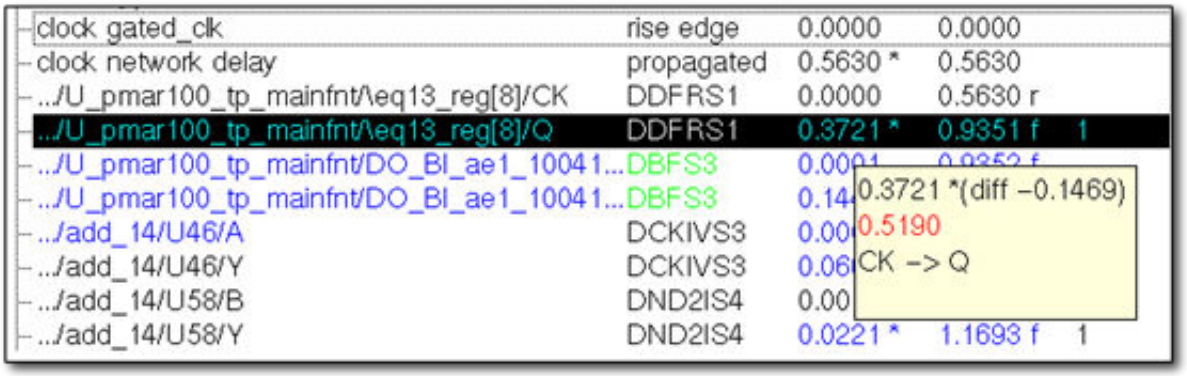
Tweaker-T1 constructs its ECO Domain based on the input STA reports, timing window files (TWF), or the output of ECO Compare. Essentially, ECO domain is defined as the domain of interest. The basic principle for handling each ECO Domain in Tweaker-T1 is to maintain its boundary's timing information so that when an ECO action is performed inside the domain, Tweaker-T1 can update timing accordingly without needing to rebuild the whole chip's time graph. For this reason, Tweaker-T1 is able to update timing much faster than other STA based solutions. In the MMMC world, this concept enables Tweaker-T1 to handle more than 100 scenarios of timing ECO at the same time.

With STA what-if analysis, Tweaker-T1 outputs partial SPEF and SDF files for each scenario based on each ECO Domain, such that you can back-annotate in the STA tools. As a whole, SPEF files carry all RC information inside the ECO Domain; however, fractional SPEF files are deemed "partial SPEF's" to distinguish from the original SPEF files. In this sense, Tweaker-T1 outputs partial SPEF's for all RC corners in a single run. You can also use "partial SDF's" for each scenario, and thus complete STA what-if analysis before the ECO is handed off.

Instant Timing Update

Tweaker-T1's innovative ECO Domain construction enables you to benefit from instant timing updates and feedback from the GUI, whether they result from a single buffer insertion or a cell move (See Figure 20). Furthermore, the built-in undo/redo mechanism makes applying manual ECO jobs in the Tweaker GUI tremendously more convenient.

Figure 20 Instant Timing Update After Buffer Insertion



In essence, Tweaker-T1 uses the complete timing information in each ECO Domain to immediately produce timing updates and feedback in the GUI. This helps to keep you informed about the status of whether timing is already met or still violated.

3

Starting Tweaker

To start Tweaker, you need to activate Tweaker's license server and correctly set up the Tweaker environment. To do this, you should reference the template for setting up the Tweaker environment in the following directory:

$Tweaker_Install_Dir/dorado/etc/cshrc.tweaker

After the environment is set up, you can verify these settings by inputting "which tweaker" in the machine's command prompt. If the executable path in Example 1 appears, you can proceed to the next step.

Example 1 Executable Path

```
linux101{ae1}:/home/ae1/etos/temp> which tweaker
/home/ae1/tool/tweaker_release/dorado/bin/tweaker
```

By entering `tweaker -help` in the command prompt, you can find all switch options compatible with "tweaker". For example, `tweaker -t` will start Tweaker-T1 and `tweaker -l` will start Tweaker-Lite. All different ECO function commands shown in Example 2 will start Tweaker using their corresponding licenses.

Example 2 Available Tweaker Options

```
linux101{ae1}:/home/ae1/etos/demo> tweaker -help
tweaker [-l|-t|-m][-x "execute command"][-cmd "command file"] [-log "log
 name"] [-exit] [-pre_check|-check_only|-no_check]
    -l: tweaker lite
    -t: timing eco
    -m: metal eco
    -no_gui: disable gui
    -exit: exit tweaker at the end, normally used with -cmd
```

The following topics are covered:

- Command Mode
- GUI Mode
- Tweaker Log
- Exiting and Pausing Tweaker
- Input Data Preparation

Feedback

Command Mode

To run Tweaker-T1 without the GUI, or to invoke and load data into Tweaker using the command prompt, you have the option to add `-cmd tcl_file` or `-no_gui` switches to the `tweaker` command. For instance, inputting `tweaker -t -cmd load_design.tcl` into the command prompt will launch Tweaker-T1 and run “load_design.tcl” in the command mode. With the `-no_gui` switch, the Tweaker shell will show up in the command prompt as `tweaker_shell>` after an ECO job is finished. Without using the `-no_gui` switch, the Tweaker GUI will open at the end of the job to allow you to analyze the ECO results. The `-exit` switch is often used in command mode to quit Tweaker after the jobs finish running. The `-pre_check` switch enables Tweaker to check whether or not all input data exists before executing the job run. This pre-check feature, along with `-check_only`, verifies data before each ECO job and prevents Tweaker from realizing very late into a run that the several essential inputs do not exist.

Because Tweaker is a Tcl-based tool, every Tcl syntax error will interrupt the proceeding job with an error message. The same interruption would happen for other abnormal events, such as non-existent files specified in the Tcl file or .lib netlist parse errors, and force Tweaker to abort.

GUI Mode

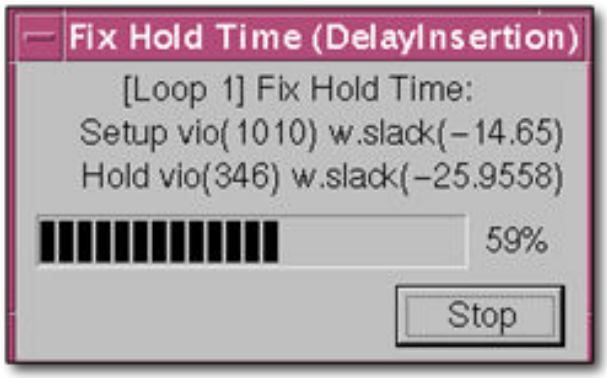
An alternative way to launch Tweaker is to directly open the GUI mode. Entering `tweaker [ -l | -t | -m ]` into the command prompt starts the Tweaker GUI. The Tweaker greeting image in Figure 21 appears and automatically proceeds to the Netlist View, Tweaker’s main operating view. In the Netlist view, you can source any Tcl file through the UI’s tool column.

Feedback

Figure 21 Tweaker's Greeting Image

Some of the many Tweaker GUI advantages are that it keeps a "recent scripts" list, as well as a status window as shown in Figure 22. This window allows you to track the status of your timing fix. You can also stop your timing fix.

Figure 22 Status Window



Tweaker Log

Starting Tweaker with the `-log` option – for instance, `tweaker -t -cmd load_design.tcl -log ./log/eco.log` – will enable the tool to create a log directory under the specified name, and write in all of the session's log messages. If you activate Tweaker without using the `-log` option, Tweaker will save the log message under the

Feedback

session's working directory by default. Each log directory includes log, error, cmd, and nlcmd information as such:

Tweaker.log – all design information related to command execution

Tweaker.error – all warning and error occurrences during ECO runs

Tweaker.cmd – saves all Tweaker execution commands provided by Tweaker scripts

Tweaker.nlcmd – Tweaker's ECO Tcl; you can reuse the eco.tcl for many purposes

Exiting and Pausing Tweaker

In the command mode, if you prefer to exit Tweaker directly without returning to the GUI, use the `-exit` option with the `tweaker -t -cmd` command. For instance, entering the following command returns you to the command prompt after the Tcl file has been executed completely in Tweaker:

```
tweaker -t -cmd tcl_file -exit
```

In the GUI mode, you can exit Tweaker by either selecting **Exit** from the **Script** pull-down menu or entering `exit` in the Tweaker command prompt.

Figure 23 To Exit Tweaker GUI

To pause loading or fixing processes using the command mode, use Ctrl+C to set the current operation into break mode. While in this mode, Tweaker will remain in its shell and wait for further instruction from you. The Tweaker shell will appear in the command prompt as `tweaker_shell>`, and you can simply resume the operation by entering `continue`. Contrary to the command mode, Tweaker's GUI contains no pause mechanism; however, most of the auto-fixing functions will open a Status pop-up window that allows you to stop a job anytime by selecting the **Stop** button. Although the **Stop** function works as a way to halt the fixing job, it is slightly different from the pausing function invoked by Ctrl+C.

Feedback

To switch between the GUI and Command modes, Tweaker provides `guion` and `guioff` commands:

- `guion` – opens Tweaker GUI from command mode
- `guioff` – invokes Tweaker shell in command mode; disables Tweaker GUI from opening

Input Data Preparation

To perform a high quality ECO job, Tweaker requires numerous data inputs provided by you, which are classified into three groups:

1. Cell library files: Timing library (.lib), LEF
2. Design-related files: Gate level netlist, DEF, CPF or UPF
3. Timing-related files: SPEF, SDF, slack reports, TWF

These input files are introduced in the order of their respective loading processes as follows:

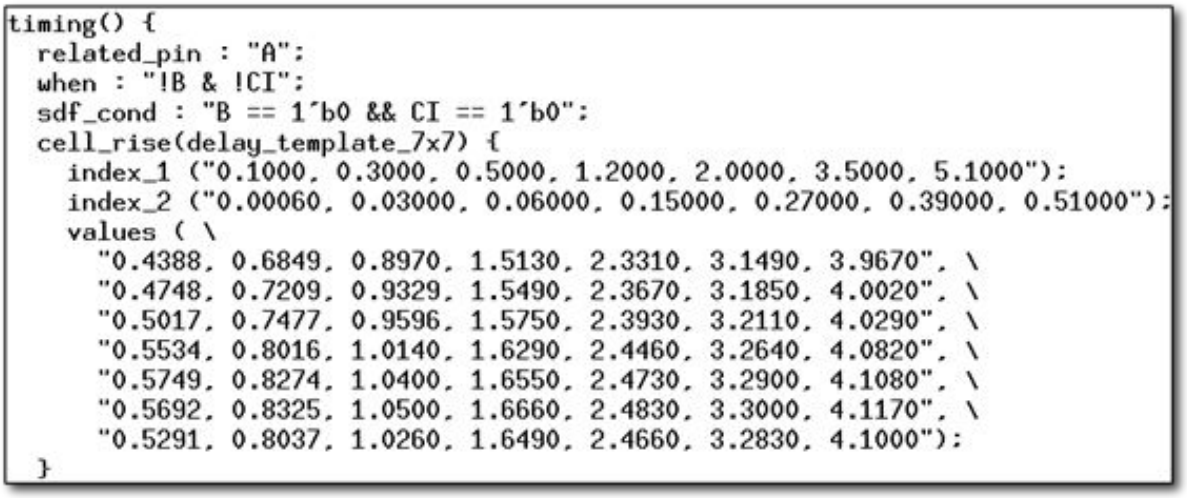
- The Timing Library File
- The LEF File
- The Netlist
- The DEF File
- The SPEF File and RC Tables
- The SDF File
- Slack Reports
- The TWF File
- CPF and UPF Files

The Timing Library File

Tweaker relies heavily on reading in the content of a timing library file (.lib). This file type defines each cell's basic attributes, such as the cell name, footprint, pin names, cell leakage, and the related transition and delay timing tables (some are shown in Figure 24). Tweaker supports NLDM (Non-Linear Delay Model) and CCS (Composite Current Source) as inputs.

Feedback

Figure 24 Example Portion of a .lib File



```
timing() {
  related_pin : "A";
  when : "!B & !CI";
  sdf_cond : "B == 1'b0 && CI == 1'b0";
  cell_rise(delay_template_7x7) {
    index_1 ("0.1000, 0.3000, 0.5000, 1.2000, 2.0000, 3.5000, 5.1000");
    index_2 ("0.00060, 0.03000, 0.06000, 0.15000, 0.27000, 0.39000, 0.51000");
    values ( \
      "0.4388, 0.6849, 0.8970, 1.5130, 2.3310, 3.1490, 3.9670", \
      "0.4748, 0.7209, 0.9329, 1.5490, 2.3670, 3.1850, 4.0020", \
      "0.5017, 0.7477, 0.9596, 1.5750, 2.3930, 3.2110, 4.0290", \
      "0.5534, 0.8016, 1.0140, 1.6290, 2.4460, 3.2640, 4.0820", \
      "0.5749, 0.8274, 1.0400, 1.6550, 2.4730, 3.2900, 4.1080", \
      "0.5692, 0.8325, 1.0500, 1.6660, 2.4830, 3.3000, 4.1170", \
      "0.5291, 0.8037, 1.0260, 1.6490, 2.4660, 3.2830, 4.1000");
  }
```

When reading in a timing library file to Tweaker, you need to specify its timing relation to STA operating conditions and library group names. Refer to the Library-Related Commands and Variables topic for instructions on how to read these files in correctly. In the case that the same cell is described in separate timing library files, Tweaker will adopt the “first-in first-apply” rule to regulate the loading priority on these identical cells.

The LEF File

Aside from the basic attributes, Tweaker needs to obtain each cell’s physical shape definition from the LEF files. This file format consists of each cell’s width and height, pin location, unit site, and metal layers which are used by the design. When a macro’s shape is irregular, Tweaker will define its shape based on a specific layer description called “OVERLAP”, particularly for macro cells like analog cells or IP blocks. Refer to the LEF-Related Commands and Variables topic for instructions to read in these files correctly.

When certain cells are defined in a LEF but not in the .lib file, Tweaker treats these cells as black boxes and ignores any timing updates performed on them. Similar to the .lib file property, identical cells defined within different LEF files will adopt the “first-in first-apply” rule.

LEF versions up to v5.8 is supported. For the best compatibility, use versions v5.6 or later.

The Netlist

Tweaker builds its database based on the netlist, and honors the netlist contents as golden even if differences exist among the input files. For example, if a cell is defined with a different cell type in each netlist and DEF files, Tweaker will choose to honor the contents

of the netlist over the DEF file's. Refer to the Netlist-Related Commands and Variables topic for instructions to read these files in correctly.

In the physical sense, Tweaker supports both hierarchical and flattened designs. And when an ECO job is done, Tweaker outputs the same amount of Verilog netlists as was provided at the input. For example, a hierarchical design in Tweaker with six Verilog netlists as input will also output six Verilog netlists. As an extra option, Tweaker allows you to forbid the output of netlists that have not been updated.

The DEF File

Through reading DEF files, Tweaker is aware of cell placement as well as physical blockages. For a physically hierarchical design, Tweaker reads multiple DEF's and allows you to do hierarchy-down and hierarchy-up transitions easily. Similar to the way of handling Verilog netlists, Tweaker outputs DEF's of all physical hierarchies after an ECO job is done. But instead of outputting a complete Verilog netlist, Tweaker outputs a partial ECO DEF for each block instead of a complete DEF of the entire design.

Tweaker reads not only placement information, but also blockage information in terms of soft blockage, hard blockage, placement blockage, or routing blockage. As a result, Tweaker will honor the same rules applied by APR during the ECO stage, and inhibit the APR tool from legalizing those ECO cells.

Additionally, Tweaker can read in detailed routing information, including clock routing, signal routing, and power straps. You are free to adjust the following settings, for what detailed routing information will be read into Tweaker:

- Congestion Awareness

  Enabling the congestion aware setting prompts Tweaker to completely avoid inserting ECO cells in detected hotspots on the design (this typically causes DRC issues).

- On-route Buffer Insertion

  By default, Tweaker uses Manhattan Distance attributes to execute buffer insertion. After the On-Route Buffering setting is enabled, Tweaker references the detail routing pattern input to perform repeater insertion.

- Detour Wire Protection

  Tweaker has this setting to collect all detour wires in the timing critical paths and provide a wire list – which includes APR routing attributes – from which each net will be rerouted.

- Route Guide Generation

  Through Tweaker, you can turn on this setting to generate a route guide containing the new ECO wire patterns for your APR tool routing reference.

Feedback

Tweaker supports DEF versions v5.8 and older. It is recommended to use v5.6 and later since only these versions support the shape of the design.

The SPEF File and RC Tables

The SPEF (or SPF) file format contains detailed descriptions of the RC network and total capacitance of each existing net in the design. These RC values play an important role in the delay calculation of the Tweaker tool, and must be provided.

As an ECO tool, Tweaker relies heavily on signoff RC values within the SPEF input to estimate the RC values of an “ECO-ed” net. With these calculated net parasitic values, Tweaker can also compute delay estimations of the ECO cells to be as close as possible to the real physical value. Take for example, when a cell is sized the tool assumes this only results in a minor change within the ECO nets. The updated RC value of the ECO nets are then be derived by slightly tuning the original signoff RC in the SPEF file.

When a completely new net is created in the ECO process, the Tweaker tool uses RC scaling in reference to the ECO-ed net’s original wire. However, if the original wire does not have RC data in the SPEF, the tool can also derive the new wire’s RC values based on the Unit RC Table (Figure 25):

Figure 25 Example Portion of a RC Table

```
RC CORNER : cworst

  TABLE
    MAX_FANOUT : 10
    LENGTH :     0.0000   100.0000   300.0000   500.0000
    RC :
          1   0.000134   0.000134   0.000162   0.000162
              0.959552   0.959552   0.450770   0.450770
              1.000000   1.000000   1.000000   1.000000
              1.000000   1.000000   1.000000   1.000000
          2   0.000149   0.000149   0.000162   0.000162
              0.533377   0.533377   0.292657   0.292657
              1.000000   1.000000   1.000000   1.000000
              1.000000   1.000000   1.000000   1.000000
          3   0.000161   0.000161   0.000159   0.000159
              0.382428   0.382428   0.237543   0.237543
              1.000000   1.000000   1.000000   1.000000
              1.000000   1.000000   1.000000   1.000000
```

The Tweaker tool generates a Unit RC Table automatically through every ECO run. The tool's derivation of this table is based on statistical analysis of all available sample nets in the SPEF input file, and the values are organized in respect to two parameters: net length and fan-out number. With this table, Tweaker has quick access to unit resistance and capacitance values, both of which were found under statistical bases. Since every

Feedback

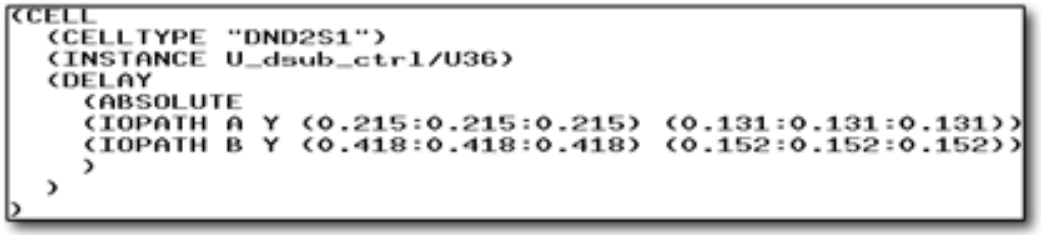
individual RC corner has its own characterized table, the amount of used RC corners within an ECO job determines the total number of generated RC tables.

The SDF File

To meet the timing prediction signoff quality, Tweaker reads in the SDF content–previously outputted from the signoff tool–as the golden delay and correlates these internally calculated delay values with the delay offset.

Figure 26 Example of Interconnect Delay in the SDF File

Figure 27 Example of Cell Delay Values in the SDF File



An SDF file contains records of interconnect delay, as well as cell delay, and these values are classified in terms of either rising or falling edge. Each rising or falling edge delay value consists of three corners: sdf_min, sdf_typ, and sdf_max. Since many functionalities of Tweaker-T1 involve STA and SDF, it is crucial that you thoroughly understand the SDF file's content and its relationship to the STA tools before executing timing fixes.

Each timing scenario built in Tweaker-T1 requires a corresponding SDF output from STA (or delay calculation) tools. Also, the significant digits of all delay value within the SDF file should be consistent with those in the slack reports; this is to prevent differences caused by floating points.

After the design and its timing data are both loaded, you must perform a consistency check before any timing fixes so that Tweaker can first verify that the delay records between each slack report and SDF are consistent. Any consistency check failures–path slack differences over 20 ps for a setup or hold path–must be addressed before fixing any timing violations. Further discussion about consistency checks can be found in the Consistency Check topic.

Feedback

Slack Reports

Since Tweaker-T1 normally builds each ECO Domain based on the slack reports, it is essential to follow the tips listed here to dump slack reports for better performance:

1. If the fixing procedure does not involve touching the clock, do not expand the full clock network. For example, doing buffer insertion to fix hold time does not require you to dump a slack report with `full_clock_expanded`; this reduces the amount of data processed by Tweaker-T1, therefore shortening runtime.
2. Adjust the `nworst` value to a reasonable number. The `nworst` values of too little value may lead to insufficient coverage of all violation domains, and possibly creating the need for additional ECO iterations. On the contrary, large `nworst` values could cause the file size to increase significantly, and require much more time for Tweaker to load and fix timing.
3. When the `nworst` value > 1, add the `-unique_pins` option onto the `report_timing` command to report more unique paths in Tweaker.
4. Input pins should be specified in slack reports for better file parsing.
5. Set “significant digits” to the same value as the one in the SDF file.

Tweaker comes with a useful set of sample Tcl scripts for timing fixes such as setup fix, hold fix, max transition fix, and so on. When dumping the STA tool’s slack reports, which are later used in Tweaker-T1, refer to either of the following directories for the example scripts:

$Tweaker_Install_Dir/dorado/etc/script/tcl/pt (for PrimeTime)

$Tweaker_Install_Dir/dorado/etc/script/tcl/tempus (for Tempus)

The TWF File

The timing window file (TWF) is a very powerful reference that enables Tweaker-T1 to safely and locally optimize timing within in the ECO Domain by monitoring timing changes at all timing scenarios. Moreover, TWF is required data input for Power ECO. Thus, through the TWF update system you can complete a fast and accurate whole chip ECO job.

TWF files can be dumped at the STA stage using the provided Tcl scripts. The scripts are located in the following directories, for PrimeTime and Tempus, respectively:

$Tweaker_Install_Dir/dorado/etc/scripts/tcl/pt/twfout.tcl

$Tweaker_Install_Dir/dorado/etc/scripts/tcl/tempus/twfout.tcl

Feedback

These files carry clock network components, design constraint information, minimum/maximum rise/fall slack/transition values at each pin, and path-based analysis (PBA) endpoint timing information.

All of the pin-based slack information enables Tweaker-T1 to do smart timing fixes while keeping the existing timing intact. In other words, the tool uses the whole-chip aspect of the TWF contents to assure that the locally-based timing optimization is achievable and reliable. Tweaker-T1's innovative TWF update system constantly updates the timing graph with each timing fix, so rebuilding the timing graph is unnecessary and thus enables handling of many scenarios in a fast manner.

Based on various operations, you have the ability to control the options to dump each operation's corresponding TWF file types. There are two types of TWF files: normal and PBA. In most cases, you will select the normal timing fix setting first. On the other hand, selecting the PBA mode will dump a realistic slack value, but with the cost of a significant amount of extra runtime. The reason for this is that if PBA mode is activated within the STA environment, the slack of every pin that's involved with the PBA analysis will be recalculated after the graph-based analysis (GBA) process and then dumped to the TWF file.

CPF and UPF Files

CPF and UPF are the two commonly used file formats for designs involving multiple power domains. Tweaker supports both file formats in terms of defining power domains, as well as power domain-related cells such as always-on buffers, isolation cells, level shift cells, and power switch cells. In addition, Tweaker matches the logical power domains defined in the CPF and UPF files to the physical power domains defined by either the `create_voltage_area` command or the DEF file.

Figure 28 Segment of CPF File Describing Power Cells

```
define_level_shifter_cell \
   -cells { M_LSZ1M* } \
   -input_voltage_range 1.2:2.8:0.1 \
   -output_voltage_range 1.2:2.8:0.1 \
   -input_power_pin DVDDI \
   -output_power_pin DVDDO \
   -ground DVSS \
   -direction bidir
```

Feedback

Figure 29 Segment of CPF File Describing Logical Power Domain

```
create_power_domain \
   -name v_md2gsys \
   -instances { md2gsys_wrapper } \
   -shutoff_condition { !md2gsys_wrapper/md2gsys_pwr_on }
```

Tweaker-T1 supports power domain-aware operations including ECO Place as well as jobs with buffer insertion, such as hold time fix and maximum transition fix.

4

Design Loading Structure

Before executing timing fixes or ECO operations, data must be input to Tweaker in the recommended sequence, to construct the database correctly:

Part 1 – Cell Definition-related Inputs

Part 2 – Design-related inputs

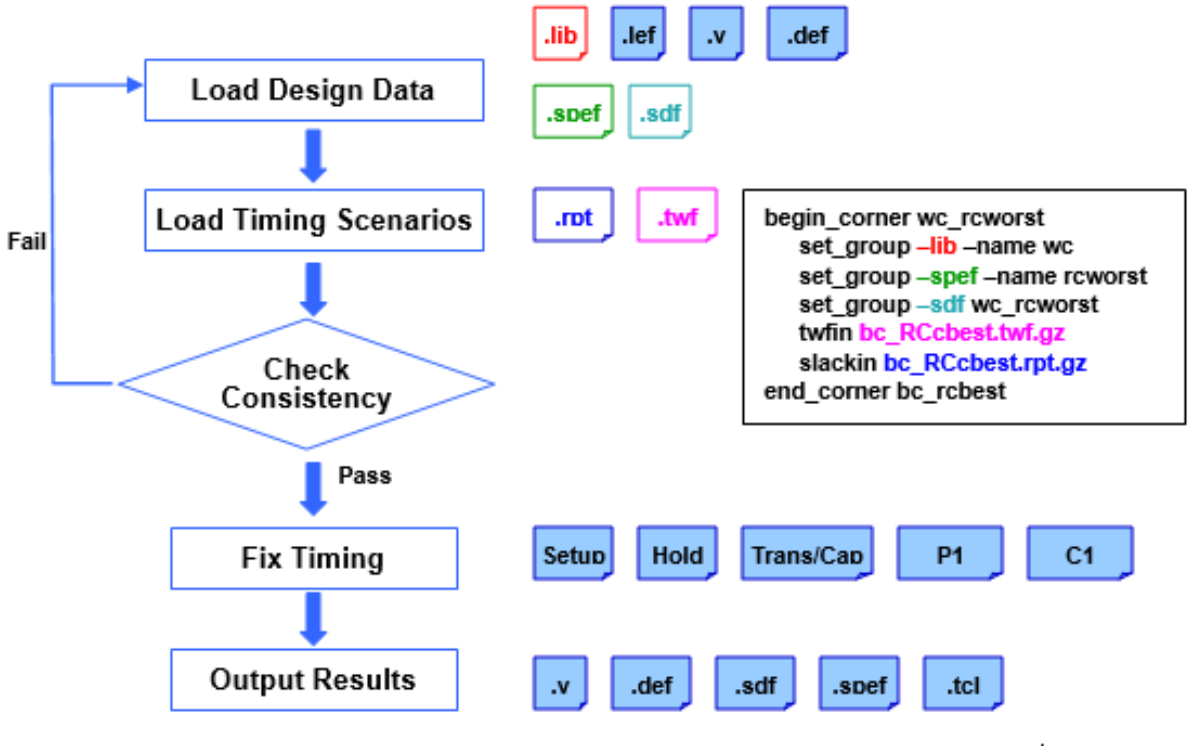
Part 3 – STA-related inputs

Most ECO operations inside the Tweaker ECO Platform share the same design loading process; for instance, a timing fix job requires all steps of the following flow diagram. You can also flexibly edit a Tcl file, and control the loading process according to each individual ECO purpose. An example of this would be a simple loading process that only consists of reading in DEF. Since each step of the loading process requires different input data, Tweaker offers a convenient “search path” feature that, when used with the input’s file name, Tweaker can auto-match and use the data.

Complete Loading Structure of Regular Timing Fix (Figure 30):

1. Load Design Data
2. Load Timing Scenarios
3. Check Consistency
4. Fix Timing
5. Output Results

Figure 30 Loading Structure of Regular Timing Fix



Design Loading: Tweaker Commands for Data Input

The following topics cover the commands for data input:

- Library-Related Commands and Variables
- LEF-Related Commands and Variables
- Netlist-Related Commands and Variables
- DEF-Related Commands and Variables
- CPF- and UPF-Related Commands
- SPEF-Related Commands
- SDF-Related Commands and Variables
- Hierarchical Designs

Feedback

Library-Related Commands and Variables

To read in timing library files (.lib), use the `libin` command. Specify this command's options carefully, since some of them can be linked to, and might therefore affect, STA operating condition settings. The `libin` command supports g-zipped files. See Table 1 for `libin` command usage.

Table 1 The libin Command Syntax

Syntax	`libin -timing_type [best | typ | worst] -name LIB_GROUP_NAME file_name`
Example	`libin -timing_type best worst -name wc A.lib`
Options	
`-timing_type`	You must specify SDF columns linked by the current timing library file. Reference **(1)** for a detailed explanation.
`-name`	You must give a .lib group nametag to be referenced in scenarios. Reference **(2)** for a detailed explanation.

(1) Timing type (sdf_min : sdf_typ : sdf_max)

If both sdf_min and sdf_max columns derive the delay value by linking the same A_lt.lib timing library file, then use the command format in Example 3 to save memory usage.

Example 3 libin –timing_type Delay Derivation Based on the Same Library

```
libin -timing_type best worst -name lt A_lt.lib
```

However, if the sdf_min and sdf_max columns derive the delay value by linking two different timing library files A_bc.lib and A_wc.lib respectively, use the `libin -timing_type` command by following the format in Example 4.

Example 4 The libin –timing_type Delay Derivation Based on Different Libraries

```
libin -timing_type best -name bc A_bc.lib
libin -timing_type worst -name wc A_wc.lib
```

(2) Group Nametag

Each established group nametag contains a collection of timing library files, usually from the same library group. See Example 5 for group nametag format.

Feedback

Example 5 The libin –timing_type Group Nametags

```
libin -timing_type best worst -name wc A_wc.lib
libin -timing_type best worst -name wc macro_wc.lib
libin -timing_type best worst -name wc io_wc.lib
```

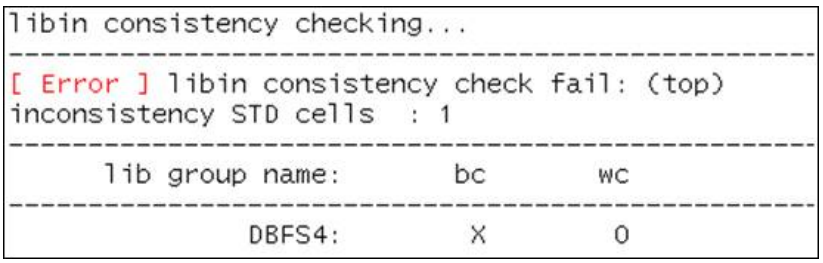
In Example 6, the A_wc.lib, macro_wc.lib, and io_wc.lib timing library files will be in the same group called “wc”. For timing scenarios that are established later on, you can simply refer back to an existing nametag and conveniently specify its corresponding timing libraries.

Example 6 Contents of a Scenario

```
begin_corner wc_RCtypical_norm
set_group -lib -name wc
end_corner wc_RCtypical_norm
```

In addition to checking whether slack values are consistent with signoff data’s values, the `slkdc -check_slack_consistency` command also verifies if every library cell within a library group is properly defined. In other words, if a specific cell is defined in one library group but missing in another one, Tweaker shows an error message similar to Figure 31. You should make the appropriate changes to recovery consistency.

Figure 31 Example of a Consistency Error Message: DBFS4 is defined at “wc” but not in “bc”



Feedback

LEF-Related Commands and Variables

The Tweaker command to read the LEF file format is `lefin`, and it supports g-zipped files. See Table 2 for command usage.

Table 2 The lefin Command Syntax

Syntax	`lefin [-tech] file_name`
Example	`lefin -cell_type std A.lef`
Options	
`-tech`	Read in the technology portion of a LEF file as needed. Often used to read in the unit RC of metal layers from a standard_cell.lef file.

Sometimes to avoid module/cell conflicts, you must ignore certain definitions within a LEF file. This is because listing the same object under different class definitions will cause errors during loading processes. One example of a conflict is when a module (sub-design) is already defined in the netlist, but is also listed as class “cell” within the LEF file. To avoid this definition conflict on the same object, Tweaker has a `library_cells_to_be_ignored` variable to flag unwanted cell definitions before implementing the `lefin` command.

Example 7 Ignoring CellA, CellB, and CellC Definitions in Top.lef

```
set library_cells_to_be_ignored {CellA CellB CellC}
lefin Top.lef
```

To verify a cell definition’s validity within a LEF file, Tweaker also provides a `check_lef_quality` command to automatically check a specific cell for its required physical attributes. See Table 3 for the command’s usage.

Table 3 The check_lef_quality Command Syntax

Syntax	`check_lef_quality -cell Pattern`
	`[-ignore Pattern]`
	`[-std_cells_only]`

Example	`check_lef_quality -cell * -ignore ABC* -std_cells_only`
Options	
`-cell`	You must specify a target cell pattern. Wildcard is supported. The asterisk symbol (*) represents all available library cells.
`-ignore`	Specifies the cells to be ignored. Wildcard is supported.

Feedback

Table 3 The check_lef_quality Command Syntax (Continued)

`-std_cells_only`	If this option is used, only standard cells are checked.

Example 8 All Standard Cells Except ABC will be Checked for Their Physical Attributes*

```
lefin -cell_type std Top.lef
# or skip the -cell_type option
# lefin Top.lef
check_lef_quality -cell * -ignore ABC* -std_cells_only
```

Netlist-Related Commands and Variables

To read in netlist files, use the `verilogin` command with the syntax shown in Table 4. This command also supports g-zipped netlist files. After all netlists are read into Tweaker, you must define the top module using `set_verilog_top_cell Top_module`.

Table 4 The verilogin Command Syntax

Syntax	`verilogin "`*`netlist_file_name`*`"`

Example	`verilogin "Top.v"`
	`verilogin "Sub.v"`
	`set_verilog_top_cell top`

DEF-Related Commands and Variables

The `defin` command reads DEF files, and g-zipped DEF files, into Tweaker using the syntax shown in Table 5. For a (physically) hierarchical design, each design may require an independent DEF file so you must read each respective DEF file one by one. Tweaker also supports duplicated sub-designs in which an individual set of DEF and a netlist files is responsible for multiple duplicated designs.

Table 5 The defin Command Syntax

Syntax	`defin`
	`[-place]`
	`[-route]`
	`[-power]`
	`[-power_strap_as_blockage]`
	`def_file`

Feedback

Table 5 The defin Command Syntax (Continued)

Example	`defin Top.def`
	`defin Sub1.def`
	`defin Sub2.def`

Options	
`-place`	Reads placement information within the DEF file
`-route`	Reads placement + routing information within the DEF file
`-power`	Reads physical power domain definition within the DEF file
`-power_strap_as_blockage`	Considers any M2 power straps within the DEF file as placement blockages during ECO optimization

In Tweaker, DEF files of version 5.8 or later are suitable for correctly defining the shape of a design. To check for the valid DEF file version, use the `min_required_def_version` variable to specify the minimum DEF version requirement. Any DEF files that do not meet the minimum required version results in an error.

Example 9 Violating the Minimum Required DEF Version Results in Error

Figure 32

```
set min_required_def_version 5.6
defin TOP.def
```

Figure 32 Error Message for DEF Version. Current Version is Older Than Required Version

CPF- and UPF-Related Commands

The command for reading a CPF (or UPF) file is `cpfin` (or `upfin`), and its usage is listed in Table 6. For hierarchical designs, each sub-design might require an independent CPF (or UPF) file to define each logical power domain.

Table 6 The cpfin Command Syntax

Syntax	`cpfin` *`cpf_file_name`* or `upfin` *`upf_file_name`*

Feedback

Table 6 The cpfin Command Syntax (Continued)

Example	`cpfin Top.cpfor  upfin Top.upf`
	`cpfin Sub1.cpfor  upfin Sub1.upf`
	`cpfin Sub2.cpfor  upfin Sub2.upf`

SPEF-Related Commands

The command for reading parasitic files is `spefin` (or `spfin`). This command's usage is specified in Table 7, and it supports g-zipped SPEF or SPF file formats.

Table 7 The spefin Command Syntax

Syntax	`spefin | spfin -design` *`design`* `-name RC_GROUP` *`file_name`*

Example	`spefin -design top -name RCtypical Top.spef.gz`
	`spefin -design Sub1 -name RCtypical Sub1.spef.gz`
	`spefin -design Sub2 -name RCtypical Sub2.spef.gz`

Options	
`-design`	Parasitic file input's corresponding hierarchical design
`-name`	User-defined nametag that is referenced in timing scenarios. Reference **(1)** for detailed explanation.

(1) The nametag group contains a collection of SPEF files, which are most likely under the same RC corner.

Example 10 Reading in Hierarchical Design SPEF Information Under the Same Collection "RCtypical"

```
spefin -design top -name RCtypical Top.spef.gz
spefin -design Sub1 -name RCtypical Sub1.spef.gz
spefin -design Sub2 -name RCtypical Sub2.spef.gz
```

When all timing scenarios are later established, you can specify a corresponding set of SPEF files by using its nametag (see Example 11).

Example 11 Specifying the SPEF File Collection Using nametag

```
begin_corner wc_RCtypical_norm
set_group -spef -name RCtypical
end_corner wc_RCtypical_norm
```

Feedback

SDF-Related Commands and Variables

The command to read in SDF file formats is `sdfin`. This command's usage is listed in Table 8, and it also supports g-zipped files.

Table 8 The sdfin Command Syntax

Syntax	`sdfin -name SDF_GROUP` *`file_name`*
Example	`sdfin -name wc_RCtypical_norm wc_RCtypical_norm.sdf.gz`
Options	
`-name`	User-defined nametag that is referenced in timing scenarios. Reference **(1)** for detailed explanation.

(1) Notice in Example 12 that each timing scenario requires a corresponding SDF file, except when building an RC table.

Example 12 Building a Timing Scenario

```
sdfin -name wc_RCtypical_norm wc_RCtypical_norm.sdf.gz
....
begin_corner wc_RCtypical_norm
set_group -sdf -name wc_RCtypical_norm
slackin ... -analysis_type \ on_chip_variation ...
end_corner wc_RCtypical_norm
```

Hierarchical Designs

As previously introduced, Tweaker-T1 is capable of fixing timing on hierarchical designs, which can be further classified into five types. Each of the design types are described as follows.

Type 1: Uniquified Subdesigns

- All designs were uniquified by synthesis or APR tools
- Each subdesign has an independent netlist, DEF, or SPEF file
- Tweaker reads each uniquified designs' netlist, DEF, or SPEF file

Example 13 A top Design Including Two Uniquified Subdesigns (sub_design_1.v and sub_design_2.v)

```
verilogin "top.v"
verilogin "sub_design_1.v"
verilogin "sub_design_2.v"
```

```
defin "top.def"
defin "sub_design_1.def"
defin "sub_design_2.def"

spefin -design top -name cworst "top.spef.gz"
spefin -design sub_design_1 -name cworst "sub_design_1.spef.gz"
spefin -design sub_design_2 -name cworst "sub_design_2.spef.gz"
```

Type 2: Non-Uniquified Subdesigns

- Some subdesigns are duplicated, and these duplicates are non-uniquified
- The duplicated design shares an identical netlist, DEF, or SPEF file

Example 14 A top Design That Includes Two Uniquified Subdesigns (sub_design_1.v and sub_design_2.v) and Two Separate Instances of the Module "sub_design_dup"

```
verilogin "top.v"
verilogin "sub_design_1.v"
verilogin "sub_design_2.v"
verilogin "sub_design_dup.v"
# only one netlist is required for duplicate designs

defin "top.def"
defin "sub_design_1.def"
defin "sub_design_2.def"
defin "sub_design_dup.def"
# only one def is required for duplicate designs

spefin -design -name cworst "top top.spef.gz"
spefin -design sub_design_1 -name cworst "sub_design_1.spef.gz"
spefin -design sub_design_2 -name cworst "sub_design_2.spef.gz"
spefin -design sub_design_dup -name cworst "sub_design_dup.spef.gz"
# only one spef is required for duplicate designs
```

*Any fixes made on *any* of the duplicates are replicated and watched across *all* of the duplicate modules.

Type 3: Subdesigns Modeled as Black Box

- A subdesign is treated as a black box or IP block, and has corresponding LEF file
- By reading the LEF file of the sub-design, Tweaker is aware of its pin locations and physical shape
- Timing within, to, or from the black box is not updated since no .lib file is read in

Example 15 A top Design Includes Two Uniquified Sub-Designs and a sub_design_3 Black Box

```
lefin "sub_design3.lef"
verilogin "top.v"
```

```
verilogin "sub_design_1.v"
verilogin "sub_design_2.v"

defin "top.def"
defin "sub_design_1.def"
defin "sub_design_2.def"

spefin -design top -name cworst "top.spef.gz"
spefin -design sub_design_1 -name cworst "sub_design_1.spef.gz"
spefin -design sub_design_2 -name cworst "sub_design_2.spef.gz"
```

Type 4: Subdesigns Modeled as Macro Cell

- A subdesign is treated as a macro cell and has corresponding .lib and LEF files
- By reading its LEF file, Tweaker is aware of its pin locations and physical shape
- By reading its .lib file, Tweaker can update timing to or from the subdesign

Example 16 A top Design Includes Two Uniquified Subdesigns and sub_design_3 as a Macro Cell

```
libin -timing_type best worst -name worst "sub_design_3.lib"
lefin "sub_design3.lef"

verilogin "top.v"
verilogin "sub_design_1.v"
verilogin "sub_design_2.v"

defin "top.def"
defin "sub_design_1.def"
defin "sub_design_2.def"

spefin -design top -name cworst "top.spef.gz"
spefin -design sub_design_1 -name cworst "sub_design_1.spef.gz"
spefin -design sub_design_2 -name cworst "sub_design_2.spef.gz"
```

Type 5: Subdesigns Modeled by ILM

- The netlist or SPEF file of a subdesign modeled by ILM only contains boundary logic circuits extracted by 3rd party tools
- To read the DEF file of an ILM subdesign, use the `-lim` option to assign all instances outside the ILM module as placement blockages; this is for Tweaker to exactly identify the free space inside the subdesign during optimization.

Example 17 A top Design Includes Two Uniquified Subdesigns and sub_design_3, Modeled by ILM

```
verilogin "top.v"
verilogin "sub_design_1.v"
verilogin "sub_design_2.v"
verilogin "sub_design_3_ilm.v"
```

Feedback

```
defin "top.def"
defin "sub_design_1.def"
defin "sub_design_2.def"
defin -ilm "sub_design_3.def"

spefin -design top -name cworst "top.spef.gz"
spefin -design sub_design_1 -name cworst "sub_design_1.spef.gz"
spefin -design sub_design_2 -name cworst "sub_design_2.spef.gz"
spefin -design sub_design_3 -name cworst "sub_design_3_lim.spef.gz"
```

With Tweaker's five hierarchical design flows, you can deliver post-ECO data. In short, the general rule for delivering post-ECO data is that each block, which possessed its own netlist/DEF file in Tweaker, will have its ECO data outputted individually. For instance, Tweaker will output each ECO netlist, ECO DEF, and ECO Tcl block-by-clock; this facilitates a block owner to ECO route in parallel through an APR tool.

Design Loading: Commands for the MMMC Structure

The following topics cover the commands for the MMMC structure:

- To Match STA Environment
- Contents of a Timing Scenario
- Additional Useful Commands for a Timing Scenario
- Slack Report-Related Commands and Variables
- TWF-Related Commands and Variables
- Default Scenario
- Automatically Reload Data Required by ECO Domain

To Match STA Environment

Tweaker-T1's capabilities are heavily linked with those of STA tools. Under this relationship, Tweaker's database is designed to match STA environments; however, Tweaker's database covers all scenarios at the same time. To define each scenario, start and end its corner definition with the `begin_corner` and `end_corner` commands, respectively. In between those two commands, you supply the corresponding library group, RC group, SDF group, slack reports, and most importantly, the TWF files. In using each set of `begin_corner` and `end_corner` scenario definitions, Tweaker-T1 builds up the MMMC database that is essential to fix timing.

Aside from each of the mentioned files specified within each scenario, Tweaker-T1 also readily incorporates various timing analysis types into each scenario, such as

single, bc_wc, and on_chip_variation. In most recent cases, timing analyses reside in the on_chip_variation category, which can be further classified under Global OCV or Advanced OCV (AOCV).

The following subsections will focus on how to set up timing scenarios using Tweaker-T1 and how to correlate the scenarios to the STA environment.

Contents of a Timing Scenario

In Example 18, notice that each scenario matches an STA environment in terms of specifying each corresponding library group name, RC group name, SDF group name, TWF file, and slack reports. Using the `-name` option for the `set_group` command will prompt Tweaker-T1 to find all files that were previously defined with identical nametags. Therefore, the `-name` option is convenient for you to group each set of related files when describing a timing scenario.

Example 18 A Basic Scenario

```
# Beginning of a scenario
begin_corner scenario_name
set_group -lib -name library_group
set_group -spef -name RC_group
set_group -sdf -name SDF_group
twfin -analysis_type on_chip_variation TWF_file
slackin -mode func -analysis_type on_chip_variation report_file
end_corner scenario_name
# End of a scenario
```

After the basic STA data is listed properly in each corner definition block, follow with the `slackin` and `twfin` commands, to read in both slack reports and TWF files. Addition usage details for these two commands will be available later in this chapter.

Example 19 Power ECO Scenario Definitions Only Require TWF Files (No Slack Report)

```
# Beginning of a scenario for "Power ECO"
begin_corner scenario_name
set_group -lib -name library_group
set_group -spef -name RC_group
set_group -sdf -name SDF_group
twfin -analysis_type on_chip_variation TWF_file
end_corner scenario_name
# End of a scenario for "Power ECO"
```

Example 20 Contents of Each Scenario for “ECO Place” Operation

```
# Beginning of a scenario for "ECO Place"
begin_corner scenario_name
set_group -lib -name library_group
set_group -spef -name RC_group
set_group -sdf -name SDF_group
```

Feedback

```
end_corner scenario_name
# End of a scenario for "ECO Place"
```

Additional Useful Commands for a Timing Scenario

Besides the commands to specify basic STA data, there are a few commands and options that help to construct detailed settings within a scenario. The first useful command, `set_timing_derate`, is used to set the global OCV derate value, and when implemented Tweaker-T1 simulates the STA environment based on this global derating factor. Additional command usage is listed in Table 9, and refer to Example 21 for a sample scenario with OCV derating. You must make sure to set derating values in advance – before the `slackin` and `twfin` commands – and also reset these values before the next scenario. Tweaker-T1 also supports LOCV/AOCV by reading through the “Derate” column of the imported slack reports that include the ECO path domain. The following is a list of ways to set timing derate values, arranged from highest to lowest priority.

1st Priority: Get derating values from the "Derate" column of STA slack reports

2nd Priority: Get derating values from the header of each path in the STA slack report

3rd Priority: Set derating values using the `set_timing_derate` command

The second useful command usage is enabling or disabling the `-name` or `-inst` options after `set_group -lib` (see Example 22 for application of this option). These options allow Tweaker to support Multi-VDD timing libraries for different modules within the same timing scenario (much like `link_path_per_instance` from PrimeTime). With the following command usage, you can easily construct and accurately match each scenario to the STA environment.

Table 9 The set_timing_derate Command Syntax

Syntax	`set_timing_derate`
	`[-early | -late]`
	`[-clock | -data]`
	`[-cell_delay | -cell_check | -net_delay]` *value*
	`[get_lib_cells` *cell_pattern*`]`

Feedback

Table 9 The set_timing_derate Command Syntax (Continued)

Example	Pair 1:
	`set_timing_derate -early 0.95`
	`set_timing_derate -late 1.0`
	Pair 2:
	`set_timing_derate -early 1.0`
	`set_timing_derate -late 1.05 -data -cell_delay`
	Pair 3:
	`set_timing_derate -early 0.95 -clock`
	`set_timing_derate -late 1.05 -clock`
	`set_timing_derate -early 0.95 [ get_lib_cells *PAD* ]`

Options	
`-early` | `-late`	For launch path (launch clock path + launch data path) or capture path (capture clock path): `-early`: launch path `-late`: capture path For setup violations: `-early`: capture path `-late`: launch path
`-clock` | `-data`	Specifies which path segment to derate, since the launch path contains both clock and data paths. Otherwise, both segments are derated.
`-cell_delay` | `-cell_check` | `-net_delay`	Specifies either cell delay, net delay, or cell timing check to derate on a path's segment. Otherwise, all factors are derated on the segment.
`get_lib_cells` *`cell_pattern`*	Gets a collection of pattern matched cell names. If this command is used, the derate value is only applied to these cells. Each cell is exclusive to a single derate value within each timing scenario.

Example 21 Scenario With OCV Derating

```
# Beginning of a scenario with OCV derating applied
begin_corner scenario_name
set_timing_derate -early 0.95
set_timing_derate -late 1.0
twfin -analysis_type on_chip_variation TWF_file
slackin -mode func -analysis_type on_chip_variation report_file
# reset back to default value
set_timing_derate -early 1.0
set_timing_derate -late 1.0
end_corner scenario_name
# End of a scenario with OCV derating applied
```

Feedback

Example 22 Applying Multi-VDD Timing Libraries for Different Modules in the Same Timing Scenario

```
libin -timing_type best worst -name bc A_bc.lib
libin -timing_type best worst -name bc_11v A_11vbc.lib

...

# Beginning of a scenario with MVDD assigned to module 11vModule
begin_corner scenario_name
set_group -lib -name bc
set_group -lib -name bc_11v -inst 11vModule
end_corner scenario_name
# End of a scenario with MVDD assigned to module 11vModule
```

Slack Report-Related Commands and Variables

The command for reading in slack reports is `slackin`. When a single report is read into Tweaker-T1, you must specify its timing analysis type. Therefore, an incorrect timing analysis type specification will be accompanied with a “consistency check failure” (to be introduced later). The command usage is listed in Table 10.

Table 10 The slackin Command Syntax

Syntax	`slackin -mode` *`mode`*
	`[-analysis_type single | bc_wc | on_chip_variation]`
	`[-type sdf_max | sdf_typ | sdf_min]`
	`[-si` *`si_corner`*`]`
	`[-si_analysis_type single | bc_wc | on_chip_variation]`
	`[-si_type sdf_max | sdf_typ | sdf_min]`
	`[-if | -inv_if]`
	`[-io | -ff]`
	`[-slack_range from to]`
	`[-design_list` *`design_name`*`]`
	`"`*`file_name`*`"`

Example	`slackin -mode func -type sdf_max "constraint.rpt"`
	`slackin -mode scan -analysis_type on_chip_variation "hold.rpt"`
	`slackin -mode func -analysis_type on_chip_variation -slack_range -3 -0.1 on_chip_variation "setup.rpt" -design_list top`

Options	
`-mode`	Specifies the slack report’s mode of operation within a scenario. This is only a nametag used to build the timing slack “scoreboard”, and is irrelevant to SDC.

Feedback

Table 10 The slackin Command Syntax (Continued)

`-analysis_type`	Specifies the timing analysis type (commonly “bc_wc” or “on_chip_variation”) `-analysis_type` *`bc_wc`*: (1) sdf_min column: annotates delay on both launch and capture path of hold violation path (2) sdf_max column: annotates delay on both launch path and capture path of setup violation path `-analysis_type` *`on_chip_variation`*: With on-chip variation, sdf_min is applied at the launch path and sdf_max is applied at the capture path of the hold violation path, and vice versa for setup violation path.
`-type`	Enforce Tweaker-T1 to adopt one of the three SDF delay columns: sdf_max, sdf_typ or sdf_min
`-if` | `-inv_if`	*Filters the report by `-if` (interface) or `-inv_if` (non-interface) timing paths
`-ff` | `-io`	*Filters the report by `-ff` (register to register) or `-io` (I/O) timing paths
`-slack_range`	*Filters the report by timing paths within specified slack range (in ns) `-slack_range lower_bound upper_bound`
`-design_list`	*Filters the report by certain hierarchical design timing paths

*Most frequently used options for the `slackin` command.

TWF-Related Commands and Variables

The command for reading in a TWF file is `twfin`. Like the `slackin` command, timing analysis type must be specified for `twfin` to identify which SDF delay column will be used to update TWF. Command usage is listed in Table 11.

Table 11 The twfin Command Syntax

Syntax	`twfin -analysis_type`
	`[single | bc_wc | on_chip_variation]`
	`[-min | -max]`
	`"file_name"`

Example	`twfin -analysis_type on_chip_variation "top_bc_cbest.twf.gz"`
Options	

Feedback

Table 11 The twfin Command Syntax (Continued)

`-analysis_type`	Specifies the timing analysis type (commonly “bc_wc” or “on_chip_variation”) `-analysis_type` *`bc_wc`*: (1) sdf_min column: used to annotate delay on both launch and capture path of hold violation path. (2) sdf_max column: used to annotate delay on both launch path and capture path of setup violation path. `-analysis_type` *`on_chip_variation`*: With on-chip variation, sdf_min is applied at the launch path and sdf_max is applied at the capture path of the hold violation path, and vice versa for setup violation path.
`-min` | `-max`	Reads in either only minimum or maximam slack from the same TWF; otherwise, both minimum and maximum slack are read in.

Default Scenario

If no scenario is established (that is, no “begin_corner” and “end_corner” description within the Tcl scripts), then Tweaker creates a default scenario. In Tweaker’s default scenario, the initial group of timing libraries and RC files are used. Although Tweaker provides a default scenario, it is strongly recommended that you to establish a scenario whenever possible.

Automatically Reload Data Required by ECO Domain

After all scenarios are specified, Tweaker-T1 will automatically reload the ECO Domain’s essential data, which is defined by either slack reports or results of ECO Compare. This process entails first reloading cell/interconnect delay within SDF files, then net RC from SPEF/SPF files, and finally pin’s slack information from TWF files. As a result, the reload processing time will depend on the size of the ECO Domain and number of built-up timing scenarios.

Feedback

Figure 33 Example of Reloading SDF Log Message

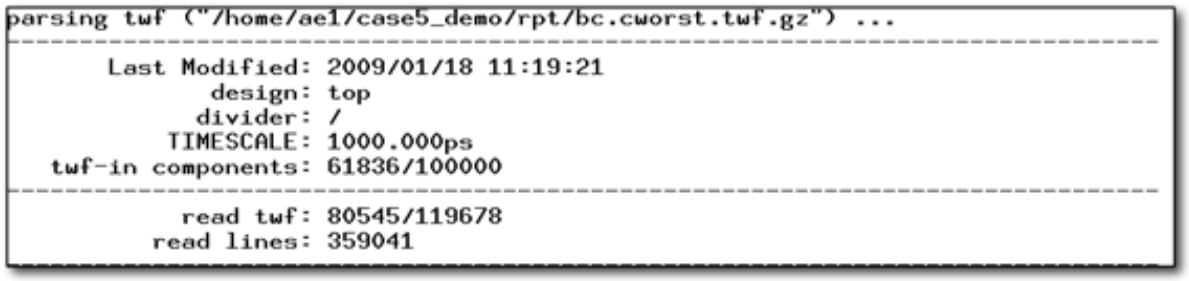
```
Slack initialize timing table of the new instances(10276)...
[ TWF ] Create path domain
Build reload inst(s)...
--------------------------------------------------------------------------------
             inst(s): 10271
              pin(s): 23572
                time: 0.106 sec
Build reload net(s)...
--------------------------------------------------------------------------------
         root net(s): 6465
        local net(s): 5068
                time: 0.068 sec
reloading sdf for new instances(10271)...
parsing sdf ( -name bc_cworst "/home/ae1/case5_demo/sdf/bc.cworst.sdf") ...
--------------------------------------------------------------------------------
       Last Modified: 2009/01/18 11:10:58
     sdf hier prefix:
              design: top
             divider: /
           TIMESCALE: 1.000ns
--------------------------------------------------------------------------------
   sdf-in components: 10271/24535
 sdf-in interconnect: 17096/55863
```

Figure 34 Example of Reloading SPEF/SPF Log Message

```
reloading spf for new nets(6465)...
parsing spf ( -name cworst "/home/ae1/case5_demo/spf/top.cworst.spf.gz") ...
--------------------------------------------------------------------------------
       Last Modified: 2008/10/14 14:44:49
              design: top
             divider: /
           I/O time: 0.092 sec
--------------------------------------------------------------------------------
   spf-in components: 2258/9947
--------------------------------------------------------------------------------
            run time: 0.328 sec
--------------------------------------------------------------------------------
```

Feedback

Figure 35 Example of Reloading TWF Log Message



Consistency Check

After the required data is reloaded into the ECO Domain, you should launch a consistency check between the slack values from signoff reports and the slack values that Tweaker-T1 recalculated after honoring SDF delay. If the consistency check fails or there is a large offset between each set of slack values, then some data may have been collected or configured incorrectly during the loading process. At this point, it would be risky to continue onto the remaining ECO jobs, since the consistency check failure may give garbage-in-garbage-out results.

Notice that Tweaker-T1 uses its own engine, along with timing libraries and SPEF input, to calculate the cell/interconnect delay. In addition, Tweaker-T1 matches signoff data by the following:

1. Referencing delay values from SDF as golden
2. Keeping an “offset factor” on each timing arc in the ECO Domain, to compensate for discrepancies

Therefore, the delay seen by Tweaker-T1 (db timing + offset) before timing fixing is exactly the same as those in SDF files. This is how Tweaker-T1, as an ECO tool, correlates to the signoff data and, therefore, produces reliable timing fix results.

The following topics are covered:

- Consistency Check Success and Failure Messages
- Inconsistency or Large Offset: Possible Causes
- How to Verify Inconsistency
- Inconsistency Work-Around

Feedback

Consistency Check Success and Failure Messages

When the difference between the slack values (values from the signoff slack report versus values that T1 recalculates) exceeds a certain threshold – defined by the `slk_consistency_check_threshold` variable – Tweaker will report an inconsistency error. The default for `slk_consistency_check_threshold` is 20 ps.

Figure 36, Figure 37, and Figure 38 show examples of a successful, a failed consistency check, and the most inconsistent path log, respectively. These log snippets contain some noticeable values such as average offset value, max offset value, and those values' corresponding pins. This part of the log will be discussed further in later sessions.

Figure 36 Successful Consistency Check Log

```
cell delay offset checking...
-------------------------------------------------------------------------------
corner name : bc_cworst
      best : avg offset(6.74671), max offset(1045.44958), pin( U347/Y )
     worst : avg offset(4.48056), max offset(1047.48645), pin( U347/Y )
-------------------------------------------------------------------------------
corner name : wc_cworst
      best : avg offset(11.83604), max offset(2176.44507), pin( U347/Y )
     worst : avg offset(7.50763), max offset(2179.97681), pin( U347/Y )
-------------------------------------------------------------------------------
Pass slack consistency check. All (slack_in_report - slack_re_calculated) < +/-20ps
```

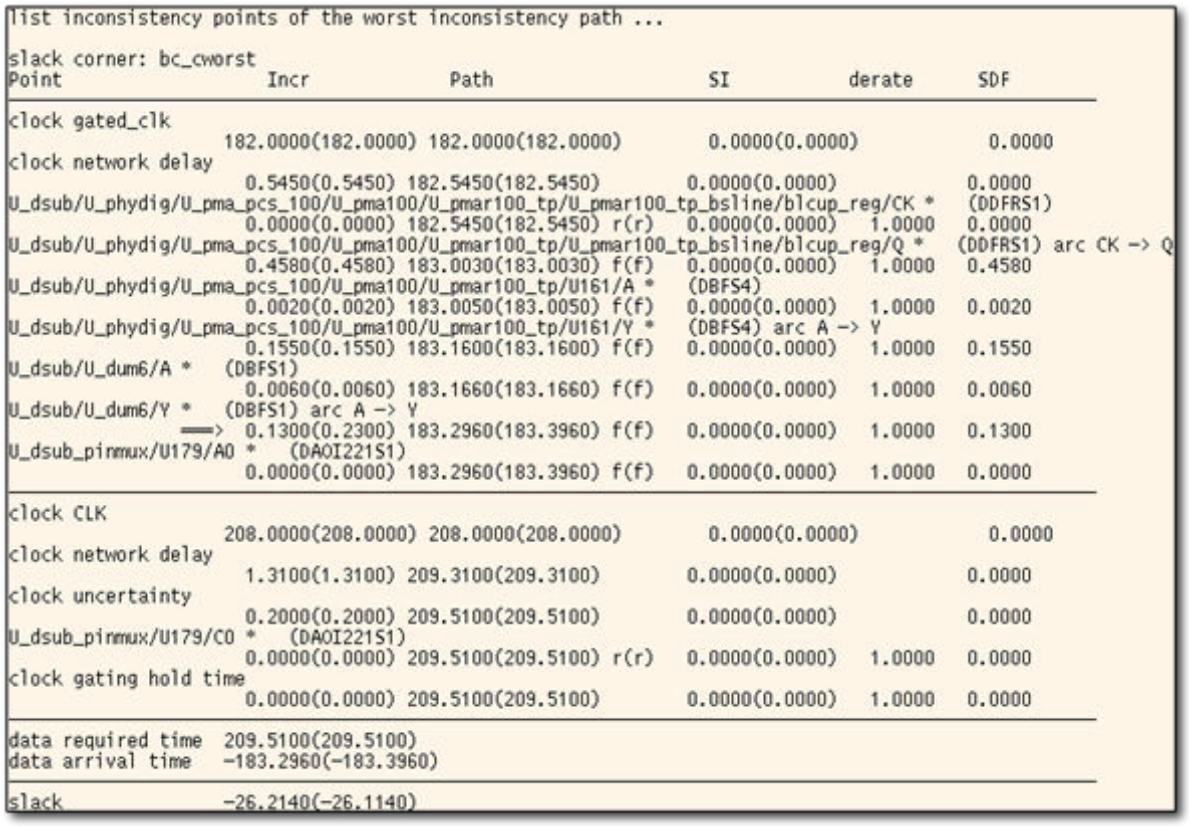
Even after passing a consistency check, a large average offset value – say, greater than 50 ps – is still considered an improper loading since it will contribute uncertainty to ECO Domain timing.

Figure 37 Failed Consistency Check Log

```
[ Error ] Slack value inconsistency over 20ps are found in 1 paths.
 The max. inconsistency of each clock group are listed below:

    clock group    slack in report     slack re-calculated        diff    slack report
-------------------------------------------------------------------------------
          (CLK)              -4.73                   -4.63        0.10    /home/ae1/case
-------------------------------------------------------------------------------
    CRP inconsistency : 0
    SI inconsistency : 0
    SDF/Derating inconsistency : 1
```

Feedback

Figure 38 Most Inconsistent Path in Log File



```
list inconsistency points of the worst inconsistency path ...

slack corner: bc_cworst
Point                    Incr              Path                    SI          derate      SDF
-------------------------------------------------------------------------------------------------------------------------------
clock gated_clk
                    182.0000(182.0000) 182.0000(182.0000)        0.0000(0.0000)            0.0000
clock network delay
                      0.5450(0.5450) 182.5450(182.5450)        0.0000(0.0000)            0.0000
U_dsub/U_phydig/U_pma_pcs_100/U_pma100/U_pmar100_tp/U_pmar100_tp_bsline/blcup_reg/CK *   (DDFRS1)
                      0.0000(0.0000) 182.5450(182.5450) r(r)   0.0000(0.0000)   1.0000   0.0000
U_dsub/U_phydig/U_pma_pcs_100/U_pma100/U_pmar100_tp/U_pmar100_tp_bsline/blcup_reg/Q *   (DDFRS1) arc CK -> Q
                      0.4580(0.4580) 183.0030(183.0030) f(f)   0.0000(0.0000)   1.0000   0.4580
U_dsub/U_phydig/U_pma_pcs_100/U_pma100/U_pmar100_tp/U161/A *   (DBFS4)
                      0.0020(0.0020) 183.0050(183.0050) f(f)   0.0000(0.0000)   1.0000   0.0020
U_dsub/U_phydig/U_pma_pcs_100/U_pma100/U_pmar100_tp/U161/Y *   (DBFS4) arc A -> Y
                      0.1550(0.1550) 183.1600(183.1600) f(f)   0.0000(0.0000)   1.0000   0.1550
U_dsub/U_dum6/A *   (DBFS1)
                      0.0060(0.0060) 183.1660(183.1660) f(f)   0.0000(0.0000)   1.0000   0.0060
U_dsub/U_dum6/Y *   (DBFS1) arc A -> Y
                ==>   0.1300(0.2300) 183.2960(183.3960) f(f)   0.0000(0.0000)   1.0000   0.1300
U_dsub_pinmux/U179/A0 *   (DAOI221S1)
                      0.0000(0.0000) 183.2960(183.3960) f(f)   0.0000(0.0000)   1.0000   0.0000
-------------------------------------------------------------------------------------------------------------------------------
clock CLK
                    208.0000(208.0000) 208.0000(208.0000)        0.0000(0.0000)            0.0000
clock network delay
                      1.3100(1.3100) 209.3100(209.3100)        0.0000(0.0000)            0.0000
clock uncertainty
                      0.2000(0.2000) 209.5100(209.5100)        0.0000(0.0000)            0.0000
U_dsub_pinmux/U179/C0 *   (DAOI221S1)
                      0.0000(0.0000) 209.5100(209.5100) r(r)   0.0000(0.0000)   1.0000   0.0000
clock gating hold time
                      0.0000(0.0000) 209.5100(209.5100)        0.0000(0.0000)   1.0000   0.0000
-------------------------------------------------------------------------------------------------------------------------------
data required time  209.5100(209.5100)
data arrival time   -183.2960(-183.3960)
-------------------------------------------------------------------------------------------------------------------------------
slack               -26.2140(-26.1140)
```

Through the log, you can see the most inconsistent path delay value between an SDF and slack report. For example, the cell delay value for U_dsub/U_dum6 is 130 p at SDF is 130 ps, whereas in the slack report it is 230 ps. This is a 100 ps difference.

The log lets you see the most inconsistent path detail delay value between SDF and slack report. For example, cell delay for U_dsub/U_dum6 is 130 p at SDF value, while it is 230 p at slack report. A 100 p difference is detected.

Inconsistency or Large Offset: Possible Causes

Based on previous experiences, inconsistencies between the slack reports and T1-calculated slack arise for the following reasons:

- Mismatch between SDF and slack report in delay numbers
- The analysis type of slackin is not identical to the type in the STA environment
- OCV derating is not identical to the set in the STA environment

Reasons for the offset between values from the SDF and Tweaker's delay engine may be the following:

- Incorrect timing library, SPEF, or SDF files; these cause mismatch between slack reports and SDF
- External slew and load at the primary inputs/outputs are not properly set by the Tweaker `set_input_transition` or `set_load` commands; the numbers should be consistent with SDC's values.

Tweaker supports multiple sets of input slew and output loading specifications for MMMC. You can place the relevant commands, or source a Tcl file in between the "begin_corner" and "end_corner" lines of a corner-developing block.

How to Verify Inconsistency

To verify inconsistency, refer to the "Check Slack Inconsistency" chapter in the *tweaker_t1_training.pdf* file, which describes the debugging procedure.

Inconsistency Work-Around

By default, Tweaker-T1 will collect inconsistent paths under the GUI's Slack View in the "search column". You can then choose to mark the "don't touch" attribute on these paths using UI commands, and fix paths the remaining paths (assuming they first pass the consistency check).

5

Start Auto Fixing

Previous topics covered how to feed in all required data, build up timing scenarios for Tweaker-T1, and passing the consistency check. With this knowledge, you can continue into auto fixing. This topic shows all the possible auto fix strategies, along with detailed descriptions for addressing different timing issues.

The following topics are covered:

Auto Fixing Overview

Tweaker-T1 implements some important “ECO Thinking” strategies while fixing, and these help to insure the quality of fixing and minimize uncertainty of timing closure.

Feedback

The first “ECO Thinking” strategy is to keep the existing timing unchanged with minimal impact, while Tweaker-T1 works on fixing violations through each incremental job. For this reason, incorporating TWF files into the fixing procedure will prevent interferences with the original timing. For example, using TWF information to fix hold time or DRV (maximum transition) violations will help to avoid any potential setup impacts that come with buffer insertion. In terms of fixing setup time, the TWF information also limits other setup impacts that are caused by sized-up instances.

The second T1-enforced “ECO Thinking” strategy is to start fixing in the safest ECO zones first and then move gradually the dangerous zones, if necessary. Each ECO zone has a corresponding safety factor, which is determined by auto-analysis, as well as constraints determined by you. Tweaker-T1 bases the fixing order of each ECO domain off of these calculated safety factors. Some fixing constraints that distinguish safe or dangerous zones include:

- To neutralize a setup violation, size up a cell without causing cell overlap
- To fix a hold violation, follow minimum required setup slack and limit buffer shift distance

Tweaker-T1’s third “ECO Thinking” strategy is to minimize cost of area by as much as possible. There are a variety of methods to reduce area overhead; for instance, Tweaker can use slack reports to find common points amongst different violation paths to do timing fixes on. As a result of the reduced area overhead, routing congestion and total leakage power significantly lessen.

The final “ECO Thinking” strategy is to assign legal placement to all inserted or moved cells, to prevent overlapping the existing cells in the design. Final placements of every ECO cell will be listed in the partial ECO DEF file that is output from Tweaker-T1.

The following section focuses on the most frequently used features of Tweaker-T1, to provide enough foundation for you to start solving timing issues.

Available Features for Timing Fix

The following available features for timing fix are:

- Cell Swapping
- Cell Sizing
- Bypass Buffers
- Auto-Split Load
- Cell Moving
- Pin Swap

Feedback

- Dummy Load Hookups
- High Fan-Out Buffer Insertion
- Delay Insertion
- Extract Setup Margin
- High Fan-Out Synthesis
- Multithreading Path Update
- Dominate-Based ECO

Cell Swapping

Purpose of cell swapping:

- Fix setup, hold, maximum transition/capacitance violations
- Power ECO
- Clock ECO

The first feature, “VT swapping”, is a meta-form of cell sizing. This particular cell sizing involves changing the cell type (master) of target instances in exactly the way that sizing does, but also limiting the sizing candidates based on a specified “mapping rule”. Modern cell libraries often provide cells with identical sizes, shapes, and pin locations, with the only difference being the VT’s (Voltage Thresholds). High VT cells create less leakage power with the cost of often being slower, whereas low VT cells consume more leakage power but operate at faster speeds.

To adopt cell swapping, switch on the `slk_auto_sizing_rule` variable in advance, and specify a mapping rule under the `slk_cell_mapping_rule` variable, like in Example 23. All cell pairs that are formed from matching the wildcard-supporting rule will be taken into account. If needed, Tweaker-T1 can also support more candidates or rules, similar to Example 24. Two or more mapping rules can be defined simultaneously by using semicolons as separators. In Example 24, the mapping rule is (*HVT to *SVT or *LVT), and (*SVT to *LVT).

Example 23 Specify Mapping Rule for the Auto-Sizing Feature

```
set slk_auto_sizing_rule cell_mapping
set slk_cell_mapping_rule { *HVT *LVT }
```

Example 24 Additional Mapping Rules for Auto-Sizing

```
set slk_cell_mapping_rule { *HVT *SVT *LVT : *SVT *LVT }
```

For complicated mapping rules, you can use the `slk_cell_mapping_rule_regexp` variable. For example, `set slk_cell_mapping_rule_regexp {@HVT @SVT @LVT : @SVT @LVT}` is equivalent to `set slk_cell_mapping_rule {*HVT *SVT *LVT : *SVT *LVT}`, with the only difference in that `slk_cell_mapping_rule_regexp` supports regular expression in mapping rule definitions. For example, `set slk_cell_mapping_rule_regexp { @[0-9]+@ @[0-9]+@ }` allows mapping only to the cells with the identical base name. However, Tweaker-T1 only honors mapping rules defined by one of the either naming rules. If there are two mapping rules set, the latter name setting will override the previous setting. Enter `slk_cell_mapping_rule_regexp` into the Tweaker command line to see further variable usage details.

When you have difficulty defining a cell mapping rule, Tweaker can also help to generate one. See Example 25 for allowing automatic cell mapping rule generation. Based on the standard cell libraries, Tweaker collects cell lists that have the same footprint, considers cells from different libraries with the same footprints as VT swapping, considers cells from the same library with the same footprints as cell sizing, etc. This way, you can control the mapping rules more easily during the ECO stage.

Example 25 Allow Tweaker to Help Define Cell Mapping Rule

```
set slk_auto_sizing_rule [footprint | sizing | vt | vt_sizing | moving |
 …]
```

During timing fixes, VT swapping is usually chosen as the first step because this action does not result in any extra area overhead. VT swapping is applicable for fixing transition, setup time (switch from HVT to LVT to improve driving strength), hold time, Power ECO (swap from LVT to HVT to save leakage), and occasionally for fine-tuning the clock.

Cell Sizing

Purpose of cell sizing:

- Fix setup, hold, maximum transition and maximum capacitance violations
- Power ECO
- Clock ECO

Sizing is a very basic function that changes the cell type of a target instance amongst cells with the same footprint. Unlike VT swapping, some area overhead will appear through a regular sizing action. However, regardless of sizing up cells to fix transition or setup time, Tweaker will choose the target cell to size. While this method helps to satisfy the required slack, it also costs the minimum area overhead among all candidates. Tweaker prefers to keep cells that need to be sized in the same placement location; however, often there is no available space around the target instance. As a solution, Tweaker-T1 provides a flexible function that allows you to legally shift the cell to the target's neighboring areas. See Example 26 for the legal placement shifting variables.

Feedback

For the sake of “Safe ECO”, certain constraints need to be established in order to reduce any potential uncertainty during ECO routing. Some of these constraints, seen in Example 27, include the “maximum fan-out count” on a net and “maximum wire length” on all connections to and from the target instance.

You can manipulate additional constraints that are relevant to sizing combinational logic, buffers, and inverters, or including sequential logic. See Example 28 for these constraints’ variables.

Example 26 Allow Tweaker to Legally Shift Cell to Nearby Location When Sizing (Unit = µm)

```
set slk_auto_fix_fit_to_free_space true
set slk_auto_sizing_max_shift_distance 10
```

Example 27 Relax Constraints to Enable Safe ECO

```
set slk_auto_sizing_max_fanout_limit 32
set slk_auto_fix_max_wire_length_limit 450
```

Example 28 Auto Sizing Constraints

```
set slk_auto_sizing_comb_logic_cell_only true|false
set slk_auto_sizing_buf_inv_only true|false
```

To ensure that sized instances will contribute enough delay improvement, you can set the minimum value of slack improvement. This constraint minimizes the amount of sized instances, to prevent any potentially significant routing (timing) changes later in the ECO process. See Example 29 for the minimum slack improvement variable. Along with each sizing action during the auto fix procedure, Tweaker-T1 uses the variable in Example 30 to block actions that could result in drastic setup impacts.

Example 29 Set Minimum Slack Improvement (Unit = ns)

```
set slk_auto_sizing_min_improved_slack 0.003
```

Example 30 Ensure Sizing will not be Detrimental to Setup

```
set slk_fix_setup_watch_setup_timing_window true
set slk_fix_setup_watch_hold_timing_window true
```

When TWF files are not available, Tweaker-T1 checks the cell delay impact by using the `slk_auto_sizing_cell_delay_impact` variable instead (see Example 31). Unfortunately, using this variable cannot take care of the exact slack impact due to the lack of the TWF pin slack information.

Example 31 Set Cell Delay Impact Limit When TWF is not Available

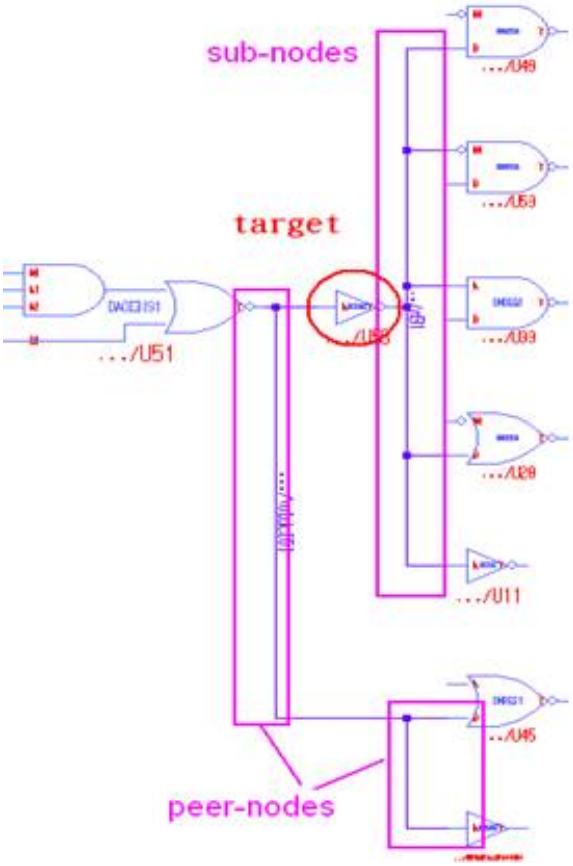
```
set slk_auto_sizing_cell_delay_impact 0.02
```

Note that when auto sizing, Tweaker honors sizing up to fix setup violations using the `set slk_fix_setup_without_sizing_down true` command and sizing down to fix hold violations using the `set slk_fix_hold_without_sizing_up true` command,

Feedback

by default. On the contrary, you can also select a cell size or cell type that seems more suitable for a particular design. By doing this, you can view the combined timing results contributed from the peer's standpoint and sub-node's standpoint of the target instance (see the relationship in Figure 39). Sizing a cell up is usually beneficial to the sub-node side, but it may hurt the peer side by adding on extra loading. Similarly, sizing a cell down usually damages the sub-node side, but it plays in favor of the peer side by lessening its load. Therefore, you can depend on Tweaker-T1's intelligence for analyzing the combined timing result and deciding which cell type is proper for sizing, regardless of a TWF-based or path delay-based updating system.

Figure 39 Relationship Between a Target Instance's Sub-Node and Peer Node



Bypass Buffers

Purpose of bypassing buffers: Fix setup violations

The bypass buffer function is designed to fix setup violations by skipping (or removing) redundant buffers to reduce the number of logic stages, and improves path slack. Figure 40 illustrats a bypass buffer diagram). A buffer inserted at a high fan-out net

Feedback

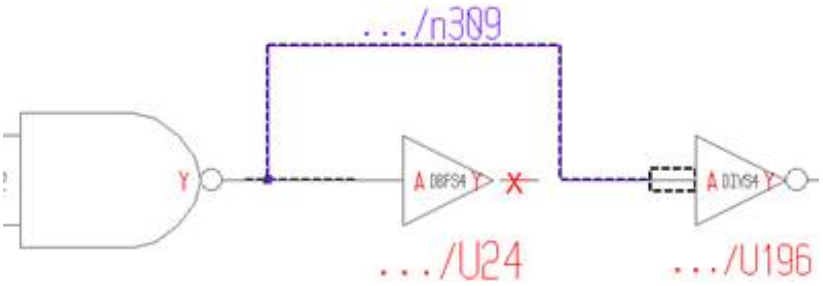
normally acts as a repeater to improve timing. However, the repeater can become a burden for timing critical fan-outs on the same net. Before bypassing the repeater, Tweaker-T1 ensures the bypass will not overwhelm the driver's driving capability. If needed, Tweaker-T1 sizes up the driver to increase its driving strength. In summary, Tweaker-T1 automatically bypasses (or removes) redundant buffers only when the driver is already strong enough.

In addition, Tweaker-T1 can also watch hold time slack through the TWFs to avoid any hold time impacts caused by buffer removals. You can customize the bypass behavior by setting the `slk_bypass_buffer_min_improved_slack` variable which decides minimum improved slack threshold for each bypass action. The purpose of the threshold is to minimize the number of ECO cells, because only the bypass action with slack improvement greater than the `slk_bypass_buffer_min_improved_slack` variable is implemented. See Example 32 for bypass buffer variable settings.

Example 32 Customize Bypass Behavior

```
set slk_fix_setup_by_bypass_buffer true
set slk_fix_setup_watch_hold_timing_window true
set slk_bypass_buffer_min_improved_slack 0.03
```

Figure 40 Bypass Buffer



In reference to the following Tcl scripts, you can enable the bypass buffer attribute by setting the `slk_fix_setup_by_bypass_buffer` variable to `true` and the remaining `slk_fix_setup_by_*` variables to `false`:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/fix_timing/fix_setup/.tcl*

Auto-Split Load

Purpose of automatically splitting load: Fix setup violations

The auto split-load function is useful when there exists a large amount of high fan-out nets on critical setup paths but regular sizing-up does not help improve the timing by much. After activating the split-load feature, Tweaker-T1 implements two algorithms. The first algorithm implements buffer insertion to group non-critical fan-outs and reduce loading on critical fan-outs to speed up timing. The second algorithm works to improve the maximum

transition, which in turn improves the setup timing. This principle is quite similar to fixing transition; Tweaker decides which buffers to insert at non-critical fan-outs or on wires with weak driving cells. A fan-out is considered non-critical if the setup slack of that fan-out is over the `slk_setup_splitload_margin` value. By default, the variable's value is 0.2 ns. During buffer insertion, Tweaker-T1 will honor the repeater buffer list in deciding which buffers are appropriate to use.

Example 33 Variable Settings for Auto Split-Load Procedure to Fix Setup

```
set slk_fix_setup_by_repeater_insertion true
set slk_fix_setup_by_split_load true
set slk_fix_setup_by_hfs true
set slk_setup_splitload_margin 0.2
set slk_fix_setup_repeater_insertion_factor 0.2
set slk_repeater_insertion_buff_list { BUFD2 BUFD4 BUFD8 }
```

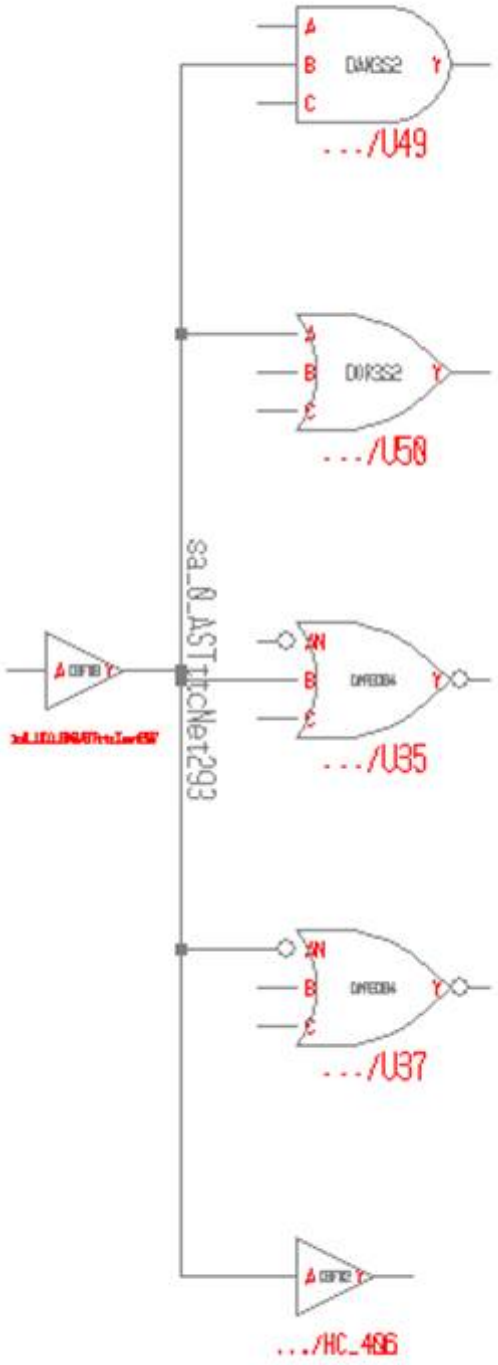
The `slk_fix_setup_max_trans_drv` variable is used to decide whether or not a net will need split-load. The auto split-load procedure will be launched when the transition time of a net is worse than the variable value. To fine-tune the split load, you can adjust the value of the `slk_fix_setup_min_improved_slack_for_repeater_insertion` variable, which will set each split-load action's threshold of the minimum slack improvement. See Example 34 for this variable's usage.

Example 34 Additional variable settings for auto split-load procedure to fix setup

```
set slk_fix_setup_max_trans_drv 0.1
set slk_fix_setup_min_improved_slack_for_repeater_insertion 0.02
```

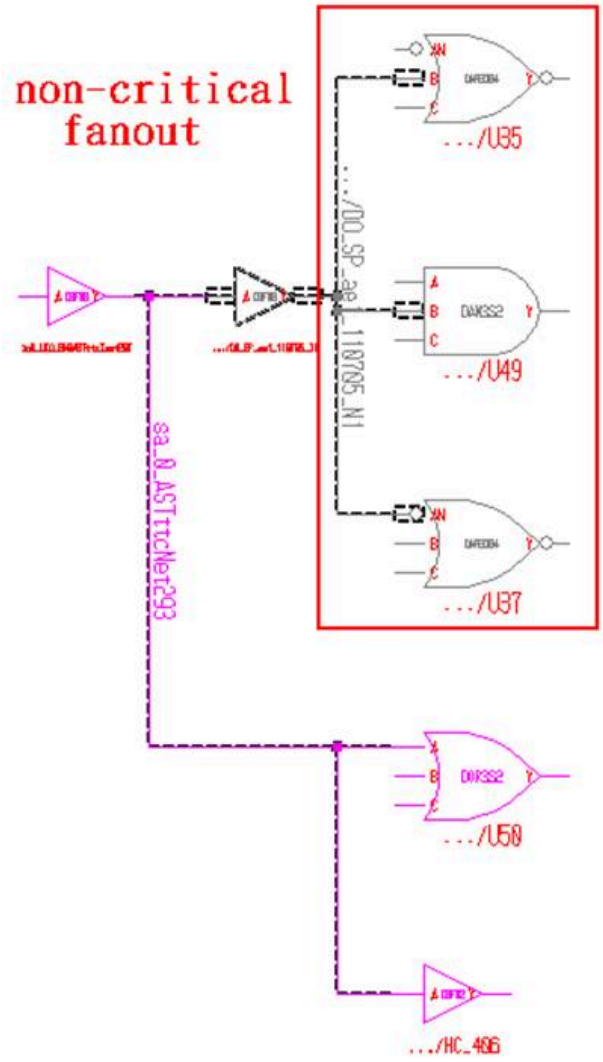
Feedback

Figure 41 Before Auto-Split Load



Feedback

Figure 42 After Auto-Split Load



Cell Moving

Purpose of moving cells: Fix setup violations

Some critical setup violation paths result from poor cell placement. Suppose some key cells should to be placed as close as possible to each other for optimal timing results. However, in some cases, these key components are not placed properly due to the absence of multiple fan-out connections or scenarios during the placement stage. As a result, Tweaker-T1 will use the setup critical paths to identify each component's connections and move them to a better location. Optimal cell placement helps to enhance

the RC and reduce unnecessary wire delay, and therefore improves setup timing results. In addition, the you can choose to size up the driver cell to potentially support sink cell movement. Since cell-moving is considered a part of the sizing methodologies, you must enable the `slk_auto_sizing_rule moving` setting for setup ECO. In addition, you can use the `slk_auto_sizing_max_shift_distance` variable to customize the moving radius. See Example 35 for the variables to implements these fixing actions. For additional variables and cell-moving settings, refer to the template script in the following directory:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/fix_timing/fix_setup/fix_setup_setting.moving.tcl

Example 35 Variable Settings for Auto Cell-Moving Procedure to Fix Setup

```
set slk_fix_setup_by_sizing true
set slk_auto_sizing_rule moving
set slk_auto_sizing_max_shift_distance 20
```

Pin Swap

Purpose of pin swapping: Fix setup violations

Some combinational cell types with two or more input pins can have their pins swapped, and still keep the same functional equivalence. For instance, an AND gate will operate the same way, regardless of the order of input pins' connections. However, according to the timing library of the AND gate, pin A and pin B will have different timing arcs that propagate different cell delays. Therefore, Tweaker will base its pin swap decision on the timing library information on the A and B pins. With this innovative pin swap feature, Tweaker can help improve critical setup timing without any cost. To enable pin swapping for setup ECO, set the `slk_fix_setup_by_pinswap` variable as shown in Example 36. For more detailed command usage, see the template script in the directory following directory:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/fix_timing/fix_setup/fix_setup_setting.pinswap.tcl

Example 36 Variable Settings for Auto Pin-Swap Procedure to Fix Setup

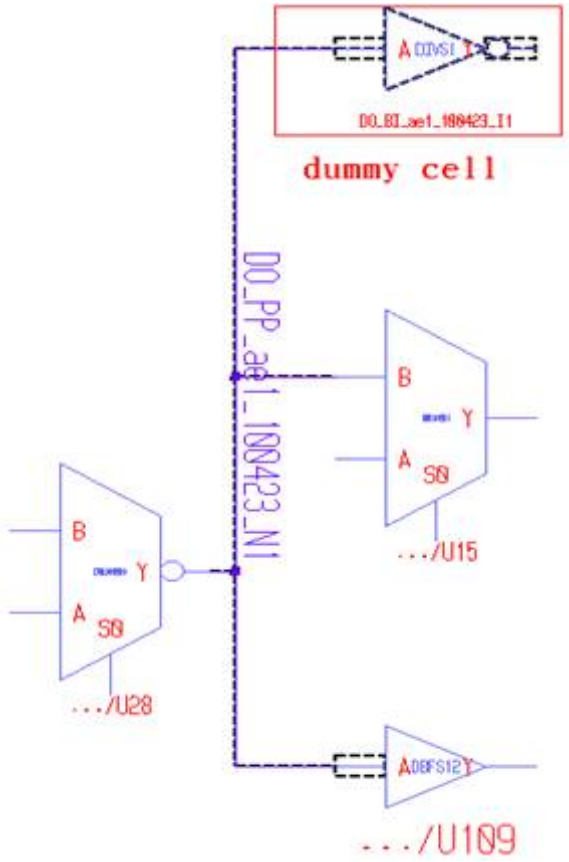
```
set slk_fix_setup_by_pinswap true
```

Dummy Load Hookups

To save area overhead and fix minor hold violations using buffers, Tweaker-T1 has a function that attaches a dummy load onto an existing net such that the input capacitance and the extra wire loading from the attached dummy will increase delay. You should use a small inverter as the dummy load to contribute input capacitance with minimal area impact. All attached dummy inverters will have a floating output to avoid functional issues. See Figure 43 for a visual reference of the dummy load methodology.

Feedback

Figure 43 Add Dummy Cells



For instance, if a path is only violating hold time by -5 ps, it is more efficient to attach a small inverter onto its net, since buffer insertion would case overfixing. You should use the dummy load hook-up method before buffer insertion to fix minor violations with very little resulting area overhead. Example 37 shows the appropriate variable settings to use inverters as dummy loads, specify the maximum number of dummy loads to insert on each sink pin, and set the maximum distance each dummy load can be placed away from every sink pin. See the following directories for additional dummy load setting usage:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/fix_timing/fix_hold/fix_hold_setting.dmy.tcl

Example 37 Variable Settings for Auto Dummy-Load-Hookup Procedure to Fix Hold

```
set slk_fix_hold_by_add_dummy_load true
set slk_range_for_add_dummy_load -0.005 0
set slk_dummy_load_cell_list { INVD0 }
set slk_max_dummy_load_cell_count 2
set slk_max_dummy_load_cell_distance 10
```

Feedback

High Fan-Out Buffer Insertion

Purpose of high fan-out buffer insertion: Fix hold violations

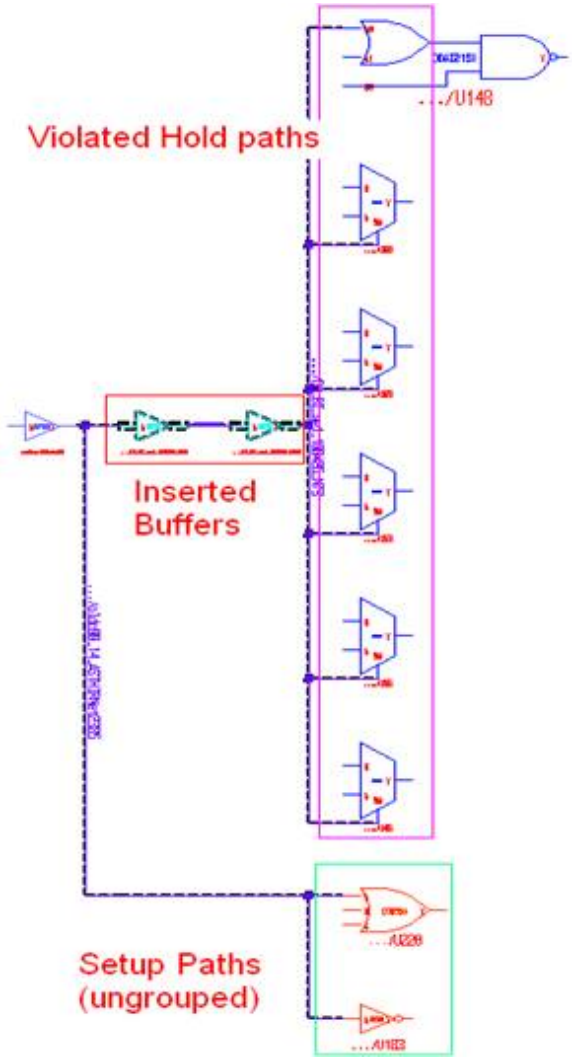
This smart buffer insertion function is developed to fix hold time violations. It requires TWF files in advance to provide the setup and hold slack information that is later used for “smart grouping”. The smart group strategy categorizes fan-outs into groups (hold groups and setup groups), then intelligently inserts buffers before the fan-out groups without creating new ports. Tweaker-T1 makes sure to use a minimal number of hold buffers, while keeping the setup slack on each node of the net intact. Example 38 shows the necessary settings to make high fan-out buffer insertion possible. Peer fan-outs with a hold slack less than the value of the `slk_hfi_peer_hold_margin` variable will be grouped and driven by the inserted buffers. The `slk_fix_setup_target_slk` variable specifies that the setup time on each node must keep a total safe margin of 0.1ns after buffer insertion.

Example 38 Variable Settings for Auto HFI Procedure to Fix Hold

```
set slk_fix_hold_watch_timing_window true
set slk_hfi_peer_hold_margin 0.02
set slk_fix_setup_target_slk 0.05
set slk_fix_hold_by_delay_insertion true
set slk_fix_hold_by_high_fanout_insertion true
```

Feedback

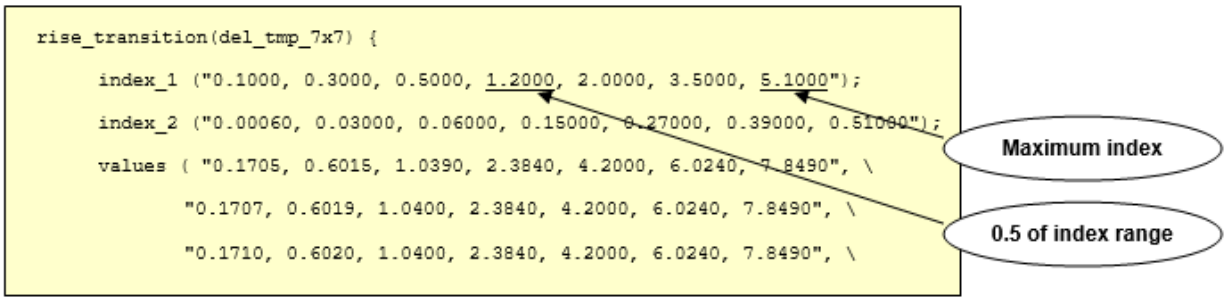
Figure 44 High Fan-Out Insertion (HFI)



To avoid corrupt timing results after ECO routing, it is highly recommended to leave some extra margin for the target setup slack. Aside from checking setup time, Tweaker-T1 also uses the `set_drv_factor` variable to check DRV (Design Rule, or slew, Violation) to avoid any potential transition problems caused by insertion of weak buffers on long wires. This variable defines the level of transition/ capacitance to maintain during any ECO job. By default, the DRV factor is applied to derate the ranges of the transition index on the timing table. To apply the DRV factor to derate the maximum value of the transition index instead, set the `drv_derate_at_index` variable to `false`. Figure 45 shows the major difference between the two variables.

Feedback

Figure 45 How drv_derate_at_index Works



Case 1:

```
set drv_derate_at_index true    # default true
set_drv_factor 0.5
```

The DRV value is 1.2

Case 2:

```
set drv_derate_at_index false
set_drv_factor 0.5
```

The DRV value is 5.1000 * 0.5 = 2.55

To speed up the application of high fan-out insertion (HFI), you should prepare a proper “delay buffer” list for Tweaker-T1. You should abide to the following rules when choosing delay buffer candidates for the list:

Delay Buffer Candidate Requirements

1. Limit each list to a maximum of 7 cells so that each list can be processed efficiently
2. Add and apply any available HVT cells
3. Provide at least one high-drive buffer to the list so that it can be used for long wires
4. Include some delay cells to fix large hold violations and minimize insertion count

Example 39 Specify Delay Buffer Candidates

```
set slk_delay_insertion_buff_list { BUFFD1HVT BUFFD2HVT BUFFD4HVT
 BUFFD8HVT DEL01 }
```

If the `slk_fix_hold_minimized_buffer_count` variable (`true`, by default) is switched on, Tweaker-T1 will tend to insert delay cells more often to save the buffer insertion count. This strategy is perfect for high-density designs where insertion count matters. See Example 40 for the variable’s usage.

Feedback

Example 40 Reduce Insertion Count While Fixing

```
set slk_fix_hold_minimize_buffer_count true
```

During HFI, Tweaker-T1 calculates the ideal placement for each inserted buffer, however, the calculated space frequently does not exist around the ideal point. Therefore, Tweaker-T1 needs to extend searching resources for legal placement locations. For this purpose, the `slk_give_up_insert_if_no_space` and `slk_give_up_insert_buf_distance` variables help Tweaker-T1 to decide how much extended resource is needed. The settings in Example 41 enable Tweaker-T1 to extend a 50 µm resource box around the ideal placement to determine whether or not insertion is valid in this area.

Example 41 Search for Legal Placement Locations

```
set slk_give_up_insert_if_no_space true
set slk_give_up_insert_buf_distance 50
```

In summary, the HFI methodology combines several metrics to effectively insert buffers while maintaining existent timing within the required constraints. You should adopt HFI as a major strategy when hold time violations are widely spread across multiple timing scenarios. Refer to the following directory for more details:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/fix_timing/fix_hold/fix_hold_setting.bi..tcl*

Delay Insertion

Purpose of delay insertion: Fix hold violations

Besides advanced HFI, Tweaker-T1 keeps simple delay insertion as a backup solution for fixing hold time violations. This strategy does not apply smart grouping; on the contrary, it simply inserts buffers at the sink pin of certain slack instances. If you prefer to insert buffers only at the endpoints, set the `slk_fix_hold_at_endpoint_only` variable to `true`. The number of inserted buffers is contingent on the desired delay cell distance, which is defined by the `slk_preferred_delay_cell_distance_range` variable. Example 42 shows that simple buffer insertion will be done every 6 to 9999 µm. For additional detailed usage, refer to the following script template directory:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/fix_timing/fix_hold/fix_hold_setting.bi.4.tcl

Example 42 Variable Settings for Auto Delay Insertion Procedure to Fix Hold

```
set slk_fix_hold_by_delay_insertion true
set slk_fix_hold_by_high_fanout_insertion false
set slk_preferred_delay_cell_distance_range 6 9999
```

Feedback

Extract Setup Margin

Purpose of extracting setup margin:Fix hold violations

Even after ECO, Tweaker cannot fix a few of the remaining violations due to insufficient setup margin for hold delay insertions. To solve this issue, you can enable a special Tweaker feature called “Extract Setup Margin” during hold time ECO. To implement this feature, first source the environment setting and set the mapping rule with the `slk_cell_mapping_rule_regexp` variable. To extract the setup margin based on the original hold timing critical path, enable the `slk_fix_hold_by_extract_setup_margin` variable. To extract the setup margin based on the hold timing critical path’s fan-in and fan-out cones, enable the `slk_fix_hold_by_extract_extra_domain_setup_margin` variable. If necessary, you can enable both of these extract setup margin capabilities simultaneously. After these environment settings are specified, Tweaker can expand the setup margin at hold conflict paths. See Example 43 for extract setup margin variable settings.

Example 43 Auto Extract Setup Margin Procedure to Fix Hold

```
source $script_path/fix_hold_setting.vtswap.tcl
set slk_cell_mapping_rule_regexp {@ @HVT}
set slk_fix_hold_by_extract_setup_margin true
set slk_extract_setup_margin_trans 0.06
set slk_fix_hold_by_extract_extra_domain_setup_margin true
```

High Fan-Out Synthesis

Purpose of high fan-out synthesis: Fix maximum transition and capacitance violations

To fix maximum transition or maximum capacitance with buffer insertion, particularly for long wires, Tweaker-T1 provides a high fan-out synthesis (HFS) function which reduces the loading of a driver by using buffers to group fan-outs. Before HFS, you must specify a “repeater” buffer list, each with at most 5 repeaters assignments to save on runtime. Example 44 shows HFS, enabled using the latest version to fix transition. To use inverter pairs to implement maximum transition, see Example 45 for the appropriate settings. To use a inverter/buffer list for maximum transition ECO or use inverters to fix transition violations, enable the `slk_hfs_use_inverter` and `slk_hfs_inverter_only` variables, respectively. In addition, Example 46 shows the environment settings for fixing maximum transition in clock paths.

Example 44 Variable Settings for Auto HFS Procedure to Fix Maximum Transition/Capacitance

```
set slk_fix_max_trans_by_repeater_insertion true
set slk_repeater_insertion_buff_list {BUFFD4 BUFFD6 BUFFD8 BUFFD16}
```

Feedback

Example 45 Use Inverter Pairs to Fix Maximum Transition

```
set slk_fix_max_trans_by_repeater_insertion true
set slk_hfs_use_inverter true
set slk_hfs_inverter_only true
set slk_repeater_insertion_inverter_list{INVD4 INVD6 INVD8 INVD16}
```

Example 46 Environment Settings for Fixing Maximum Transition in Clock Path

```
set slk_fix_max_trans_by_repeater_insertion true
set slk_fix_max_trans_dont_touch_clock_tree false
set slk_hfs_use_inverter true
set slk_hfs_inverter_only true
set slk_repeater_insertion_clock_buff_list {CKBUFFD4 CKBUFFD6}
set slk_repeater_insertion_clock_inverter_list{CKINVD4 CKINVD6}
```

When executing HFS, Tweaker-T1 checks if inserted buffers violate the required setup target slack. Refer to the following directory for more details:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/fix_timing/fix_max_transition/.tcl*

Multithreading Path Update

To reduce the ECO runtime of update slack paths during autofixing, use the multithreading path update mechanism. In the ECO stage, each ECO operation needs to update the related slack paths. The more slack paths there are to update, the more runtime is improved.

To enable this feature, set the `slk_enable_multi_thread_path_update` variable to `true` before autofixing:

```
[CMD] set slk_enable_multi_thread_path_update true
```

The thread number used to update the path is the scenario count x 2.

Dominate-Based ECO

Dominate-based ECO is used for designs with a large path count (over 100K paths) and focuses on fixing violations on major paths. This enhances the path-based timing fixing performance and quality of results.

To enable dominate-based ECO, ensure that the `slk_enable_dominicate_path_fixing` variable is set to `true` (the default). You can also set the threshold value with the `slk_dominate_path_fixing_threshold` variable. The default is set to 100000. If the violation path count is more than the threshold, the tool focuses only on the dominate path fixing. However, if the violation path count is more than the threshold and the dominate-based path fixing is not enabled, the tool watches all paths.

Use the following command syntax to enable dominate-based ECO:

```
set slk_enable_dominate_path_fixing true
set slk_dominate_path_fixing_threshold threshold_value # default value is
 100000
set slk_auto_fix_max_loop loop_value
slkfix -setup -all
```

Start Fixing Hold Violations

Hold time fixing is one of the major reasons that you choose Tweaker-T1. As described in Fixing Hold Time Violations, you are encouraged to VT swap, size and add dummy cells in the beginning of the ECO iteration to sweep out minor violations with minimum cost of area and leakage power. After that, high fan-out insertion (HFI) will resolve most of hold violations. For all of the remaining setup/ hold violation paths, you can use the extract setup margin methodology to optimize setup at hold-critical paths. At the end, simple buffer insertion can clean the remaining violations. For the descriptions of the mentioned features, refer to Available Features for Timing Fix.

Instead of following to the “90% auto fix and 10% manual fix” strategy in fixing setup time, you might prefer targeting all existing hold violations and fixing them through all scenarios in a single round. Thus, to minimize the number of ECO iterations, you should raise the NWORST number in the following directory, so that Tweaker can dump a slack report that covers more violation paths:

$Tweaker_Install_Dir/dorado/etc/scripts/tcl/pt/report_hold_to_tweaker.tcl

Unfortunately, increasing the NWORST value will result in a larger slack report, which costs more runtime and memory. The following case outlines the amount of resources needed for a typical design:

Design profile: 5.2M / 8 sub-design blocks

Violating hold paths: 24362

Number of scenarios: 32

Runtime: 1hr 45mins

Memory: ~15G

Feedback

Start Fixing Setup Violations

Since Tweaker reads in setup path reports that are output from STA tools, you must dump the slack reports in a format compatible with Tweaker-T1. The script to dump PrimeTime slack reports in a Tweaker-suitable format is in the following directory:

$Tweaker_Install_Dir/dorado/etc/scripts/tcl/pt/report_setup_to_tweaker.tcl

Tweaker-T1 does not fix timing path by path. Instead, Tweaker uses all input setup and hold slack reports to build up “ECO slack path domains”, and then prioritizes the fixing candidates based on the result of cost functions consisting of total slack through the pins, slew constraints, fan-out constraints, and so on. In this way, the target instances at the timing bottleneck will be fixed in correspondence to its higher priority. You should use the templates in the following directory to fix setup violations:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/fix_timing/fix_setup/.tcl*

Usually, VT swapping or sizing are the most efficient and safe methods to bring down the distribution of violation counts. Besides sizing, Tweaker-T1 also provides bypass buffer, split load, cell moving, and pin swapping options which are designated for fixing setup time. Refer to Available Features for Timing Fix for descriptions of the available features.

The sizing option allows two strategies for you to focus on:

- Bring down violation count while keeping existing timing intact as much as possible
- Minimize the worst slack, with the cost of potentially impacting other paths’ timing

Normally for a chip’s scope, the first strategy is ideal for timing closure purposes. On the other hand, an IP (Intellectual Property) macro scope, which could be a CPU block, is fit for the second strategy since the worst slack determines how fast the block will be running. The variable provided for the purpose is `slk_fix_setup_minimize_worst_slack`.

Tweaker-T1 also allows you to apply sizing actions on lists of instances specified within a file (see Example 47 for variable usage). This usage model is particularly useful when recovering timing impact after Power ECO. Refer to Leakage Power ECO With Tweaker-P1 for the Power ECO flow, and see the following directories for its detailed usage:

*$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/power_eco/**

*$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/recovery/**

Example 47 Only Size Instances Listed Within an Instance File

```
set slk_auto_sizing_within_inst_file true
set slk_auto_sizing_inst_file list_file
```

Feedback

Some sizing options are related to don't-touch cell or don't-use cell variables. For example, when some instances with specific cell types should stay unchanged or some candidates of a footprint are not welcome, you can use the variables in Example 48 to help restrict the sizing behavior. In Example 48, the `slk_dont_use_cells` variable will exclude OD cells from being candidate target cells, while the `slk_dont_touch_cells` variable will keep instances of FA or HA cells untouched.

Example 48 Choose Which Cells to not Touch During Fixing Process

```
set slk_dont_use_cells {*OD*}
set slk_dont_touch_cells {*FA* *HA*}
```

Start Fixing Maximum Transition, Capacitance, Fan-Out, and Noise

Tweaker-T1 reads in "report_constraint" reports written by STA tools, which contain maximum capacitance, maximum transition, noise, and maximum fan-out violations in various formats. Refer to either of the following directories to generate constraint reports in the format suitable with Tweaker-T1:

$Tweaker_Install_Dir/dorado/etc/scripts/tcl/pt/report_max_*_to_tweaker.tcl
$Tweaker_Install_Dir/dorado/etc/scripts/tcl/pt/report_noise_to_tweaker.tcl

It is ideal to start fixing these violations by VT swapping to save area overhead, then sizing up, and finishing with high fan-out synthesis (HFS) for high fan-out nets or long wires. The VT swapping, cell sizing, and HFS methodologies are explained in the Available Features for Timing Fix topic.

Before fixing, you can set up an instance naming pattern for inserted instances using the `naming_rule` variable, such as in Example 49. The inserted instance will be named similar to "dorado_0401_dsp0_max_cap_I1", which helps you identify the cells in the remaining design cycle. Some naming abbreviations include:

- HFS ("high fan-out synthesis") to fix maximum transition, capacitance, noise, and setup
- HFI ("high fan-out insertion") to fix hold
- HBI ("hold buffer insertion") to fix hold
- ADL ("add dummy load") to fix hold

Example 49 Set a Naming Rule

```
set naming_rule dorado_date_design_max_cap_
```

It is highly recommended that you specify a proper repeater buffer list for HFS. For example, `set slk_repeater_insertion_buff_list {BUFFD4 BUFFD8 BUFFD12 BUFFD16}`.

If this variable is not set in advance, Tweaker-T1 gathers all available buffers in the .lib file as candidates by default for buffer insertion, which can potentially be time consuming.

Refer to the following directory for example scripts for fixing maximum transition/ capacitance/fan-out/noise:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/fix_timing/fix_max_transition.

Start Fixing Minimum Pulse Width

With minimum pulse width ECO, you can fix minimum pulse width violations by reducing the rising or falling delay difference. The delay difference can be reduced by sizing the buffers and inverters to lower the rising or falling variation cells on violations paths. By sizing high rising or falling delay cells to low rising or falling delay cells, the delay difference becomes balanced.

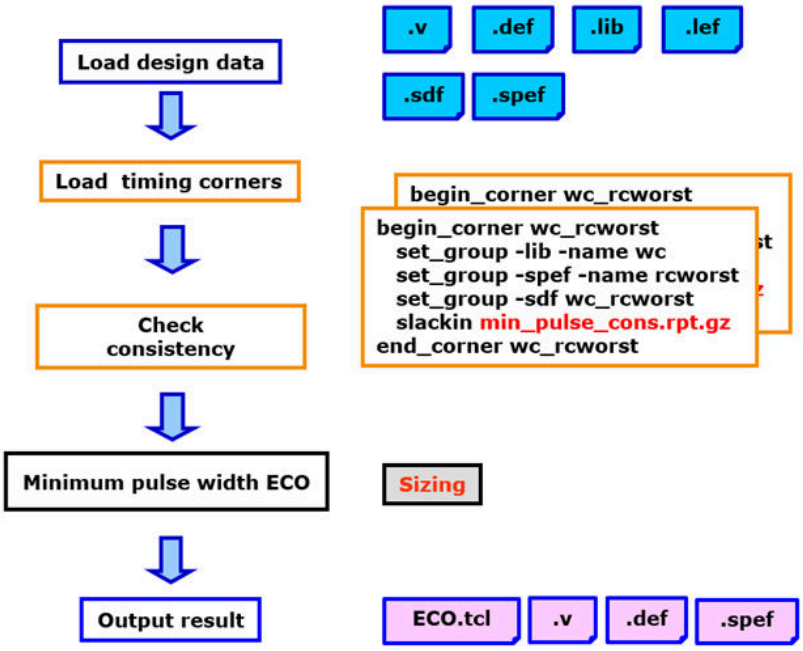
You can also fix violations by improving the input transition of a constraint pin, if the required value of the minimum pulse width is determined by the input transition. By sizing the driving cell of the constraint pin, the required value of the minimum pulse width is reduced.

The following are the steps for the minimum pulse width ECO flow:

1. Load the design data
2. Load timing scenario to read the minimum pulse width violation report
3. Check consistency
4. Enable minimum pulse width ECO
5. Output the results

Feedback

Figure 46 Minimum Pulse Width ECO Flow



Tweaker can perform minimum pulse width autofix to fix the pulse width violations. The sizing method is used to fix forbidden chain violations. Set the following variable to `true` to fix minimum pulse width violations with the sizing method:

```
[CMD] set slk_fix_min_pulse_width_by_sizing true
```

To perform minimum pulse width ECO, run the following command:

```
[CMD] slkfix -min_pulse_width
```

Additional settings that can be used in minimum pulse width ECO include:

- Specifying the target slack with the following command:

  ```
  [CMD] set slk_fix_min_pulse_width_target_slk target_slack
  ```

- Setting whether to include sequential cells with the following command:

  ```
  [CMD] set slk_auto_sizing_combo_logic_cell_only false
  ```

- Providing the sizing rule when fixing violations by the sizing method with the following command:

  ```
  [CMD] set slk_auto_sizing_rule vt_sizing
  ```

Example 50 is a template example of loading the timing corners and the minimum pulse width report.

Example 50 Loading the Timing Corners and the Minimum Pulse Width Report Template

```
set LIB {wc bc}
set PARA {cworst cbest}
set MODE {norm scan}
foreach lib $LIB {
  foreach para $PARA {
    foreach mode $MODE {
      begin_corner ${lib}_${para}_${mode}
        set_group -lib -name $lib
        set_group -spf -name $para
        set_group -sdf -name ${lib}_${para}_${mode}
        twfin -analysis_type on_chip_variation
 "${twf_path}/${lib}_${para}_${mode}.twf.gz"
        slackin -mode $mode -type sdf_max
 "${rpt_path}/${lib}_${para}_${mode}.constraint_mpw.rpt"
        end_corner ${lib}_${para}_${mode}
        }
    }
}
slkdc -check_slack_consistency
```

Example 51 is a template example of the minimum pulse width setting to fix violations with the sizing method.

Example 51 Setting to Fix Violations With the Sizing Method Template

```
source $script_path/fix_min_pulse_width_setting.sz.tcl
set slk_cell_mapping_rule_regexp { @D[0-9]+ @D[0-9]+ }
slkfix -min_pulse_width -all
```

Example 52 is an example of a minimum pulse width ECO setting template:

Example 52 Minimum Pulse Width ECO Setting Template

```
############################################################
# Setting for Fixing Min Pulse Width (type "printvar slk" to see more
 options)
############################################################
# Turn on Sizing for Minimum Pulse Width Fix
set slk_fix_min_pulse_width_by_sizing true

# Fix minimum pulse width constraint
set slk_fix_keep_pin_geometry false
set slk_auto_fix_fit_to_free_space true

# sizing constraint
set slk_auto_sizing_high_effort false
set slk_auto_sizing_max_shift_distance 10
set slk_auto_fix_max_wire_length_limit 1200
set slk_auto_sizing_max_fanout_limit 120
set slk_auto_sizing_keep_long_wire_slew false
set slk_auto_sizing_min_improved_slack 0.005
set slk_auto_sizing_comb_logic_cell_only false
```

Feedback

```
# Minimum pulse width target slack
set slk_min_pulse_width_target_slk 0.005
```

Example 53 shows a minimum pulse width full clock expanded report:

Example 53 Minimum Pulse Width Full Clock Expanded Report

```
[CMD] report_min_pulse_width -all_violators -derate -path_type \
             full_clock_expanded -input_pins -significant_digits 4

  Pin: END
  Related clock: CLK_1
  Check: clock_tree_pulse_width

  Point                                    Incr       Path
  -----------------------------------------------------------
  clock CLK_1 (fall edge)                  0.7450     0.7450
  clock CLK (source latency)               0.0000     0.7450
  SOURCE (in)                              0.0000     0.7450 f
  CLKINST_1/X (CKBUF1)                     0.5577 *   1.3027 f
  END (out)                                0.0000 *   1.3027 f
  open edge clock latency                             1.3027
  clock CLK_1 (rise edge)                  1.4900     1.4900
  clock CLK (source latency)               0.0000     1.4900
  SOURCE(in)                               0.0000     1.4900 r
  CLKINST_1/X (CKBUF1)                     0.5118 *   2.0018 r
  END (out)                                0.0000 *   2.0018 r
  clock reconvergence pessimism            0.0001     2.0019
  close edge clock latency                            2.0019
  -----------------------------------------------------------
  required pulse width (low)                          0.7152
  actual pulse width                                  0.6992
  -----------------------------------------------------------
  slack (VIOLATED)                                   -0.0160
```

ECO Compare, ECO Place, and ECO Synthesis

Occasionally during the ECO phase, you precede timing ECO with functional ECO. During functional ECO, front-end designers may add some new logic gates. In order to precisely reflect the timing influence on these newly added, but unplaced, instances, Tweaker provides ECO Compare, ECO Place, and ECO Synthesis functions as described in the ECO Compare, ECO Place, and ECO Synthesis topic.

The following topics are covered:

- Benefits of ECO Compare, ECO Place, and ECO Synthesis
- Required Input Files and Settings

Feedback

- ECO Place and ECO Synthesis for Designs With Multiple Power Domains
- Partial SPEF and Partial DEF Files

Benefits of ECO Compare, ECO Place, and ECO Synthesis

- Give functional ECO cells legal placement, with a sense of timing
- Reflect precise timing prediction for functional ECO cells

Required Input Files and Settings

For ECO Compare, ECO Place, and ECO Synthesis, it's unnecessary to establish multiple timing scenarios. All you have to do is load the design with only one timing scenario, ideally a slow scenario, then use the `eco_compare -netlist { original.v }` command to read and compare the ECO netlist, or simply use the `eco_compare` command to compare the new netlist against the old DEF file. This straightforward flow is shown in the following steps. For example scripts, refer to the following directory:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/eco_place/ecoplace.tcl

Required input files:

- Library cell related: LIB, LEF
- Design related: Netlist, DEF, SPEF, SDF

Contents within each scenario for ECO Place:

```
begin_corner scenario_name
set_group -lib -name library_group
set_group -spef -name RC_group
set_group -sdf -name sdf_file
end_corner scenario_name
```

Steps:

1. Recognize and construct the ECO Domain

   Compare the post-ECO netlist with the original netlist to identify all newly added ECO cells and connectivity changes.

   ```
   eco_compare -netlist original_netlist    # compare two netlist
   ```

   or

   ```
   eco_compare              # compare new netlist and old DEF
   ```

2. Place ECO cells into the design

```
eco_place
```

3. Output partial SDF and partial DEF files

```
defout directory/file_prefix
sdfout directory/file_prefix
```

ECO Place and ECO Synthesis for Designs With Multiple Power Domains

During ECO Place and ECO Synthesis, Tweaker honors each power domain definition both physically and logically. Logical power domains can be defined by either a CPF or UPF file, while physical power domains can be defined by the `create_voltage_area` command. Refer to the Low Power Application in Tweaker topic for more power domain details.

Tweaker will first recognize the logical power domains that the ECO cells reside in, and then place them into each of their corresponding physical power domains. Note that the name of the logically or physically defined power domains must matched such that Tweaker can map them to one another correctly.

Partial SPEF and Partial DEF Files

Partial SPEF files that are output from Tweaker should be back-annotated to the STA tool, to check if any timing violations are induced by functional ECO.

Partial DEF files are delivered to the back-end team for P&R tools to do ECO routing. Since Tweaker places its ECO cells while taking timing into consideration, it is not recommended for you to activate ECO place and cell legalization in their P&R tool.

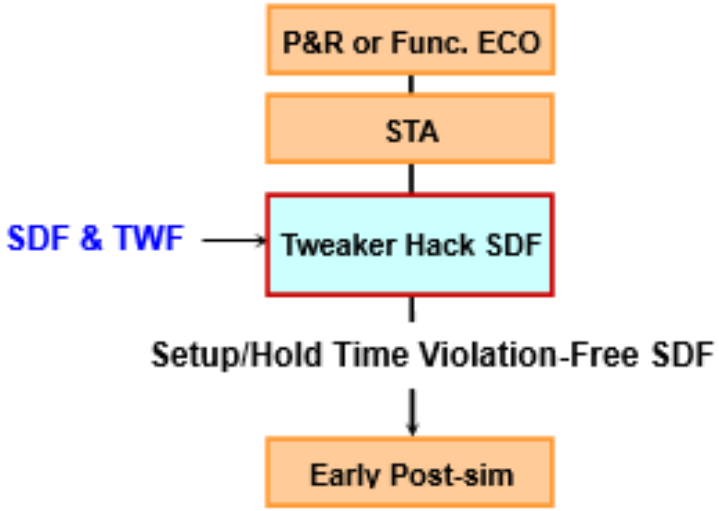
Hack SDF for Post-Sim

As stated in the Hack SDF for Early Post-Layout Simulation topic, designers can use a hacked timing-violation-free SDF to start the post-layout simulation and survey potential problems earlier on, while keeping the current design unchanged.

Hack SDF is one of the most popular features. Tweaker-T1 can read either TWF files or violation reports and write out an SDF file, clear of violations, for a violation-free logic simulation. Since there is no netlist ECO in this process whatsoever, the simulation takes only minutes to complete. Hack SDF can be used either at the post-P&R stage or post-ECO stage.

Feedback

Figure 47 Hack SDF Incorporated Into Flow



Required Input Files and Settings

You can apply Hack SDF to hack setup and hold violations in a single run of Tweaker-T1, and its flow is exactly the same as a regular timing fix flow. The Hack SDF flow simply follows the same design loading procedure and reads the TWF file within the established timing scenario, then “pretends” to fix the hold and setup time violations.

Required input files:

- Library cell related: LIB, LEF
- Design related: Netlist, SDF
- Scenario related: TWF

Contents within each scenario for Hack SDF:

```
begin_corner scenario_name
set_group -lib -name library_group
set_group -sdf -name sdf_file
twfin -analysis_type on_chip_variation twf_file
end_corner scenario_name
```

During the hacking operation, there are three variables that matter the most, the first two being `slk_setup_target_slk` and `slk_hold_target_slk`. Tweaker-T1 will honor these first two variables to guarantee reaching the hold target and meeting setup time as much as possible. The third variable is `slk_hack_setup_min_cell_delay` and it controls the

Feedback

minimum cell delay kept on the hack setup instances during the delay-decreasing process for hacking setup. To start hacking by using the TWF, you should specify the `slkfix -twf_hack_sdf` command in place of the command that fixes regular slack reports, `slkfix -hold/setup -all`. See Example 54 for these variables' usages. Example scripts for Hack SDF are under the following directory:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/hacksdf/hacksdf_by_twf/

Example 54 Hack SDF Using TWF Settings

```
set slk_hold_target_slk 0.02
set slk_setup_target_slk 0.01
set slk_hack_setup_min_cell_delay 0.005
slkfix -twf_hack_sdf
```

Clock ECO With Tweaker-C1

There are two ways to do clock ECO, provided that the "tweaker_c1" license is available. The first way is to feed PrimeTime slack reports, with the clock network fully expanded, into Tweaker and launch the clock surgery flow with well-designed clock ECO GUI. The second clock ECO method is to generate PrimeTime TWF reports for Tweaker and create the ECO domain by Tweaker's tracing scheme, then do clock path skew optimization.

Tweaker-C1 can parse TWF information to identify each flip-flops' setup and hold timing. Based on each flip-flop, you can build the clock structure for setup and hold timing graphs, and apply Tweaker-C1's useful skew setting to determine their fixing strategies.

The current options available for Clock ECO are sizing and buffer insertion. Sizing is Tweaker-C1's default, and is highly recommended for Clock ECO, as most sizing actions keep cell placement fixed so that there is almost no routing changes on the sensitive clock network.

The `slk_ftc_setup_max_fine_tune` variable, which sets the maximum allowed skew value for each setup path, and `slk_ftc_hold_max_fine_tune` variable, which sets the maximum allowed skew value for each hold path. See Example 55 for these variables' usages.

In order to minimize impact to the routed clock wires, Tweaker-C1 will only insert buffers at the end of a clock path (right before the clock pin of a register).

Example 55 Clock Fine-Tuning Settings for Setup Time

```
By sizing:
set slk_ftc_fix_setup_by_sizing true
set slk_ftc_setup_max_fine_tune 0.1
slkfix -setup -all

By buffer insertion:
set slk_ftc_fix_setup_by_insertion true
```

Feedback

```
set slk_ftc_setup_max_fine_tune 0.05
set slk_repeater_insertion_clock_buffer_list { CLKCFS2 CLKBFS4 CLKBFS8 }
slkfix -setup -all
```

Aside from the versatile options offered by Tweaker-C1, there are four additional fixing strategies, each with their own purpose. Each strategy is activated with the `slk_ftc_setup_eco_target` and `slk_ftc_hold_eco_targert` variables and the following values:

- wns: Minimizes the worst slack
- tns: Minimizes the total slack value
- nfe: Reduces the number of negative failing endpoint
- lwns: Minimizes wns and tns in the same time

During setup clock ECO, you choose to target fixing on the TNS and especially on the WNS, since fixing WNS with useful skew helps to speed up chip performance. Through hold clock ECO, on the other hand, you can minimize the number of hold violations by making small adjustments to the skew, which then reduces the number of delay insertions and minimizes the power and area overhead. Fortunately, Tweaker offers both of these setup and hold clock ECO strategies. The descriptions of the these features are in the Available Features for Timing Fix topic. Refer to the following directories for example scripts of these strategies:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/ftc/c1

To verify results after fixing, you can output a partial SPEF from Tweaker-C1 for STA tools to back-annotate and analyze the STA what-if results before continuing with the ECO. If cell sizing is the only applied fixing action, providing an eco.tcl file for PrimeTime is most ideal for STA what-if analysis. This method simply requires you to restore the PrimeTime session, source the eco.sta.tcl file, then update_timing. The results of this method are very reliable, since the RC network is left almost unchanged. The command for dumping an eco.tcl file in PrimeTime format is:

```
ecotclout -pt pt_tcl_file_name
```

Metal ECO With Tweaker-M1

With an active Metal ECO (Tweaker-M1) license, Tweaker approaches all timing fix jobs with the metal ECO (post-mask ECO) mode. Since the free space attribute from Tweaker-T1 is not available in Tweaker-M1, you must instead use spare cells or gate array

cells as resources for sized up instances or buffer insertion. To prepare the metal ECO environment, use the following the procedure:

1. Enter metal ECO mode by switching on the `metal_eco_mode` variable. This tells Tweaker-M1 to use only spare cells or gate array cells for ECO activities

   ```
   set metal_eco_mode true
   ```

2. Identify spare cells or gate array cells as resources for timing fixing with either of the following methods

   - Specify the name of the spare modules, and Tweaker-M1 will treat all cells under the specified module as spare cells

     ```
     eco -spare_module module_name
     ```

   - Specify the naming patterns of the spare cell instances, then Tweaker-M1 will treat all matched instances as spare cells

     ```
     eco -spare_inst inst_pattern
     ```

   - Specify the list of instances to be reused as spare cells

     ```
     eco -spare_file filename
     ```

   - Automatic recognition. Tweaker-M1 identifies a cell as a spare cell if its input is tied high/ low and output is floating

     ```
     eco -spare_inst -auto
     ```

3. Specify gate array resources' body and available cell type separately

   ```
   set_gate_array_cell -body DCAP* -cell GA*
   ```

   All physical cells in the DEF with names matching DCAP* are treated as gate array bodies, while all cell names matching GA* are the gate array cells that are available for use.

4. Non-functioning cells with floating outputs can easily be collected and recycled to the SPARE cone for later Metal ECO use

   ```
   recycle_floating_cones
   ```

   This command first collects a list of the cells with floating outputs, then back-traces all the way up each cell's corresponding logic cones until reaching a cell with at least one output pin connected to logic gates that are relevant to the design's functions. This back-tracing feature is called recycle_floating_cones.

With the previous settings, Tweaker-M1 is capable of recognizing all available spare cells and gate array cells. You can highlight and see the distribution of these spare and gate

array cells in the GUI at the Physical View, or reference the log file for a summary of the cells.

Tweaker-M1 offers several useful options that are best to set properly in advance. For example, you can choose to keep the spare cells in the netlist without connecting or removing them entirely even after some of the spare cells are used. The setting of this variable depends on your ECO routing flow as shown here:

```
set keep_used_spare_cell_at_netlist true|false
```

Apply timing fixes under metal ECO mode is essentially the same as in regular timing ECO. Note that the timing fix constraints must be more generous under the metal ECO mode, as resources are not as readily available compared to the full layer ECO. The following are some examples of the variable settings:

Sizing-related settings:

```
set slk_auto_sizing_max_shift_distance 200
set slk_auto_sizing_max_fanout_limit 64
set slk_auto_fix_max_wire_length_limit 100
```

Buffer insertion-related settings:

```
set slk_give_up_insert_if_no_space true
set slk_give_up_insert_buf_distance 800
```

Refer to the following directory for the Metal ECO example script templates:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_m1/fix.tcl*

Leakage Power ECO With Tweaker-P1

For Power ECO, Tweaker-P1 honors the mapping rules to swap low VT cells to high VT cells, or size cells down in efforts to save leakage power. Due to their weak driving strength, cells with high voltage thresholds are bad for setup timing. So Tweaker-P1 swaps as many cells as possible while relying on the TWF update system to keep the setup timing intact. To maintain setup timing through multiple scenarios, you can establish each scenario with “begin_corner” and “end_corner” pairs, similar to the MMMC structure.

For Power ECO, Tweaker-P1 analyzes whole chip as its ECO Domain, so it does not need slack reports and it only counts on the signoff quality TWF to maintainthe timing changes of each pin. To output TWF’s in preparation for Power ECO, see The Timing Window File: Definition and Role for more detailed usage. Each Tweaker-P1 scenario contents use the following format:

```
begin_corner scenario_name
set_group -lib -name library_group
set_group -spef -name RC_group
set_group -sdf -name sdf_group
```

```
twfin -analysis_type on_chip_variation TWF_file
end_corner scenario_name
```

The required files are essentially identical to the files for normal timing fixes, with the only difference in that slack reports are not mandatory for Power ECO. However, there are variables exclusive to, or closely related to, Power ECO:

- `slk_leakage_power_lib`

  For advanced process nodes, such as 40 nm and below, some slow library corners might behave differently with other library corners in regards to leakage power. For example, a corner with HVT (high VT) cells consumes more leakage power than RVT (regular VT) cells. This variable enables Tweaker-P1 to evaluate the leakage power with the correct library group, while simultaneously watching timing impact on another slow library group.

  For example:

  ```
  set slk_leakage_power_lib most_leak
  ```

  where *most_leak* is the name of the library group specified by the `libin -name` command. If the variable is not set, Tweaker-P1 uses the first library group for leakage power evaluation, by default.

- `slk_power_eco_target_slk`

  This variable defines the candidate cells to be swapped or sized.

  For example:

  ```
  set slk_power_eco_target_slk 0.05
  ```

  defines instances with setup slack larger than 0.05 will be the candidates to be swapped or sized.

- `slk_setup_target_slk`

  This variable defines the worst slack that Tweaker-P1 can maintain.

  For example:

  ```
  set slk_setup_target_slk 0.05
  ```

  Note:

  To avoid confusion, you should keep the power ECO and setup target slack values the same.

- `slk_auto_fix_max_wire_length_limit` and `slk_auto_sizing_max_fanout_limit`

  VT swapping does not change cell size, and likely does not change the original routing. Therefore, the settings can be less restrictive to allow more opportunities to swap.

VT Swap example:

```
set slk_auto_sizing_enable_cell_mapping true
set slk_cell_mapping_rules { * *HVT }
set slk_auto_fix_max_wire_length_limit 9999
set slk_auto_sizing_max_fanout_limit 999
...
slkfix -power_eco
```

If you try to size cells down even further to save more leakage, this causes changes in the pin placement of each cell. Therefore, you should apply a conservative maximum wire length limit and maximum fan-out limit as their sizing down settings.

Sizing down example:

```
set slk_auto_sizing_enable_cell_mapping true
set slk_cell_mapping_rules_regexp { @D[0-9]+@  @D[0-9]+@}
set slk_auto_fix_max_wire_length_limit 450
set slk_auto_sizing_max_fanout_limit 32
...
slkfix -power_eco
```

Multiple rules can be set at the same time.

```
set slk_cell_mapping_rule {*LVT *RVT *HVT: *RVT *HVT}
```

If the mapping rules are complicated, you can use the `slk_cell_mapping_rule_regexp` variable. The `set slk_cell_mapping_rule_regexp {@LVT @SVT @HVT : @SVT @HVT}` syntax is equivalent to `set slk_cell_mapping_rule {*LVT *SVT *HVT : *SVT *HVT}`, with the only difference in that the `slk_cell_mapping_rule_regexp` variable supports regular expressions defined in the mapping rules. For example,

```
set slk_cell_mapping_rule_regexp { @[0-9]+@ @[0-9]+@ }
```

allows mapping only to the cells with the same base name. However, Tweaker-P1 only honors one of either mapping rules in the variable definition. If two different mapping rules are set, the later setting overrides the previous setting. For more details about each variable, in the GUI command window, enter either the `man slk_cell_mapping_rule` or `man slk_cell_mapping_rule_regexp` command.

Tweaker-P1 provides two flows for optimizing leakage power. The first flow is developed when you prioritize the tapeout schedule over the amount of reported leakage power. The flow applies relatively more conservative VT swapping for minimizing setup time impact, which shortens the time needed to fix any setup violations caused by Power ECO.

The second flow, which uses more aggressive swapping, is developed for projects in which meeting leakage power specifications is the highest priority. This flow enables swapping on flip-flops and disables other aggressive options. By ignoring potential timing impacts caused by the change of flip-flops, the flow reaches nearly exhaustive swapping

Feedback

and inevitably induces some timing impact. Therefore, a “setup recovery” flow allows you to apply an extra ECO iteration for timing recovery.

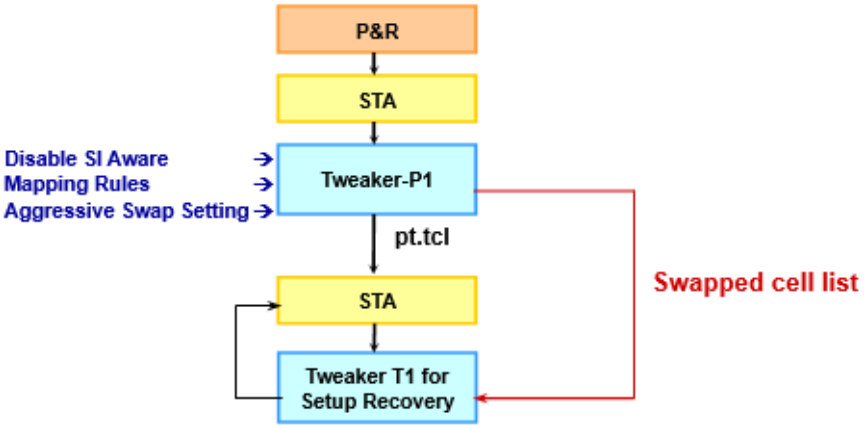
For example:

```
set slk_auto_sizing_comb_logic_cell_only false
```

Enables swapping FF’s

When Tweaker-P1 finishes VT swapping and sizing cells down, you can output an eco.tcl file to a STA (for PrimeTime by default), to verify the results and then use Tweaker-T1 to recover any newly induced setup time. Setup time recovery can be achieved in just one ECO iteration since Tweaker-T1 applies cell sizing based on the cell swapped list file that is output from previous Tweaker-P1. The file containing all swapped instances is named “*tweaker.peco.list.xxx*”, by default. Use the `set slk_power_eco_swap_list_filename` *`filename`* command to override the default setting.

Figure 48 Power ECO flow



To start Tweaker-P1, use the `slkfix -power_eco` command after the design data and timing scenarios are completely loaded in. The example script templates for Power ECO are in the following directories:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/power_eco/power_eco.tcl

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/power_eco/fix_power_eco_setting.tcl

The templates for setup time recovery are under the following directory:

$Tweaker_Install_Dir/dorado/etc/template/tweaker_t1/recover

Feedback

Boundary Leakage Power

In advanced designs, cell context effect from neighboring cells can have significant impact on cell performance, which causes a certain amount of leakage power in addition to cell leakage power.

Boundary leakage power, also known as context leakage power, is caused by the physical location of the cells after placement is completed in the implementation stage.

You can calculate the boundary leakage power by setting the boundary leakage power calculation variable to true and specifying context leakage data. This must be done before running the `slkdc -check_slack_consistency` command.

Set the `enable_boundary_leakage_power_calculation` variable to `true` (the default is false) to enable the boundary leakage power feature.

```
set enable_boundary_leakage_power_calculation true
```

The `read_context_leakage_data` command reads boundary leakage power information from library cell-based source and drain side files and boundary leakage files. The following is the syntax:

```
tweaker_shell> read_context_leakage_data -boundary_leakage_files \
{leakage_mapping_file leakage_probability_file} \
-libcell_sdside_files {source_drain.txt}
```

The `-boundary_leakage_files` option reads the boundary leakage table mapped by the VT group and the leakage probability file.

The `-libcell_sdside_file` option reads the library-based source and drain, and VT group information. The library cell source and drain information side file defines the internal fin type from the left boundary (the `left_row_fin_type` attribute) and from the right boundary (the `right_row_fin_type` attribute). Use the `left_vt_category` and `right_vt_category` attributes to specify the VT group information.

Name	Description
`cellName`	The standard call library cell name
`left_VT` and `right_VT`	The standard library cell used to look up the leakage side file
`left_row_fin_type` and `right_row_fin_type`	The pair of the source, drain or filler, and the fin layer number from origin to top
`left_depth`and `right_depth`	The filler layer number. No depth information is required for source and drain.

Feedback

Run the `slkreport -power` command to check the boundary leakage power, including the maximum, minimum, and expected boundary leakage power for the design.

```
[ CMD ] slkreport -power
# leakage power: lib(ss)
#-------------------------------------------------------------
#       Clock Network:   0.000000 ->   0.000000 mW (0.0%)
#    Sequential Cells:   0.006064 ->   0.002000 mW (67.0%)
# Combinational Cells:   0.008469 ->   0.002770 mW (67.3%)
#-------------------------------------------------------------
#                        0.014533 ->   0.004769 mW (67.2%)
# ...
#-------------------------------------------------------------
# Maximum Boundary Leakage Power:  1.237e-04   ->  1.1 mW
# Minimum Boundary Leakage Power:  3.400e-05   ->  1.2 mW
# Expected Boundary Leakage Power: 5.641e-05   ->  2.5 mW
#-------------------------------------------------------------
```

JSR Aware Power ECO

The joint success rate (JSR) aware power ECO watches the ECO margins, including the setup and hold JSR margins from the TWF file to improve violations. After enabling this feature, running the `slkfix -power_eco` command pushes whole paths in the design to critical paths and positive slack areas. This provides additional space to swap cells to reduce power.

The following topics are covered:

- Enabling the JSR Aware Power ECO
- JSR Aware Power ECO Report Script Example

Enabling the JSR Aware Power ECO

To enable JSR aware power ECO, perform the following steps:

1. Source your Tcl file, containing the setup design data variation information.
2. Build the unit RC table, if you do not have an existing RC table:

   ```
   build_unit_rc_table -slack_domain
   slkfix -create_power_eco_domain
   ```

3. Create the ECO domains and update the TWF file with the JSR margins. For example:

   ```
   slkdc -check_slack_consistency
   slkdb -update_twf_by_path -sync
   slkfix -design_list {SUB1 SUB2 SUB3 SUB4 SUB5 SUB6 SUB7 SUB8 TOP}
   ```

Feedback

4. Set the VT cell for the VT ratio results. For example:

```
set slk_vt_cell_naming {*LVT *HVT *SVT}
set slk_power_eco_swap_list_filename ./tweaker.peco.rpt
```

5. Set the power ECO setting for VT swap. For example:

```
source $script_path/fix_power_eco_setting.1.tcl
set slk_cell_mapping_rul_regexp { @LVT @SVT @HVT : @SVT @HVT }
set_drv_factor 0.8
set_drv_factor 0.7 -cell *HVT*
slkfix -power_eco
```

6. Set the power ECO setting for sizing down. For example:

```
source $script_path/fix_power_eco_setting.2.tcl
set slk_cell_mapping_rule_regexp{ @D[0-9]+@ @D[0-9]+@ : @M[0-9]+@
 @M[0-9]+@ }
set_drv_factor0.7set_drv_factor0.6 -cell *HVT*
slkfix-power_eco
```

JSR Aware Power ECO Report Script Example

Example 56 shows an example of a JSR aware power ECO report script. You must set the `dva_enable_positive_slack_reporting` variable to `true` (the default is `false`).

Example 56 JSR Aware Power ECO Report Script Example

```
set dva_enable_positive_slack_reporting true | false

set hold_variation_args" -pocv_pruning -path_typefull_clock_expanded
 -pba_mode
 $args_pba_mode-delay_typemin -nworst$args_nworst-max_paths
 $args_max_path-slack_lesser_than$args_slack_threshold -cover_design"

set hold_variation_paths [eval get_timing_paths $hold_variation_args]
set dv [get_design_variation -sample_size 1000000 $hold_variation_paths]

report_timing $hold_variation_paths -include_hierarchical_pins
 -input_pins \
   -significant_digits 6 -variation
```

Reliability ECO With Tweaker-R1

The following are covered in this topic:

- IR-Drop ECO
- IR Auto ECO
- Vmin ECO

Feedback

- Cell Robustness ECO
- Voltage Robustness ECO
- Forbidden Chain ECO

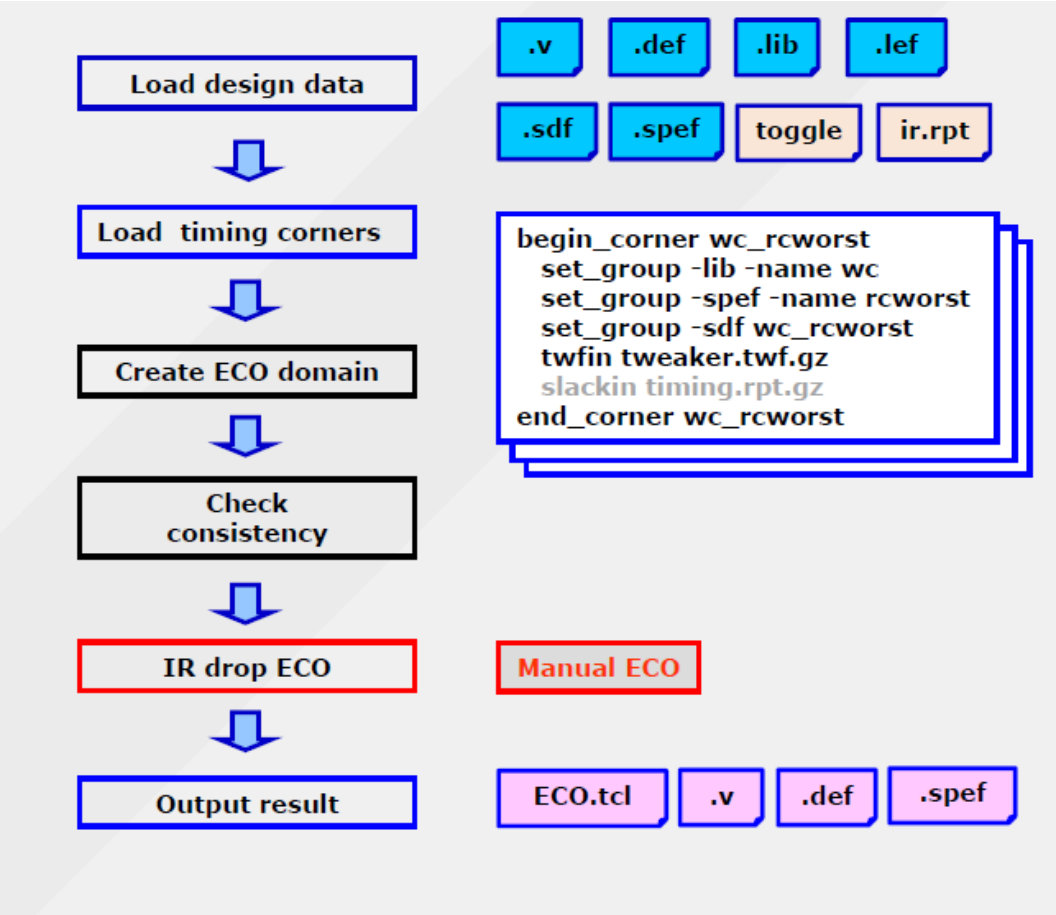
IR-Drop ECO

After proper power planning, in some set of instances you might observe a high voltage drop inside the chip. Instances with a high voltage drop need a high margin (setup/hold) as a guard band to operate. If the voltage drop is beyond the certain limit, then it must be corrected either by ECO methodology of cell transform or adding an extra power strap to supply additional power to the respective region. Tweaker-R1 has an effective reliability solution for IR drop-reductions. The following are the steps for the IR-drop ECO flow:

1. Load the design data
2. Load the timing corners
3. Create the ECO domain
4. Check the consistency
5. IR-drop ECO
6. Output results

Feedback

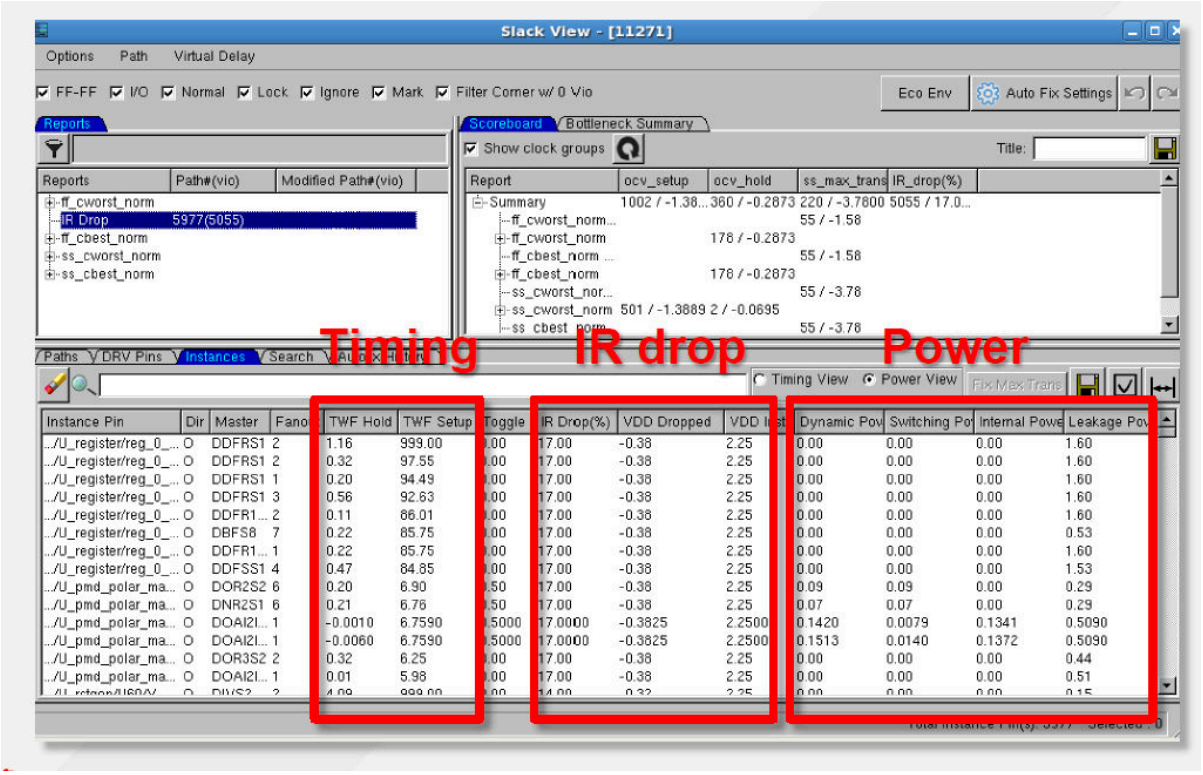
Figure 49 The IR-Drop ECO Flow



You can view IR-drop violations at each path in the Slack View, as shown in the following figure.

Feedback

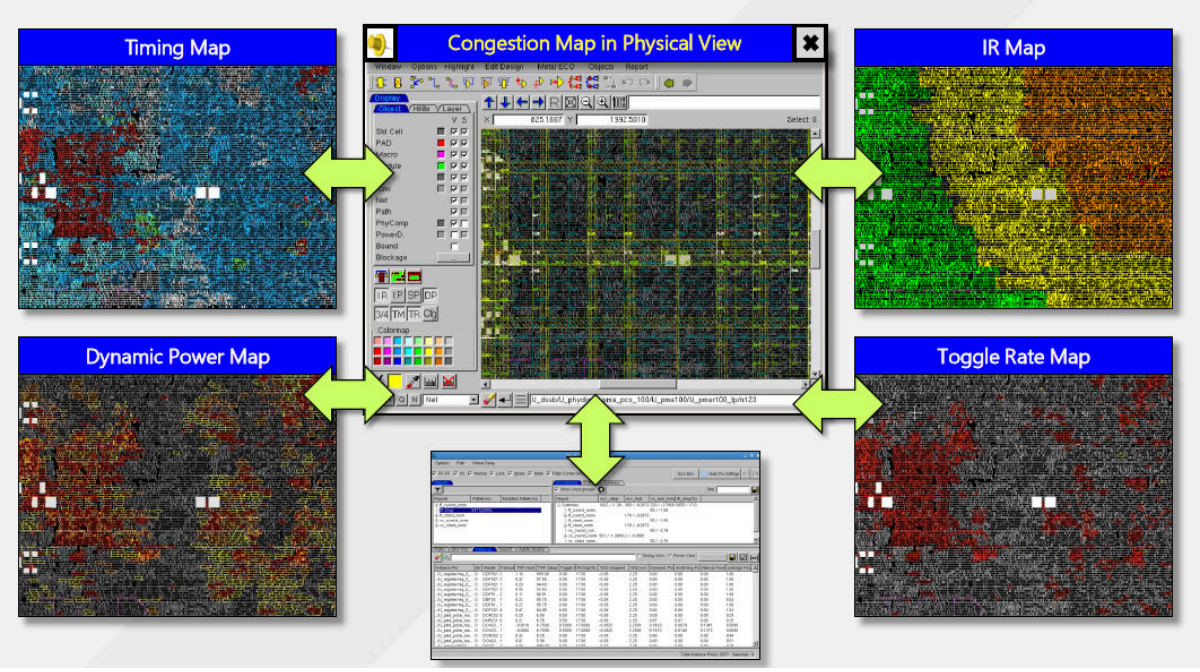
Figure 50 IR-Drop Violations in the Slack View



The following figure shows the GUI information for IR-drop manual ECO. Manual ECO updates timing, power, and congestion concurrently and rapidly.

Feedback

Figure 51 GUI Information for IR-Drop Manual ECO



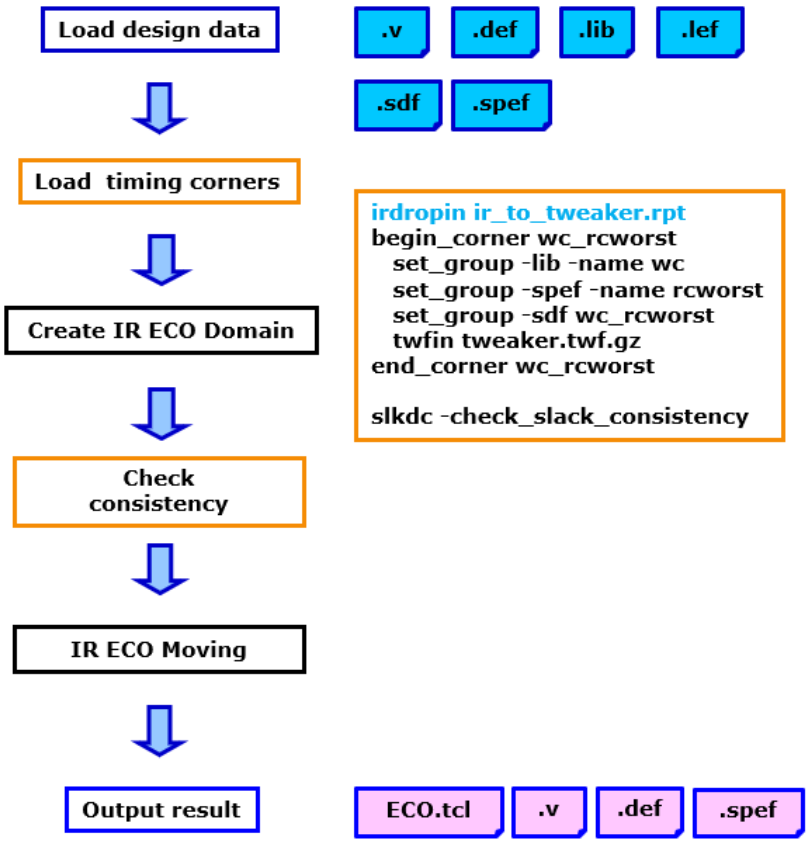
IR Auto ECO

IR auto ECO supports the incremental update engine in aggressor/victim cells, supports aggressor information to adjust the IR ECO cost, and enhances the IR ECO autofixing summary type. Tweaker uses its unique timing and powerful ECO capabilities in IR auto ECO, which include an ECO domain concept with a fast timing incremental update engine, physical awareness ECO placement engine, and an interactive GUI interface. The following are the steps for the IR auto ECO flow:

1. Load the design data
2. Load the timing scenarios
   - Load the IR dropped information report
3. Create the IR ECO Domain
   - Define IR dropped violations
   - Start IR dropped analyzer by RH-SC
4. Check the consistency
5. IR ECO by Sizing
6. Output the results

Feedback

Figure 52 The IR Auto ECO Flow



To use this feature, you must create the IR ECO domain, launch the incremental IR update (RH-SC), and set IR autofixing. For example:

```
# Create the IR ECO Domain
irdropin ./ir_to_tweaker.rpt
slkfix -create_ir_drop_eco_domain -percentage 12 -guardband 20
…
slkdc -check_slack_consistency
…
# Launch the RHSC
set ir_eco_rhsc_tool_path "/tool/rhsc_version_1/bin"
set ir_eco_rhsc_work_dir_path "/home/user01/rhsc_run/my_log"
set ir_eco_rhsc_restore_db_path "/home/user01/db"
set ir_eco_rhsc_timing_corner "ss_cworst"
set ir_eco_rhsc_voltage_impact_view_name vim
set ir_eco_rhsc_analysis_view_name av
set ir_eco_rhsc_aggressor_view_name aggr
begin_ir_drop_analysis ; # Launches the RHSC for IR dropped prediction
…
```

Feedback

```
# Set the IR autofix setting
set slk_fix_ir_eco_by_sizing true
set slk_ir_eco_min_improved_ratio 0.0001
set slk_ir_eco_target_dropped_ratio 0.12
slkdb -ir_eco
…
end_ir_drop_analysis ; # Closess RHSC IR dropped prediction
…
```

Vmin ECO

Vmin is defined as the minimum Vdd in which a design can have the same performance as in nominal Vdd. Vmin ECO fixes Vmin violated paths in the PrimeShield Vmin report to improve the timing margin. Vmin ECO recovers the voltage robustness of a design.

Vmin ECO estimates the timing path slack based on voltage slack and Vdd sensitivity slack. Vmin ECO sizes up cells to gain additional timing slack and improve the voltage slack. Cells that have lower cell voltage sensitivity are chosen to keep the correct trend to improve timing.

The Vmin ECO watches mutli-mode multi-corner (MMMC) timing. Setup time is fixed only if the voltage sensitivity of the instanced is reduced. Degradation of the voltage sensitivity is not allowed, even if timing is improved. Only the sizing method is supported. Vmin ECO is recommended on combinational cells only.

The following are the steps for the Vmin ECO flow:

1. Load the design data

   Scale library for voltage sensitivity

2. Load timing corners

   Create additional corner for voltage sensitivity

3. Create the ECO domain
4. Check the consistency
5. Perform Vmin ECO
6. Output the results

Feedback

Figure 53 The Vmin ECO Flow

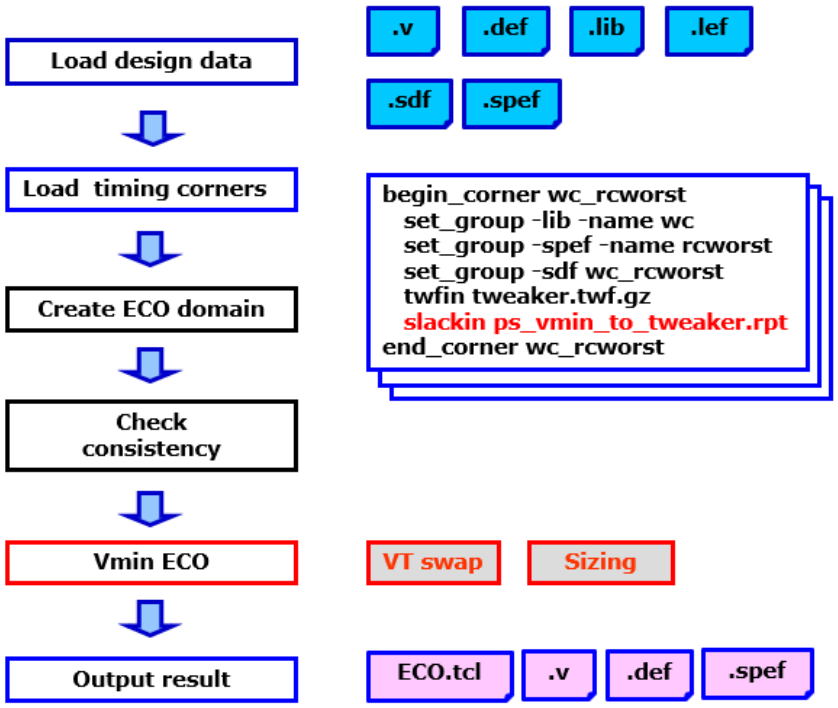


Table 12 provides Vmin terms and its definition.

Table 12 Vmin Terms

Term	Unit	Definition
voltage slack	V	The voltage (Vdd) margin of the path. When the Vdd drops the voltage slack, the timing slack of the path drops to zero.
v_sensit	ps	The timing delta of the instance between the nominal Vdd and the 10mV-dropped Vdd. Instance-based sensitivity depends on the cell type, input transition time, and output loading.
slack_vdd_sensitivity	ps/V	The timing-to-voltage sensitivity of the path.

Before you can enable Vmin ECO, you must scale the library for voltage sensitivity, create an additional corner for voltage sensitivity, pair the report corner and the reference corner, and set the voltage slack target.

Feedback

Use the `scale_lib` command to prepare additional library groups to view voltage sensitivity. For example,

```
scale_lib -voltage 0.80 -ref_group { lib1 lib2 } -name lib1_drop_10mV
```

When you load the timing corners, you need to create an additional corner to view voltage sensitivity. For example,

```
begin_corner corn1_drop_10mV
  set_group -lib  -name lib1_drop_10mV      ; # 10mV drop lib
  set_group -spef -name rc_same_as_corn1    ; # use the same RC in
                                              Vdd_nom
end_corner corn1_drop_10mV
```

Use the `set_voltage_sensitivity_corner` command to pair the report corner and the reference corner. For example,

```
set_voltage_sensitivity_corner -base corn1 -ref corn1_drop_10mV
```

To set the Vmin ECO target in the voltage slack, set the value of the `voltage_slack_target` variable. When you adjust the voltage slack target, the path timing shifts concurrently. Only paths that have slack Vdd sensitivity and voltage slack are shifted. It might shift the timing slack from positive to negative in the setup time. In general, the timing critical path is the ECO. It is hard to enhance the timing of the path. To avoid unnecessary power consumption, it is reasonable to align the Vmin ECO target to the voltage slack of the timing critical path. Vmin ECO can enhance those paths having better timing than the critical path and worse voltage slack.

```
set voltage_slack_target value    ; # (unit:V)
```

To perform Vmin ECO, use the following command:

```
[CMD] slkfix -vmin [-path_list $collection]
```

Example 57 shows an example of a Vmin ECO autofix log file.

Example 57 Vmin ECO Autofix Log File

```
#Begin Fix Vmin Setup Time(Sizing by footprint)...
#-----------------------------------------------------
#           Voltage Slack Target: 0.060
#                  Setup Target: 0.003
#                   Hold Target: 0.000
#                Adjust Placement: false
#                  Watch Hold TWF: false
#                 Watch Setup TWF: true
#  Setup TWF Impact(Accumulated): 0.001
#              Fit To Free Space: true
#  Max Shift Distance For Sizing: 10.000 micron
#   Auto Sizing Max Fanout Limit: 32
# Auto Fix Max Wire Length Limit: 450 micron
```

Feedback

```
#                      Ignore DRV: false
#-----------------------------------------------------
......

#End Fix Vmin Setup Time(Sizing by footprint)
#-----------------------------------------------------
#         grow up area: 253.440
#         sizing count: 20
# total moved distance: 40.10 micron
#-----------------------------------------------------
......

#-----------------------------------------------------
# Vmin summary: (voltage slack target = 0.060 V)
#            Vmin vio paths: (     5 ->      4)
#       worst voltage slack: (-0.780 -> -0.081)
#-----------------------------------------------------
#         voltage slack (V):   original    current
#           -0.780 ~ -0.770           1          0
#           -0.770 ~ -0.760           0          0
#           -0.760 ~ -0.750           0          0
......
```

To view the Vmin violation summary, run the `slkreport -vmin -resolution` command to report the histogram. The resolution value specifies the voltage interval for the histogram report. The decimal should end with a 1 or a 5 (for example, 0.01, 0.005, 0.001). The default is 0.01. For example,

```
slkreport -vmin -resolution 0.01
```

Example 58 shows an example of a Vmin violation summary report.

Example 58 Vmin Violation Summary Report

```
#-----------------------------------------------------
# Vmin summary: (voltage slack target = 0.030 V)
#        Vmin vio paths: (    1234 -> 56      )
#   worst voltage slack: (  -0.123 -> 0.027   )
#-----------------------------------------------------
# voltage slack (V):   original    current
#     -0.123 ~ -0.120    1           0
#     -0.120 ~ -0.110    10          0
#     -0.110 ~ -0.100    20          0
#     -0.100 ~ -0.090    10          0
#     -0.090 ~ -0.080    20          0
#     -0.080 ~ -0.070    10          0
#     -0.070 ~ -0.060    0           0
#     -0.060 ~ -0.050    10          0
#     -0.050 ~ -0.040    30          0
#     -0.040 ~ -0.030    20          0
#     -0.030 ~ -0.020    10          0
#     -0.020 ~ -0.010    520         0
#     -0.010 ~ -0.000    510         0
```

Feedback

```
#       0.000 ~ 0.010    30          1
#       0.010 ~ 0.020    30          5
#       0.020 ~ 0.030    3           50
#      target = 0.030 ----------------------------
#       0.030 ~ 0.040    0           200
#       0.040 ~ 0.050    0           150
#       0.050 ~ 0.060    0           230
#       0.060 ~ 0.070    0           598
#-------------------------------------------------
# total path:            1234        1234
#-------------------------------------------------
```

Example 59 is an example of a Vmin ECO script.

Example 59 Vmin ECO Script

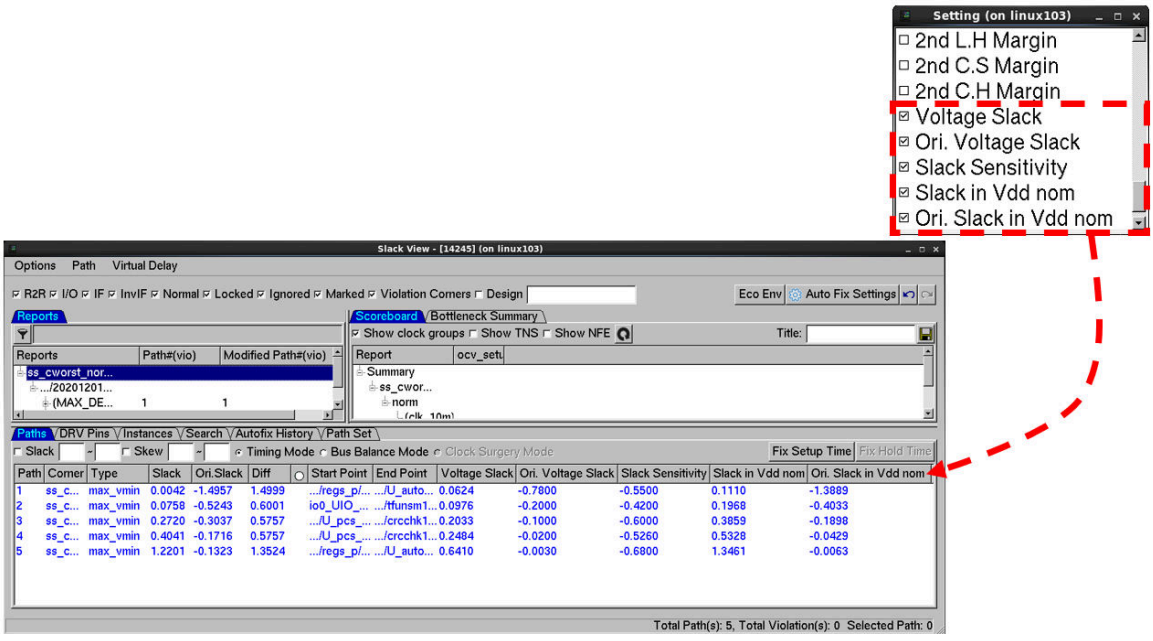
```
source load_lib_lef.tcl

scale_lib -voltage $lib1_Vnom_drop_10mv -ref_group { lib1 lib2 } \
            -name lib1_drop_10mv

source read_netlist.tcl
source read_def.tcl
......

begin_corner corn1
  set_group -lib  -name lib1
  set_group -spef -name corn1_spef1
  set_group -sdf  -name corn1_spef1_lib1
  set_group -pocv -name spef1_lib1
  ......  ; # derating settings
  twfin -analysis_type on_chip_variation design.twf.gz
  slackin ps_vmin_to_tweaker.rpt.gz
  ......
end_corner corn1

begin_corner corn1_drop_10mv
  set_group -lib -name lib1_drop_10mv       ; # 10mV drop lib1
  set_group -spef -name corn1_spef1         ; # RC is the same as corn1
end_corner corn1_drop_10mv

source build_other_corner.tcl

build_unit_rc_table -slack_domain
......
#### Voltage sensitivity reference corner ####
set_voltage_sensitivity_corner -base corn1 -ref corn1_drop_10mv
......
slkdc -check_slack_consistency
......
##################################################################
## Fix Vmin by vtswap & sizing
##################################################################
```

Feedback

```
set voltage_slack_target 0.060        ; # unit: V

source $script_path/fix_vmin_setting.vtswap.tcl    ; # VTSWAP
set slk_cell_extended_mapping_rule_regexp  { @HVT @ }
set slk_setup_target_slk 0.003
slkfix -vmin

source $script_path/fix_vmin_setting.1.tcl         ; # SIZING w/o MOVING
set slk_cell_mapping_rule_regexp { @S[0-9]+@ @S[0-9]+@ : @S[0-9]+
 @S[0-9]+ }
set slk_setup_target_slk 0.003
slkfix -vmin

source $script_path/fix_vmin_setting.2.tcl         ; # SIZING w/ MOVING
set slk_cell_mapping_rule_regexp { @S[0-9]+@ @S[0-9]+@ : @S[0-9]+
 @S[0-9]+ }
set slk_setup_target_slk 0.003
slkfix -vmin

#### OUTPUT ####
slkreport -vmin -resolution 0.100 > ${output_path}/post_vmin.rpt
source output.tcl
```

In the Slack View window of the GUI, you can view the path attributes. Vmin paths are indicated as “max_vmin” in the Type column. The “Slack” and “Ori. Slack” are shifted from the “Slack in Vdd nom” and “Ori. Slack in Vdd nom”, respectively. See Figure 54.

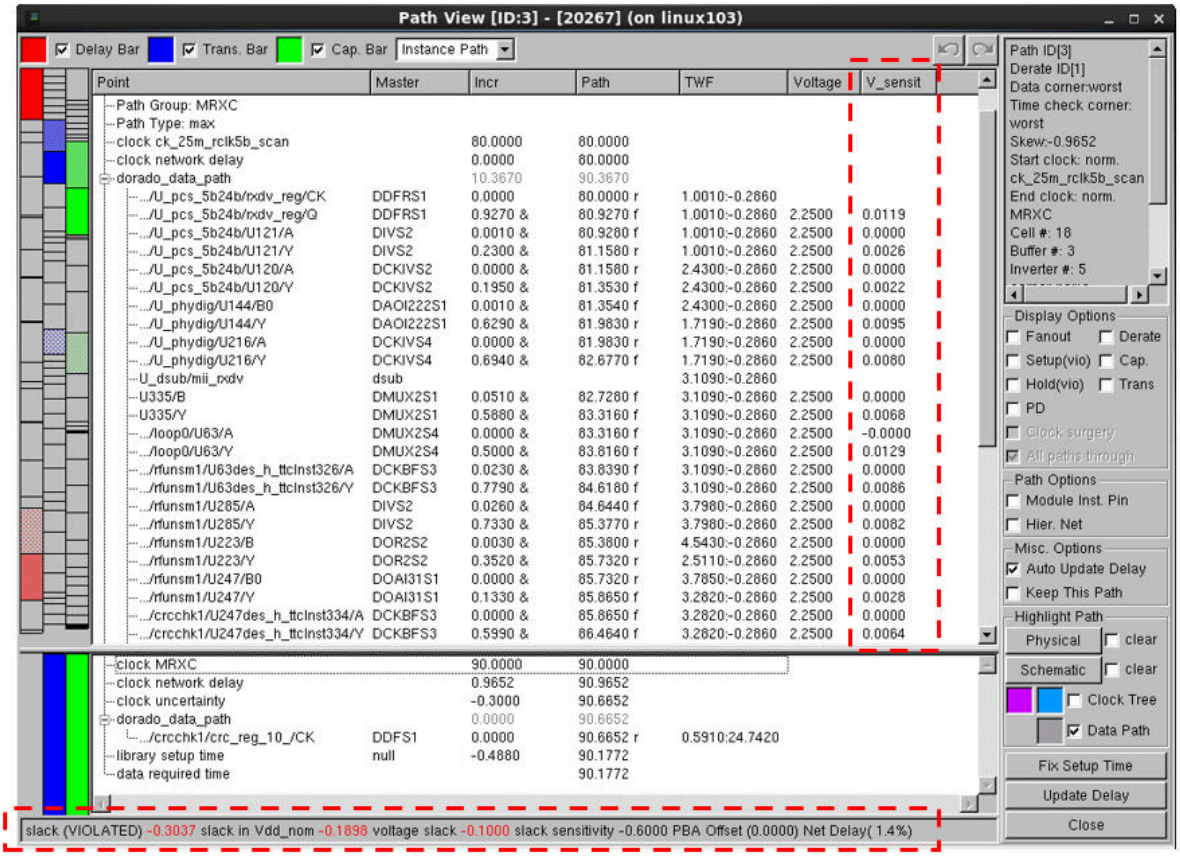
Figure 54 Vmin Path Attributes in the Slack View



Feedback

After performing Vmin ECO, the voltage sensitivity, voltage slacks, and timing slacks are updated, as shown in Figure 55.

Figure 55 Viewing Voltage Sensitivity, Voltage Slacks, and Timing Slack Updates in the Path View



For templates, reference the following directory:

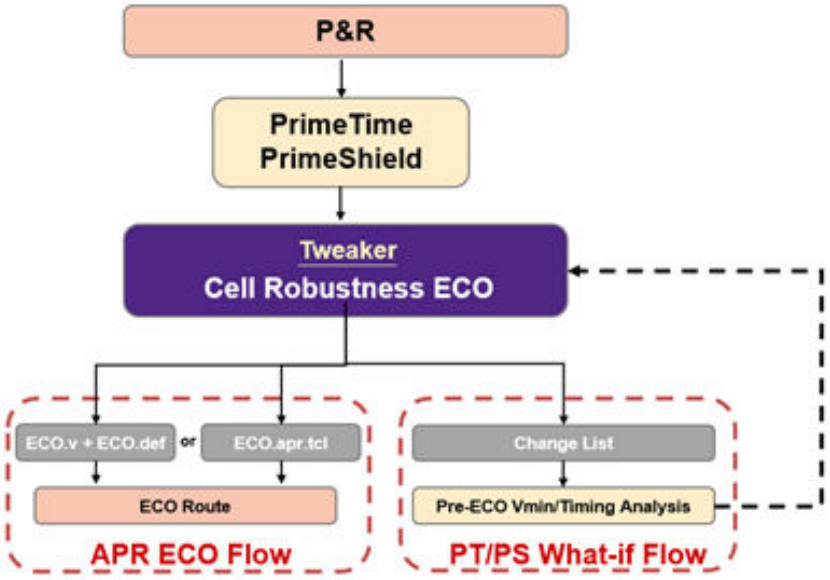
$tweaker_install_path/etc/template/tweaker_t1/vmin

Cell Robustness ECO

The Cell Robustness ECO increases the instance success rate loss to improve the design robustness. Improving the HSFR value can make powerful chip handling with high sigma timing environment without success rate loss. Tweaker parses the cell robustness from the PrimeShield robustness report. With cell robustness ECO, Tweaker watches the multi-corner, multi-mode (MCMM) with physical awareness. From the report, Tweaker calculates the HSFR value, then calculates the HSFR offset between the cell robustness report and Tweaker. The cell robustness ECO autofixes by sizing and VT swap.

Feedback

Figure 56 Cell Robustness Overview



To enable cell robustness by sizing, use the following command:

```
[CMD] slk_fix_cell_robustness_by_sizing
```

To run cell robustness ECO, use the following command:

```
[CMD] slkfix -cell_robustness
```

Voltage Robustness ECO

Voltage robustness analysis identifies cells with large delay changes due to rail drops and inadequate positive slack. Fixing IR timing cell bottlenecks can improve the V-min design. Voltage robustness provides a way to waive IR-drop violators.

Voltage Robustness ECO reads the instance-based voltage robustness report from the PrimeShield tool. You can manually update and fix the voltage robustness violations, as well as autofix the violations.

Voltage robustness identifies cells with large delay change due to a rail drop and inadequate positive slack. The “delay impact” determines the sensitivity of the cell or path to voltage change. Fixing IR timing cell bottlenecks is expected to improve the V-min design (improves design resilience to IR drops). Voltage robustness can waive IR-drop violators.

Feedback

Before you enable voltage robustness ECO, the following are required:

- Enable PrimeShield before running the `report_voltage_robustness` command.

  ```
  [CMD] set_app_var ps_enable_analysis true
  ```

- Set the following variable to `true` before running the `report_voltage_robustness` command.

  ```
  set timing_save_pin_arrival_and_required true
  ```

- Enable the voltage scaling flow for library group of two or more libraries.

  This bypasses robustness checks on cell with the exact library definition matches.

- You can specify the value for voltage shifts.

  The DSLG range should have sufficient coverage for the range specified for the analysis. If it unspecified, the default of 5% of the nominal VDD is used.

To enable fix voltage robustness by sizing, set the `slk_fix_voltage_robustness_by_sizing` variable to `true`. To enable autofixing, run the `slkfix -voltage_robustness` command.

You can perform voltage robustness to build an additional corner for calculating V sensite. Here is an example syntax:

```
source load_lib_lef.tcl

source read_netlist.tcl
source read_def.tcl
......

begin_corner corn1
  set_group -lib  -name lib1
  set_group -spef -name corn1_spef1
  set_group -sdf  -name corn1_spef1_lib1
  set_group -pocv -name spef1_lib1
  ......  ; # derating settings
  twfin -analysis_type on_chip_variation design.twf.gz
  slackin ps_voltage_robustness_with_40mv_voltage_shift_to_tweaker.rpt.gz
  ......
end_corner corn1

begin_corner corn1_drop_40mv                ; # Depends on how much
 voltage drop when dump report
  set_group -lib  -name lib1_drop_40mv -voltage $lib1_Vnom_drop_40mv
 -ref_group { lib1 lib2 } ; # 40mV drop lib1
  set_group -spef -name corn1_spef1         ; # RC is the same as corn1
end_corner corn1_drop_40mv

source build_other_corner.tcl
```

Feedback

```
build_unit_rc_table -slack_domain
......
```

The Voltage Robustness Report

To view the voltage robustness report, use the `report_voltage_robustness` command.

The following are various examples for using the `report_voltage_robustness` command:

- Reports voltage robustness with a 10mV drop

  `[CMD]` **`report_voltage_robustness -voltage_shift 0.01`**

- Reports voltage robustness with a 5% voltage drop. Note that you cannot use the `-voltage_shift` and `-voltage_shift_ratio` options together.

  `[CMD]` **`report_voltage_robustness -voltage_shift 0.05`**

- Reports voltage robustness in PBA mode

  `[CMD]` **`report_voltage_robustness -voltage_shift 0.01 -pba`**

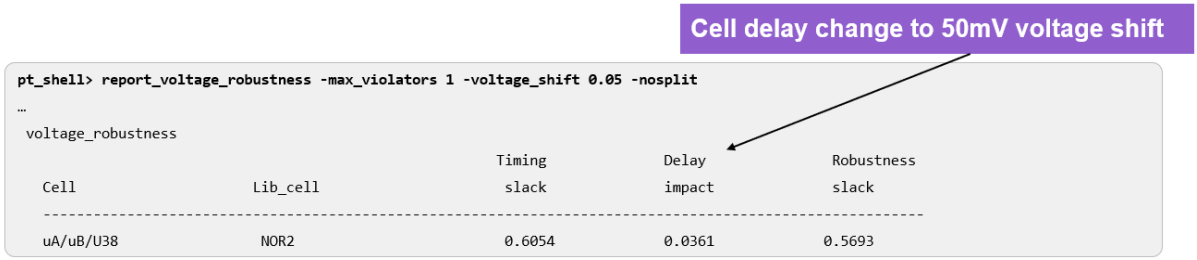
- Reports voltage robustness for cells in reg-to-reg paths. The `-pba` option is required with the `-reg_to_reg` option.

  `[CMD]` **`report_voltage_robustness -voltage_shift 0.01 -pba -reg-to-reg`**

  For more information, see the `report_voltage_robustness` command man page.

The following is an example of a voltage robustness report:

Figure 57 Voltage Robustness Report Example



Voltage robustness reporting should identify cells with large IR impact on timing. Therefore, the reported robustness slack is equal to the delay impact when the initial pin slack is negative. IR bottlenecks are avoided in marginally passing paths to be masked by stages in already violating paths.

Feedback

```
voltage_robustness
                                                  Timing          Delay          Robustness
  Cell                  Lib_cell                   slack           impact          slack
  ------------------------------------------------------------------------------------------
  uA/uB/U38               NOR2                   -3.5523         0.0321          -0.0321
```

Voltage Robustness Error Messages

Voltage robustness analysis requires cells in a linked library to be present in the scaling library group for interpolation. Ensure that:

- all scaling library groups are defined with the `define_scaling_lib_group` command before performing voltage robustness analysis.
- the `-exact_match_only` option is not specified with the `defined_scaling_lib_group` command.

  If the previous criteria are not met, you get the following PYSR-001 error message:

  ```
  Warning: Voltage robustness analysis is not performed because no
   define_scaling_lib_group for interpolation (PYSR-001)
  ```

A PYSR-005 error message appears If the shifted voltage is beyond the library scaling range. For example,

```
Warning: An '0.800000' extrapolation exceeding voltage scaling range
 [0.950000, 0.850000]
has been detected. The accurancy of the voltage robusness may be
 affected.
[Cell: 'U38'] (PYSR-005)
```

Transition Robustness ECO

In a design, there are instances in which transition drops can occur, causing a setup violation. To avoid this from happening, you can use the Transition Robustness ECO to fix cells with a larger delay change to enhance the robustness.

When you enable this feature, in the fixing process, the VT swap and Sizing methods are used to fix any violation instances and on each instance's driving. The tool also watches other timing edge transition robustness slacks to avoid any power violations. When the transition robustness ECO process is complete, the tool outputs the fixing results.

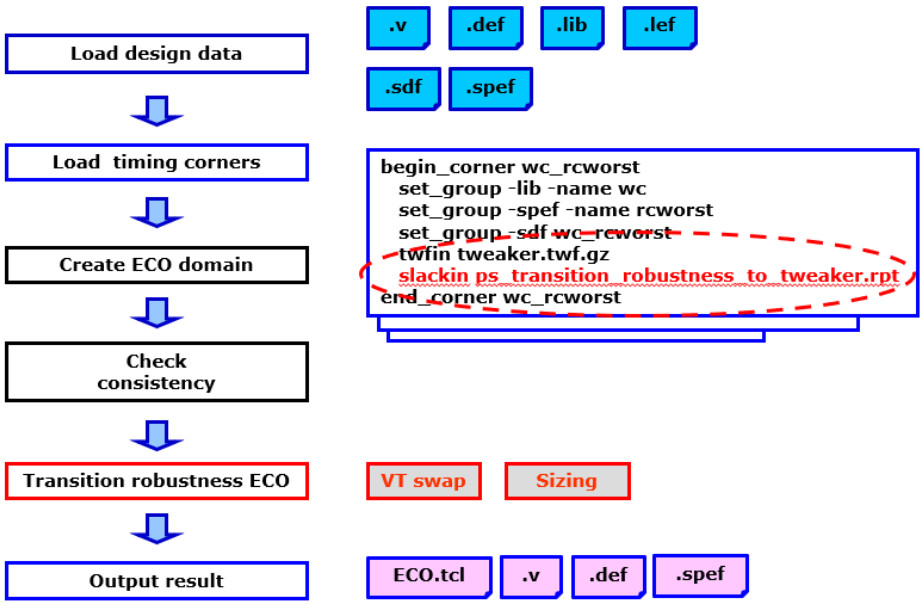
The following are the steps for the Transition Robustness ECO flow:

1. Load the design data
2. Load the timing corners
3. Create the ECO domain

Feedback

4. Check the consistency
5. Enable transition robustness ECO
6. Output the results

Figure 58 The Transition Robustness ECO Flow



When you load the timing corners, you must provide a transition robustness report. To output the transition robustness report, use the following syntax example:

```
restore_session pt_session
set timing_save_pin_arrival_and_required true
set timing_pocvm_enable_analysis true
set ps_enable_analysis true
report_cell_robustness -type transition -tran_shift_ratio 0.05 -nosplit
 -verbose_debug
```

The following is an example a transition robustness report.

Feedback

Figure 59 Example of a Transition Robustness Report

```
DORADO_TIME_UNIT 1e-9
DORADO_CAP_UNIT 1e-15
DORADO_SIGMA 6
****************************************
Report : transition_robustness
    -nosplit
    -transition_shift_ratio 0.050000
    -verbose
Design : d_ifft_512
Version: ...
Date   : ...
****************************************


   transition_robustness

                                                                  New      Timing     Delay      Robustness
   Cell   Lib_cell   From_pin   To_pin                       RF     Delay      delay    slack      impact     slack
   ------------------------------------------------------------------------------------------------------------------------------
   di_ifft_a/norm/HFSINV_173_132413

          P6L11B_INVX4

                      di_ifft_a/norm/HFSINV_173_132413/i

                                di_ifft_a/norm/HFSINV_173_132413/o   rise  0.232333  0.237693  -0.007071  0.004932  -0.004932
```

The Delay Impact column shows the impact on cell delay when the transition is shifted by the specified value. The robustness slack calculation is identical to voltage robustness analysis. If the timing slack is less than 0, the robustness slack is equal to a delay impact. If the timing slack is greater than 0, the robustness slack is equal to the timing slack minus the delay impact. The `report_cell_robustness` command parsers the shift ratio value from the report header, calculates the factor for the Delay and the New delay columns, and parsers the DORADO unit and SIGMA.

To perform the transition robustness ECO, run the following command:

```
[CMD] slkfix -transition_robustness
```

Alternatively, you can use the following command:

```
[CMD] fix_eco_robustness -type {transition_robustness}
          -methods size_cell
```

Forbidden Chain ECO

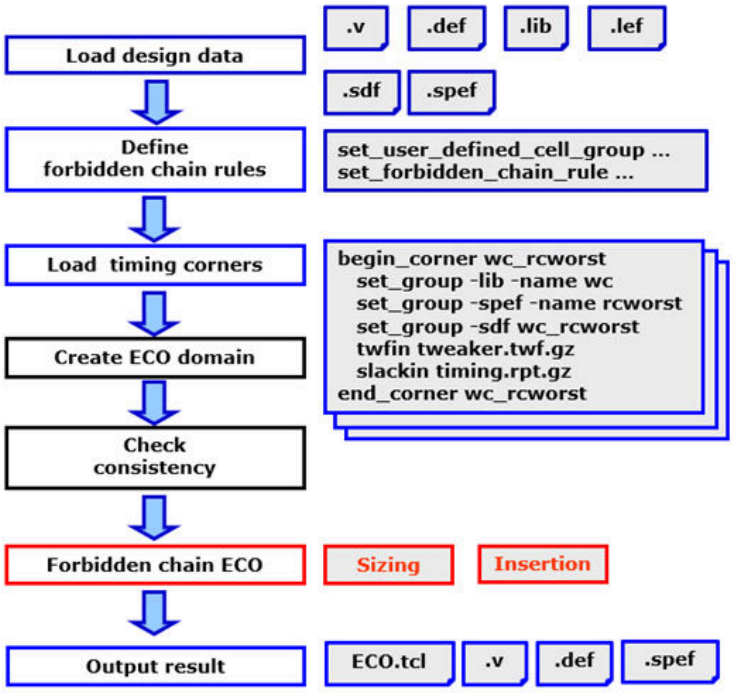
Tweaker-R1 has an effective reliability solution for fixing forbidden chain violations. Forbidden Chain ECO improves the reliability of the design using either the sizing or insertion method to fix any forbidden chain violations. The following are the steps for the Forbidden Chain ECO flow:

1. Load the design data
2. Define the forbidden chain rules
3. Load the timing corners

Feedback

4. Create the ECO domain
5. Check consistency
6. Enable forbidden chain ECO
7. Output the results

Figure 60 The Forbidden Chain ECO Flow



Before you can enable Forbidden chain ECO, you need to define the forbidden chain rules (see the Forbidden Chain Rule Settings topic).

When you create the ECO domain, enable Forbidden chain ECO to include all forbidden chain violation cells by running the following command:

```
[CMD] slkfix -create_forbidden_chain_eco_domain
```

You can create the forbidden chain domain without clock components. To create the Forbidden chain ECO domain with data segment only, use the following command:

```
[CMD] slkfix -create_forbidden_chain_eco_domain -data_segment_only
```

To perform the forbidden chain ECO, run the following command:

```
[CMD] slkfix -forbidden_chain
```

Feedback

By default, this command fixes all forbidden chain rules violations that you defined. To fix a specified forbidden chain rule, use the `-chain_name` option:

```
[CMD] slkfix -forbidden_chain -chain_name chain_name
```

Forbidden Chain Rule Settings

To define the forbidden chain rules, use the following commands:

```
[CMD] set_user_defined_cell_group
[CMD] set_forbidden_chain_rule
```

For example, the follow commands define the forbidden cell group in Power ECO and the forbidden chain rules. All HVT cells are defined in the cell group named “HVT” and the rule is set so that the HVT cells are not chained together.

```
[CMD] set_user_defined_cell_group -group_name HVT -cell *HVT
[CMD] set_forbidden_chain_rule -chain_name HVT_chain \
        -chain_rule {HVT HVT}
```

To relax the forbidden chain rules, you can set the relax condition settings by providing the relax margin and condition option information. If the forbidden chain meets the relax condition constraint, the cell group chain is not forbidden.

Example 60 Relax Margin and Relax Condition Setting Examples

Relax Margin setting:

```
-setup_margin 0.2 -hold_margin 0.2 -slew_factor_margin 0.4
```

Relax Condition setting:

```
-relax_condition  "(@setup>setup_margin && @hold>hold_margin)
|| @slew_factor<slew_factor_margin"
```

The forbidden chain relax condition supported options are:

```
-setup_margin    "float"
-hold_margin     "float"
-slew_value_margin   "float"
-slew_factor_margin  "float"
-cap_vlaue_margin    "float"
-cap_factor_margin   "float"
```

To apply the relax condition settings, you can add the relax condition option in each forbidden chain rule. For example:

```
set_forbidden_chain_rule -chain_name HVT_chain -chain_rule {HVT UHVT HVT}
-setup_margin 0.2 -hold_margin 0.2 -slew_factor_margin 0.4
-relax_condition "(@setup> setup_margin && @hold>hold_margin) ||
 @slew_factor<slew_factor_margin"
```

Feedback

Example 61 Forbidden Chain With Relax Condition Setting

```
########################################
# Cell group setting (essential)
########################################
set_user_defined_cell_group -group_name HVT -cell *HVT
set_user_defined_cell_group -group_name UHVT -cell *UHVT

########################################
# Forbidden Chain with Relax Condition setting (essential)
########################################
set_forbidden_chain_rule -chain_name HVT_chain -chain_rule {HVT HVT} \
-setup_margin 0.1 -hold_margin 0.1 \
-relax_condition "@setup> setup_margin && @hold>hold_margin"

set_forbidden_chain_rule -chain_name HVT_chain -chain_rule {HVT UHVT HVT}
 \
-setup_margin 0.2 -hold_margin 0.2 -slew_factor_margin 0.4 \
-relax_condition "(@setup> setup_margin && @hold>hold_margin) ||
 @slew_factor<slew_factor_margin"
```

Forbidden Chain by Sizing

To fix forbidden chain violations with the sizing method to improve the design reliability, set the following variables:

```
[CMD] set slk_fix_forbidden_chain_by_sizing true
[CMD] set slk_fix_forbidden_chain_by_insertion false
```

Forbidden Chain by Insertion

Use insertion to fix slew relax condition constraints. Tweaker sets different margins for different forbidden rules.

To fix forbidden chain violations with the insertion method, set the following variables:

```
[CMD] set slk_fix_forbidden_chain_by_sizing false
[CMD] set slk_fix_forbidden_chain_by_insertion true
```

Watching the Forbidden Chain

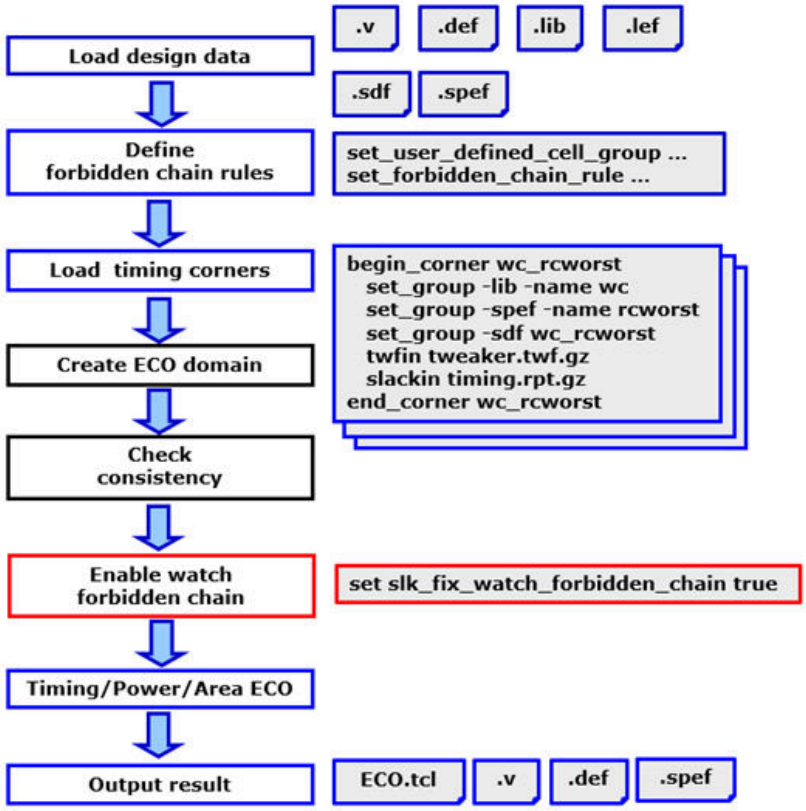
You can watch forbidden chain rules to avoid an unreliable design. The following are the steps for the Watch forbidden chain ECO flow:

1. Load the design
2. Define forbidden chain rules
3. Load the timing corners
4. Create the ECO domain
5. Check the consistency

Feedback

6. Enable the watch forbidden chain
7. Non-forbidden chain ECO
8. Output the results

Figure 61 The Watch Forbidden Chain ECO Flow



To enable watch forbidden chain ECO, set the following variable to `true`:

```
set slk_fix_watch_forbidden_chain true
```

The Forbidden Chain Rules and Violation Reports

To report the forbidden chain rules, use the `report_forbidden_chain_rules` command. To report all rules, use the `-all` option:

```
[CMD] report_forbidden_chain_rules -all
```

To report a specific rule, you can provide the chain name. For example:

```
[CMD] report_forbidden_chain_rules -chain_name s2_x3
```

Feedback

Example 62 Forbidden Chain Rules Report

```
Chain_name     Chain_rule   Relax_condition
s0_x3          S0-S0-S0     (null)
s1_x3          S1-S1-S1     (null)
s2_x3          S2-S2-S2     (null)
s2_x3_setup    S2-S2-S2     @setup>2.25
s2_x3_slew     S2-S2-S2     @slew_value>0.5
```

To report forbidden chain violations in the design, use the `report_forbidden_chain_violation` command. To report all violations, use the `-all` option:

```
[CMD] report_forbidden_chain_violation -all
```

To report a specific rule, you can provide the chain name. For example:

```
[CMD] report_forbidden_chain_rules -chain_name s2_x3_setup_4
```

Example 63 Forbidden Chain Violations Report

```
[CMD] report_forbidden_chain_violation  -chain_name s2_x3_setup_4

Violation report:
Chain name: s2_x3_setup_4
Chain rule definition: S2 - S2 - S2
Relax condition: @setup>4.198

# Library Cell Name       Setup    Pins Name
-------------------------------------------------------------------------
1 (DMUX2S2-DIVS2-DAN2S2) -1.388*  U338/Y - U376/A - U376/Y - U348
2 (DIVS2-DOR2S2-DIVS2)   -0.286*  main0/MACMod/mac0/rfun1/rfunsm1/U285/Y
                                 - main0/MACMod/mac0/rfun1/rfunsm1/U242/B
                                 - main0/MACMod/mac0/rfun1/rfunsm1/U242/Y
                                 - main0/MACMod/mac0/rfun1/rfunsm1/U243/A
3 (DAN4S2-DOR2S2-DIVS2)   2.792*  main0/MACMod/mac0/rfun1/rsizcnt1/U70/Y
                                 - main0/MACMod/mac0/rfun1/rfunsm1/U242/A
                                 - main0/MACMod/mac0/rfun1/rfunsm1/U242/Y
                                 - main0/MACMod/mac0/rfun1/rfunsm1/U243/A
…
…
```

Additional Settings Related to the Forbidden Chain

The following are additional settings related to the forbidden chain.

Feedback

- To watch the forbidden chain rules during autofixing when you perform other ECOs, set the following variable to `true` (the default is `false`):

  ```
  [CMD] set slk_fix_watch_forbidden_chain true
  ```

- To watch the setup and hold timing window when you perform forbidden chain ECO, set the following variables to `true` (the default is `false`):

  ```
  [CMD] set slk_fix_forbidden_chain_watch_setup_timing_window true
  [CMD] set slk_fix_forbidden_chain_watch_hold_timing_window true
  ```

- To fix forbidden chain violations to include clock components, set the following variable to true. By default, Tweaker does not touch clock components for forbidden chain ECO.

  ```
  [CMD] set slk_fix_forbidden_chain_dont_touch_clock true
  ```

- To specify a buffer list when you perform forbidden chain ECO, set the following command:

  ```
  [CMD] set slk_repeater_insertetion_buff_list { buffer_list }
  ```

- The insertion factor controls the candidate nets of repeater insertion during the autofix operation. When you set a larger value to the insertion factor, more net candidates are listed as the insertion candidate. To set the insertion factor during forbidden chain ECO, set the following command. To disable this feature, set the value to 0.

  ```
  [CMD] set slk_fix_forbidden_chain_insertion_factor value
  ```

- When addressing forbidden chain with the insertion method, Tweaker can control the minimum improvement for fixing each forbidden chain violation. You can define how many forbidden chain violations should be fixed for one inserted buffer by setting the following variable:

  ```
  [CMD] set slk_fix_forbidden_chain_min_improved_for_insertion value
  ```

- By default, Tweaker groups all forbidden chain violations and insert a buffer to fix the forbidden chain violations. When the following variable is set to true, the tool inserts all the buffer on the sink pin only to fix the forbidden chain violation:

  ```
  [CMD] set_fix_forbidden_chain_at_sink_pin_only true
  ```

Feedback

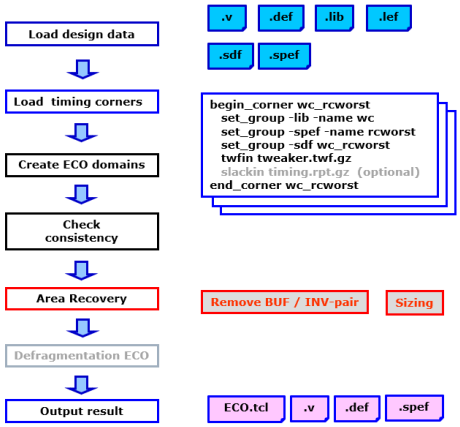
Area ECO With Tweaker-A1

For Area Recovery ECO, Tweaker-A1 reduces the cell utilization and gains free space for further ECO optimizations. The basic steps in Area Recovery ECO using Tweaker-A1 are as follows:

1. Load the design data (library files, LEF files, netlist, and so on)
2. Load timing scenarios
3. Create the ECO domain
4. Area recovery
5. Output ECO results

Area Recovery – Flow

1. Load design data
2. Load timing scenarios
3. Create ECO domain
4. Check consistency
5. Area recovery
6. Defrag ECO (optional)
7. Output result



The following topics describe some of the preceding steps in more detail.

- Build Up the MMMC Database for Area Recovery ECO
- Create the ECO Domain for Area Recovery ECO
- Area Recovery ECO Methodologies
- Report and Highlight Deleted ECO Cells
- Summary

Feedback

Build Up the MMMC Database for Area Recovery ECO

To maintain setup timing through multiple scenarios, you can establish each timing scenario with `begin_corner` and `end_corner` pairs, just like in the MMMC structure. For Area Recovery ECO, the slack reports are not mandatory and it only counts on the signoff quality TWF file to maintain each pin's timing changes. Each of Tweaker-A1's scenario contents use the following format:

```
begin_corner scenario_name
set_group -lib -name library_group
set_group -spef -name RC_group
set_group -sdf -name sdf_group
twfin -analysis_type on_chip_variation TWF_file
end_corner scenario_name
```

Create the ECO Domain for Area Recovery ECO

Creating the are recovery domain must be done after you create the timing corners and before the actual area recovery ECO. The various ways in which Area Recovery ECO can be created are:

1. Create the ECO domain by the user-defined window

   ```
   slkfix -add_window 1500 1600 1800 2000 ; # x1 y1 x2 y2

   slkfix -create_eco_domain_by_window
   ```

2. Create the ECO domain by BUF/INV pair timing margins for a complete design

   ```
   set slk_setup_target_slk 0.01

   set slk_hold_target_slk 0.01

   slkfix -create_area_recovery_domain
   ```

   This usage creates an area recovery domain defined by the setup and hold target slacks.

3. Create the Tweaker ECO domain by slack path

   ```
   slkfix -add_window -path_list [get_paths *]

   slkfix -create_eco_domain_by_window
   ```

   To create windows not by rectangles but by input violation paths. This is useful during the area recovery stage if you only want to remove buffers/inverter pairs along with the violation paths.

4. Create the ECO window based on grids

   ```
   slkfix -add_window -grid grid_collection
   ```

Feedback

To create the ECO window based on the specified grids, you need the design analysis grid db.

5. Create whole chip ECO domain

```
slkfix -create_whole_chip_domain
```

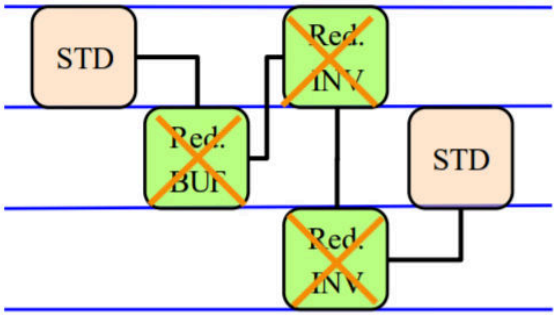
Area Recovery ECO Methodologies

The following are area recovery ECO methodologies:

- Removing Redundant Buffers and Inverter Pairs
- Sizing Down Cells
- Moving Cells With Advanced Defragmentation

Removing Redundant Buffers and Inverter Pairs

Tweaker can remove redundant buffers and inverter pairs based on cost area.

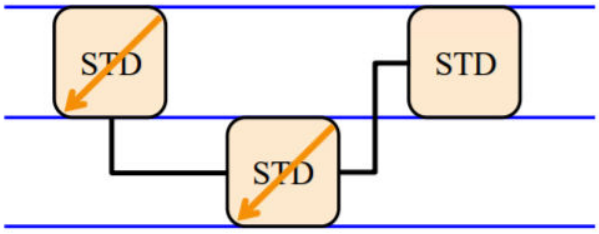


The following is the basic template of Area Recovery ECO based on deletion:

```
set slk_fix_area_recovery_by_deletion true
set slk_fix_area_recovery_by_sizing false
set slk_setup_target_slk 0.20
set slk_hold_target_slk 0.20
set_drv_factor 0.7
slkfix -area_eco
```

Sizing Down Cells

Tweaker can downsize the cells to reduce the area overhead.



The following is the basic template of Area Recovery ECO based on sizing:

```
set slk_cell_mapping_rule_regexp \
   {@S[0-9]+ @S[0-9]+ : @S[0-9]+@ @S[0-9]+@}
set slk_fix_area_recovery_by_deletion false
set slk_fix_area_recovery_by_sizing true
set slk_setup_target_slk 0.060
set slk_hold_target_slk 0.030
set_drv_factor 0.7
slkfix -area_eco
```

Moving Cells With Advanced Defragmentation

Tweaker can move cells in a congested region to create larger and more available space. Advanced defragmentation moves cells to a different row. To enable advanced defragmentation, set the following variable to `true` (the default is `false`):

```
[CMD] set_slk_fix_enable_defrag_to_different_row true
```

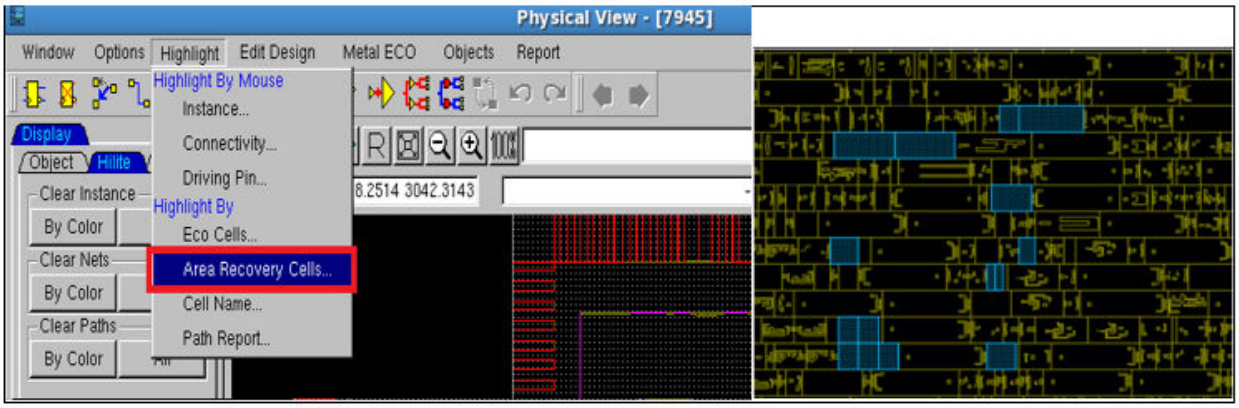
Report and Highlight Deleted ECO Cells

You can report and highlight deleted cells during area recovery with the following command:

```
report_eco_cells file_name [-deleted_cells]
```

You can highlight the location and shape of deleted ECO cells in the Physical View to show the recovered area.

Feedback



Summary

For Area ECO with Tweaker-A1, you can

- remove redundant buffers and inverter pairs, or downsize cells using area recovery.
- perform on multiple windows as well as a complete design.
- use violations based both for setup/hold area recovery.
- report and highlight deleted cells in the GUI.

High Performance Option ECO With Tweaker-HPO

Tweaker-HPO provides high performance features for faster convergence while providing better quality results. Adaptive ECO dynamically fine-tunes the ECO recipes based on the results of the run, then launches another incremental fix.

The following topics are covered:

- Adaptive ECO

Adaptive ECO

Adaptive ECO automatically relaxes ECO constraints. You define the blocking code release settings CSV file by specifying a process node. The tool parses the CSV file, then performs autofixing based on the CSV file to do the blocking code release.

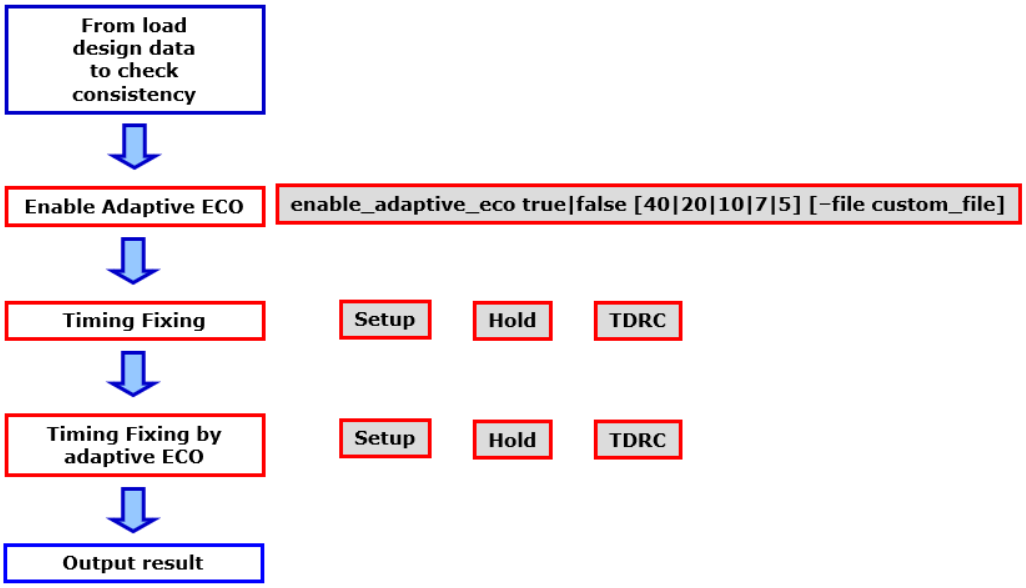
The following are the steps for the adaptive ECO flow:

1. Load the design to check consistency
2. Enable adaptive ECO

Feedback

3. Fix timing for setup, hold, TDRC
4. Review the blocking code
5. Fix timing for setup, hold, TDRC by adaptive ECO

Figure 62 The Adaptive ECO Flow



When you enable adaptive ECO, you can specify the process node for the project. The process node is specified in the CSV table name file (adapative_*process_node*.csv). For example, to specify the process node of 40, the CSV table name file is adaptive_40.csv. The command syntax is:

```
[CMD] enable_adaptive_eco true|false [40|20|10|7|5][-file custom_file]
```

If you do not specify the process note, the default of `false` and `7` (the adaptive_7.csv file) are used. You can also use the `-file` option which reads in your customized adaptive ECO settings.

The CSV file can contain the following types of fixes:

- TDRC (maximum transition, maximum fanout, and maximum capacitance)
- Setup
- Hold

Feedback

Example 64 shows the CSV file table format:

Example 64 CSV File Table Format

```
B022 tdrc:
 set+ slk_auto_sizing_max_fanout_limit 80
 set+ slk_max_shift_distance 5
B023 tdrc:
 if {slk_fix_hold_by_high_fanout_insertion} {
  set+ slk_on_route_search_range 8
 } else {
  set+ slk_on_route_buffer_insertion false
 }
# End tdrc
# Begin setup
B022 setup:
 set+ slk_auto_sizing_max_fanout_limit 80
 set+ slk_max_shift_distance 5
B022 hold:
 set+ slk_auto_insertion_max_fanout_limit 200
```

Alternatively, you can use your own custom file using the `-file` option. The command syntax is:

```
Blocking_code ECO_type:
   [set+ | set-] variable value1 value2
   command value1
```

Example 65 shows the custom file table format:

Example 65 Custom File Format

```
B023 setup:
 set+ slk_auto_fix_max_wire_length_limit 80

B022 setup:
 set+ slk_auto_sizing_max_fanout_limit 80
 set+ slk_max_shift_distance 5

B024 setup:

B022 hold:
 set+ slk_auto_insertion_max_fanout_limit 200

B034 setup:
    set_drv_factor 0.8
```

6

Output ECO Results

The output ECO results are covered in the following topics:

- Tweaker File Outputs From Successful Jobs
- Link to P&R and STA Tools

Tweaker File Outputs From Successful Jobs

Tweaker file outputs from successful jobs:

- Netlist
- DEF
- SDF
- SPEF
- Mapping List for Metal Layer ECO
- Tcl Files for P&R and STA Tools

Netlist

After each completed ECO job, Tweaker outputs the resulting ECO netlist file.

Note:

The amount of input and output netlist files from Tweaker is one-to-one.

Example 66 Output Specific Netlist Files From Tweaker-T1

```
# Tweaker outputs all netlists into the specified directory
verilogout directory_path
# Tweaker outputs only modified netlists into the specified directory
verilogout directory_path -eco
```

Feedback

DEF

With each additional ECO job, Tweaker outputs a set of partial (post-ECO) DEF files, each describing the placements of all of the job's ECO cells. For a hierarchical design Tweaker will output multiple DEF's for the post-ECO sub-designs. However, when no ECO changes are made to a certain sub-design, Tweaker will not generate a corresponding eco.def.

Example 67 Output ECO DEF Files From Tweaker-T1

```
defout directory/prefix                # ECO DEF file name will be
 "prefix.design.def"
defout -folder directory     # ECO DEF file name will be "design.def"
```

SDF

Tweaker outputs a partial post-ECO SDF file of each ECO Domain on a scenario-by-scenario basis. Each partial SDF file will contain both interconnect and cell delay values within the respective ECO Domain. If incremental SI SDF files are also fed in, Tweaker will output its updated contents in separate post-ECO SI SDF files. Tweaker will output a total of two ECO SDF files, one is the partial SDF and the other is the partial SI SDF file.

Example 68 Output ECO SDF Files From Tweaker-T1

```
sdfin -name wc_RCtypical_norm ...
....
sdfout directory/prefix       # ECO SDF file name will be
 "prefix.corner.sdf"
sdfout -folder directory      # ECO SDF file name will be "corner.sdf"
```

SPEF

Tweaker outputs partial ECO SDF files that contain Tweaker-estimated delay information for STA tools to back-annotate. In addition to that, you might sometimes prefer to output partial ECO SPEF files containing estimated RC information, such that STA tools can annotate and recalculated delay using its own engine. For instance, when an STA tool adopts PBA, you should annotate Tweaker-estimated RC values and recalculate delay instead of annotating Tweaker-estimated delay during the STA's what-if stage. This makes it easier for you to approach more reasonable timing results. Example 69 shows how to output the ECO SPEF files from Tweaker for each RC corner.

Example 69 Output SPEF Files From Tweaker-T1

```
spefout directory/prefix     # eco spef file name will be
 prefix.rc_corner
```

Feedback

Mapping List for Metal Layer ECO

Tweaker-M1 outputs a spare cell mapping list and gate array mapping list at the same time to feed into the P&R tool and finish the remaining ECO job. The command to output these lists is shown in Example 70.

Example 70 Output Spare Cell Mapping List From Tweaker-M1

```
spare_cell_map_out file_name
```

Tcl Files for P&R and STA Tools

As an alternative to ECO netlists and ECO DEF files, Tweaker outputs Tcl files to establish a link to third party tools. Currently, Tweaker supports outputting Tcl commands compatible with Synopsys PrimeTime, IC Compiler, Cadence EDI, ETS, Tempus, Mentor Olympus, and ATOP tech/ ATOP Tcl. See Example 71 for the appropriate Tcl output format commands.

Example 71 Output Tcl Files From Tweaker-T1

```
ecotclout -icc tcl_file
ecotclout -pt tcl_file
ecotclout -soce tcl_file
ecotclout -tempus tcl_file
ecotclout -olympus tcl_file
ecotclout -atop tcl_file
```

Summary Reports

Tweaker outputs auto-fixing, slack and leakage summaries. See Example 72 for the corresponding summary report output commands. The `slkreport -autofix` command summarizes all runs of the auto-fix (format shown in Figure 63). Figure 64 shows the reported slack summary from using the `slkreport -summary` command.

Example 72 Output Slack Reports From Tweaker-T1

```
slkreport -autofix auto_fix_summary
slkreport -summary slack_summary
slkreport -leakage file_name
```

Feedback

Figure 63 Summary of All Auto Fix Runs

```
# Fix Hold(Delay Insertion)
#-------------------------------------------
#         grow up area: 149.760
#  inserted cell count: 3
#                DBFS1: 2
#              DDLAY1S1: 1

# Fix Hold(Delay Insertion)
# autofix detail log summary :
#-------------------------------------------
# Blocked by Setup Path(155)
# Blocked by no setup margin (twf)(1223)
# Blocked by Timing Window (Setup)(38)
# Blocked by the disabled design(728)
# Blocked by twf clock pin(396)
# -------------------------------------------

output autofix summary : autofix.sum
```

Figure 64 Slack Summary

```
Write path summary report (report.sum) ...
File : ./bc.cworst.scan.hold

Group : all_group                    setup        hold
-------------------------------------------------------------
    Critical Path Slack:                        -25.8471
    Total Negative Slack:          0.0000     -219.4970
    No. of Violating Paths:             0           532
-------------------------------------------------------------
```

The `slkreport -leakage` command reports leakage summary in the format of Figure 65. You can report leakage before and after Power ECO to see the change in the amount of leakage power.

There are two ways to view additional details about a blocked timing fix:

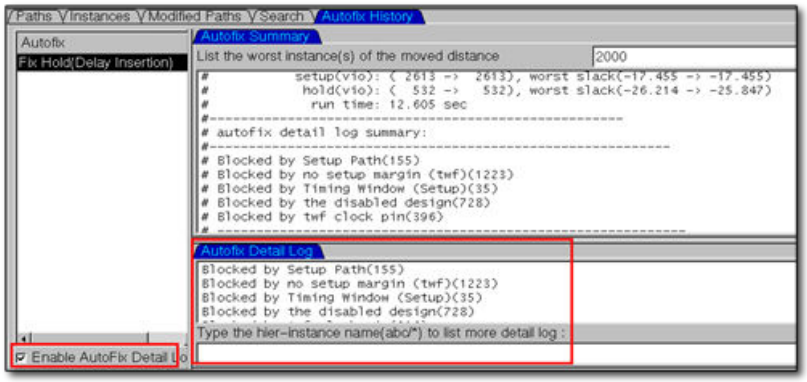
- Select and enable the **Enable Detail Log** check box in Tweaker GUI's Slack View (shown in Figure 66)
- Add the `-verbose` option after the `slkfix` command (shown in Example 73)

Feedback

Figure 65 Leakage Report Format

```
[ CMD ] slkreport -leakage_power
# leakage power:
#-------------------------------------------------------------
#      Clock Network:   0.01000 ->   0.01000 mW
#   Sequential Cells:   0.00614 ->   0.00614 mW
# Combinational Cells:   0.07528 ->   0.04311 mW
#-------------------------------------------------------------
#                     0.09142 ->   0.05926 mW
#
# vt cell count:
#-------------------------------------------------------------
#             *LVT*:   2502.376K ->    256.756K
#             *RVT*:   1750.945K ->   1553.673K
#             *HVT*:    680.592K ->   3123.484K
#            Normal:      0.000K ->      0.000K
#-------------------------------------------------------------
#                     4933.913K ->   4933.913K
# vt cell ratio:
#-------------------------------------------------------------
#             *LVT*:     50.73% ->     5.20%
#             *RVT*:     35.48% ->    31.49%
#             *HVT*:     13.79% ->    63.31%
#             ormal:      0.00% ->     0.00%
```

Figure 66 Detailed Log Provided in UI (Activated by the Checkbox on the Left)



Example 73 Option Usage: –verbose

```
slkfix -setup -all -verbose
```

Link to P&R and STA Tools

After each ECO job, you can output a single ECO netlist and partial ECO DEF file, or multiple netlist and DEF's for physically hierarchical designs. The P&R tools can read the ECO netlist before implementing an ECO Compare with the original database, and then use the eco.def file to place the ECO cells before launching ECO routing. Tweaker

supports outputting Tcl files for P&R tools including Synopsys IC Compiler, Cadence EDI, ATOP, and Mentor Olympus. The command to do this is:

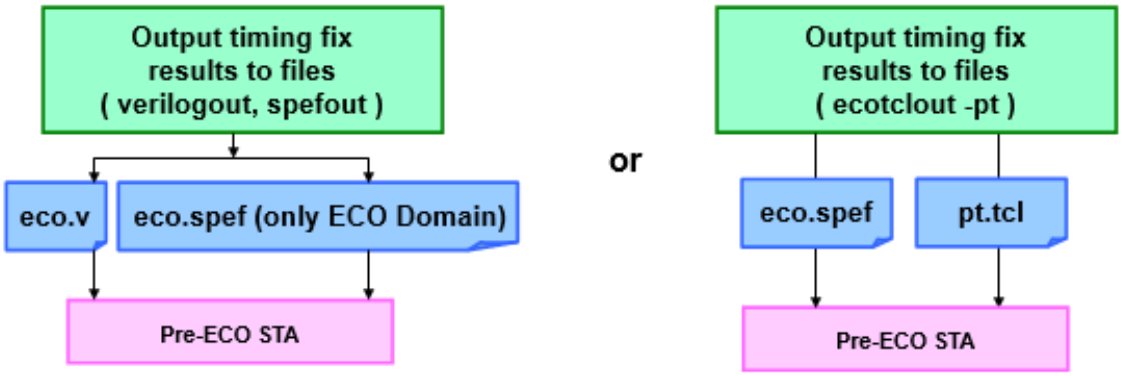
```
ecotclout -icc | soce | atop | olympus tcl_file.
```

To predict STA results before the ECO routing is submitted, you can use the ECO netlist to back-annotate the partial SPEF file to STA environment. This procedure called “Pre-ECO STA” is convenient because you can output an eco.tcl file to feed into STA tools. For example, you can implement “Pre-ECO STA” by restoring a saved STA session, source the eco.tcl file, then report timing. Note that when the eco.tcl file is sourced into the STA, Tweaker will recalculate the delay of Tweaker’s inserted or moved cells based on the STA’s wire load models. For better delay accuracy, you can use physical information such as the Tweaker-estimated delay info (eco.sdf) or estimated RC info (eco.spef) to back-annotate into the STA engine. Tweaker supports STA tools such as PrimeTime, Tempus, and GoldTime. The corresponding Tcl outputting command is:

```
ecotclout -pt | gt | tempus | ets | tc tcl_file.
```

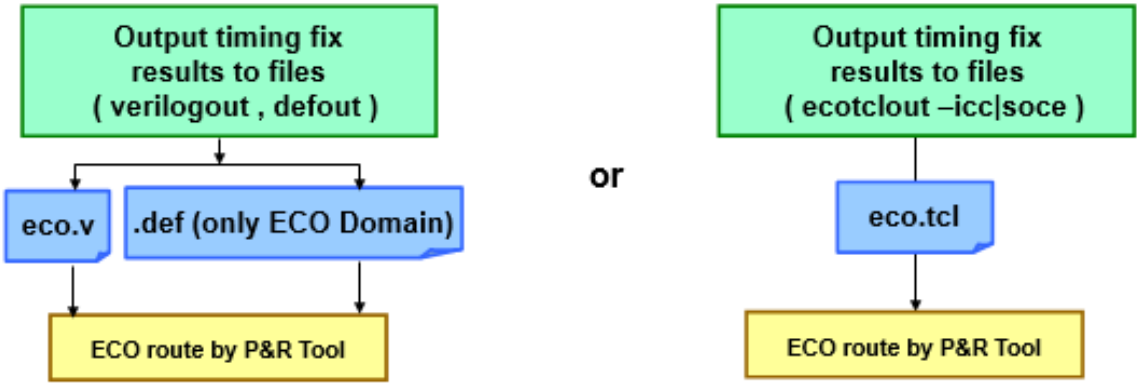
Here is the flow for Tweaker output files to succeeding third party tools:

Figure 67 Feed Tweaker Results Into STA Tool for Pre-ECO



Feedback

Figure 68 Feed Tweaker Results Into P&R tool for ECO Route



7

Filter Input STA Reports

For a large number of scenarios (more than eight), many timing violation paths that reside within different scenarios might actually be identical paths, with the only difference being in the slack values. This situation calls for a pre-process operation dedicated to filtering out repeated paths and keeping the dominant ones, which is the worst negative slack (WNS) path. As a result of the pre-process path filter, the amount of Tweaker input data shrinks significantly, which then dramatically optimizes runtime and also maintains the same fixing quality. Use the `extract_report` command to filter specified violation reports.

The following topics are covered:

Benefits

- The filtering process saves you from building redundant scenarios; this means less loading, faster runtime.
- Each dominant slack path is accurately maintained in each corresponding scenario such that timing is updated correctly.

Key Commands and Settings

Before fixing, Tweaker filters out not only path report redundancies*, but also constraint violation reports**. Tweaker's key command for filtering is `extract_report`, which extracts all dominant paths or constraints from every violation report and provides them to Tweaker-T1. Use the `man extract_report` command at the prompt to view the filtering options and other usage information.

* Generate violation path reports using the PrimeTime `report_timing` command

** Generate constraint violation reports using the PrimeTime `report_constraint` command

Follow these steps to extract the path or constraint violation report:

1. Load in the design
2. Establish the scenarios
3. Run the `extract_report [-path | -constraint]` command
4. Timing ECO

8

The Timing Window File: Definition and Role

One of the vital elements to Tweaker-T1's timing fixes is Timing Window Files (TWF), which are received from an STA tool. Use the `source twfout.tcl` Tcl-sourcing command to dump the correct TWF files in preparation for Tweaker analysis. The twfout.tcl Tcl script can be found in the following directories, in respect to the STA tool in use:

$Tweaker_Install_Dir/dorado/etc/scripts/tcl/pt (for PrimeTime)

$Tweaker_Install_Dir/dorado/etc/scripts/tcl/tempus (for Tempus)

This topic introduces the TWF format and how to dump a TWF for various operations.

The Timing Window File Format

The following is an example of the TWF file format.

Example 74 PrimeTime TWF File

```
DORADO_PT_TWF
DATE "Thu May 12 15:26:16 2011"
DESIGN "top"
DELIMITERS "/[]"
TIME_SCALE 1e-9
CLK_COMP
U11/A
U12/A
SLACK_COMP
*****************************************************
Design : top
Version: ...
Date   : ...
*****************************************************
      Max_Rise    Max_Fall    Min_Rise    Min_Fall      Point
-----------------------------------------------------------
      36.972      *           36.972      *               U13/CK
      94.336      94.834      2.789       2.316         U13/Q
1
PBA_COMP
```

```
U16/Q 95.0 96.0
TRANS_COMP
U13/CK 1.00000 VCLK VCLK1 VCLK2 (0.199:0.299)(*:*)
U13/Q * NULL (*:*)(0.138:0.146)
U14/Q 3000.000 VCLK3000.000 (*:0.20)(0.13:*)
```

TWF Glossary:

1. CLK_COMP: Clock network components in the design
2. SLACK_COMP: Maximum rise slack, maximum fall slack, minimum rise slack, and minimum fall slack for each pin in the design
3. PBA_COMP: Recalculated pin slack value in PBA mode within the specified slack range

Use Example 75 as a reference for usage formats of each TWF-dumping option. Using the proper options allow you to receive the appropriate TWF's for upcoming ECO operations.

Options:

- `-normal`: Appropriate for regular timing and power ECO without the SI_AWARE feature
- `-pba`: Use this option to output TWF files when the STA is under PBA mode

Example 75 TWF -dumping Options

```
pt_shell> source twfout.tcl
pt_shell> twfout -normal "../data/chip.twf"
pt_shell> twfout -pba "../data/chip.pba.twf"
```

Role of the Timing Window File

Tweaker counts on the values in the timing window file (TWF) to judge whether or not an ECO action will impact the existing timing. Through each run, the TWF is constantly updated to consider ECO Domain timing at the whole-chip level and for this reason, Tweaker guarantees safe ECO.

Simply put, the TWF acts like a reference system, and it can be applied during all available timing fix functions, including Clock ECO. For example, one of your biggest concerns is whether or not fixing hold time will impact the existing setup time. Since the TWF is incrementally updated, Tweaker can easily predict the remaining setup margins available for buffer insertion at the hold paths. An additional example in terms of fixing setup time is the consideration of timing impact on peer paths, since sizing will increase the loading on drivers. Again, with the TWF incrementally updated, Tweaker foresees the impact and can check whether or not sizing is a valid fix.

Tweaker will maintain an independent TWF for each scenario. Furthermore, when timing fixes are executed, Tweaker will merge all TWFs and then select the worst slack value.

In the case that TWF's are not referenced, Tweaker will still fix timing based on slack reports only; however, Tweaker will not be able to predict timing impacts precisely through this process. Therefore, you should prepare TWFs for Tweaker in any given situation, to avoid timing impact complications down the road.

9

Low Power Application in Tweaker

For designs with multiple power domains, Tweaker supports both CPF (Common Power Format) and UPF (Unified Power Format). As an ECO tool, Tweaker is responsible for inserting ECO cells without violating either logical or physical low power rules. For example, when inserting a buffer for fixing transition violation, it must be logically inserted at the correct module, and physically placed at the correct power domain.

To support timing fixing for designs with multiple power domains, Tweaker needs to read in either CPF or UPF to define the power-related cells and logical power domains. Then, Tweaker must read in physical power domain information with the `create_voltage_area` command.

The following topics are covered:

- Power-Related Cells
- Power Domain Definition
- Power Domain ECO Insertion Behavior

Power-Related Cells

Power-related cells are recognized by Tweaker either through the attributes defined in the Liberty library (.lib) or through the commands in the CPF or UPF files. Currently, supported power cells are always-on cells, isolation cells, power switch cells, and level shifters.

Power Domain Definition

You can define physical power domains using the `create_voltage_area` command. The command usage is as shown in Table 13.

Note:

Before applying the `create_voltage_area` command, you must specify the corresponding hierarchical design using the `current_design` variable, shown in Example 76.

Feedback

Table 13 The create_voltage_area Command Syntax

Syntax	`create_voltage_area`
	`[-name | -power_domain` *`domain_name`*`]`
	`[-polygon {x1 y1} {x2 y2} … | -coordinate {llx1 lly1 urx1 ury1} {llx2 lly2 urx2 ury2} … ]`
	`[-instances` *`module_instance_name`*`]`

Example	`create_voltage_area -name PD_1 -polygon {x1 y1} {x2 y2} {x3 y3}`
	`create_voltage_area -name PD_2 -coordinate {llx1 lly1 urx1 ury1} {llx2 lly2 urx2 ury2}`
	`create_voltage_area -name PD_3 -polygon {x1 y1} {x2 y2} -instances module_A module_B`

Options	
`-name` `-power_domain`	Specify the power domain name
`-polygon`	Define power domain using a polygon, with each of the polygon's vertex coordinates specified
`-coordinate`	Define power domain using multiple rectangular blocks; each rectangular block is constructed by specification of its bottom-left and top-right corners' coordinates
`-instances`	Define logical power domains on certain module instances

Example 76 Specify Design Hierarchy in Tweaker

```
set current_design TOP
source create_voltage_area_top.tcl
set current_design SUB1
source create_voltage_area_sub1.tcl
set current_design SUB2
source create_voltage_area_sub2.tcl
```

*Each Tcl file contains `create_voltage_area` commands.

Power Domain ECO Insertion Behavior

From a logical standpoint, Tweaker prohibits any insertion between a module port and a power cell to avoid power domain conflicts. From a physical standpoint, Tweaker simply confirms whether an inserted ECO cell is placed into a physical power domain that corresponds to the logical power domain. Therefore, it is crucial that you verify that the information in your logical and physical power domains match before launching an ECO.

Feedback

Some low power projects may involve macro power domains, especially for memory modules, and the information for these macros are often defined in the CPF files. Macro power domains are specified through the `set_macro_model` command, through which each power domain (that is, each memory module) can be defined by you as an independent design through select ports.

To match the *top* and *macro* power domains, Tweaker supports a `set_instance` command, which works with the `set_macro_model` command to accomplish more comprehensive low power structures within the ECO platform. Refer to Example 77 to observe how Tweaker supports both commands.

Example 77 Creating Multiple Power Domains

```
set_macro_model modelA
create_power_domain -name PD1X -boundary_ports { BISTER BISTEW }
create_power_domain -name PD2X -boundary_ports { AWTR OER }
end_macro_model

# Define power domain
create_power_domain \
   -name v_TOP \
   -default

create_power_domain \
   -name v_D1 \
   -instances { I544555/d1 }

create_power_domain \
   -name v_TOP_M5085 \
   -instances { I544555/I544516/I364288/I361008/I360626 }

set_instance
 I544555/I544521/I399391/I399353/I398663/I398630/I397516/I397126 \
   -model modelA \
   -domain_mapping { {PD1X v_D1} {PD2X v_TOP_M5085} }
```

10

Signal Integrity in Tweaker

As an ECO tool, Tweaker's responsibilities in terms of signal integrity (SI) are to:

1. Recognize the incremental SI delay components embedded in the slack reports
2. Update the SI delay properly as ECO actions are implemented

Tweaker-T1 supports the Celtic SI flow and PrimeTime SI flow methodologies.

The following topics are covered:

PrimeTime Signal Integrity Flow

Because there is no independent SI SDF file for PrimeTime, it needs to extract SI components in advance using two SDF files, SDF with SI (sdf_w_si) and the SDF without SI (sdf_wo_si). To get these SDF files, sdf_w_si can be dumped from PrimeTime SI, while "sdf_wo_si" can be outputted from PrimeTime.

The Tweaker command to extract SI components for the PrimeTime SI flow is `extract_sisdf`, and its usage is listed in Table 14.

Table 14 The extract_sisdf Command Syntax

Syntax	`extract_sisdf [sdf_wo_si] [sdf_w_si] [output_si_sdf]`
Example	`extract_sisdf top_wo_si.sdf.gz top_w_si.sdf.gz pt.si.sdf`

Notice that the SI components in the pt.si.sdf file can be found not only on the interconnect delay arcs, but also at the cell delay and timing check arcs. Refer to Example 78 for how to extract SI SDF and assign them to each corresponding scenario.

Feedback

Example 78 Extract SI SDF

```
extract_sisdf mainA.sdf.gz mergedA.sdf.gz pt.A.si.sdf
extract_sisdf mainB.sdf.gz mergedB.sdf.gz pt.B.si.sdf
extract_sisdf mainC.sdf.gz mergedC.sdf.gz pt.C.si.sdf

sisdfin -name A_si pt.A.si.sdf
sisdfin -name B_si pt.B.si.sdf
sisdfin -name C_si pt.C.si.sdf

begin_corner A
set_group -lib -name bc
...
slackin ... -si A_si ...
end_corner A

begin_corner B
set_group -lib -name bc
...
slackin ... -si B_si ...
end_corner B

begin_corner C
set_group -lib -name wc
...
slackin ... -si C_si ...
end_corner C
```

Signal Integrity Update

Any effects on SI vary based on how cells or wires are changed. In conventional timing ECO, updating SI delay accurately requires a time-consuming detailed SI analysis based on detail routing. As an ECO-dedicated tool, Tweaker-T1 incorporates a way to instantly estimate the SI effect with reasonable accuracy, even before the ECO routing process is launched. Compared with the conventional flow, Tweaker-T1 provides a shortcut to gauging the effect of SI and saves enough time for detailed ECO routing, extraction, SI analysis, delay calculation, and STA.

The way that Tweaker-T1 estimates SI effect is very straightforward, it scales SI delay based on the change in output transition.

Simple Flow Without extract_sisdf Usage

When sdf_w_si and sdf_wo_si are both unavailable, Tweaker-T1 provides a simple flow to quickly start timing fixes without needing to use “extract_sisdf”. In this flow, there will be no “SI updates” since Tweaker-T1 does not know where each SI component is. Refer to Example 79 for how to build the Power ECO flow using PrimeTime SI and Tweaker-T1.

Example 79 Power ECO Flow for PrimeTime SI and Tweaker-T1

```
# In PrimeTime SI
restore_session TOP
write_sdf           /* this is the "sdf_w_si" */
source twfout.tcl
twfout -power_eco * "../report/top.twf"
quit

# In Tweaker-T1
begin_corner ss_cworst
set_group -lib -name ss
set_group -spef -name cworst
set_group -sdf -name cworst_ss
twfin -type sdf_max "../report/top.twf.gz"
end_corner ss_cworst

slkdc -check_slack_consistency
source $script_path/power_eco_setting.tcl
slkfix -power_eco
ecotclout -pt "power_eco.pt.tcl"

# Then in PrimeTime SI again to verify the result before ECO routing
restore_session TOP
source power_eco.pt.tcl
update_timing
report_constraint -all_violators    /* Verify if there's timing impact */
```

11

Don’t-Touch, Don't-Use, and Ignore Settings

Prior to timing fixes, you might prefer to skip optimization on certain objects. To fulfill this preference, Tweaker features don’t-touch settings to avoid any fixing procedures on paths, instances, or cells.

The following topics are covered:

Don’t-Touch Paths

To avoid fixing any specific slack paths, use the `dont_touch_slk_path` command. Paths that match your don’t-touch specifications within each command option will be locked from any fixing. Enter `man dont_touch_slk_path` into the Tweaker command prompt for more usage details.

Example 80 Set don’t-touch Paths

```
dont_touch_slk_path -from | -through| -to given_pin
dont_touch_slk_path -slack_range -99 -0.5
dont_touch_slk_path -ff | -io| -all
```

Don’t-Touch Instances

To avoid fixing on any specific instances, use the `set_dont_touch_instance` command. Instances that match your don’t-touch specifications will have their pins locked from fixing and will become incapable of getting sized, inserted, or moved. Enter `man set_dont_touch_instance` into the Tweaker command prompt for more usage details.

Example 81 Set don’t-touch Instances

```
set_dont_touch_instance true|false instance_list | instance_pattern
set_dont_touch_instance true | false -file file_name
set_dont_touch_instance true | false -cell cells
set_dont_touch_instance true | false -from | -through | -to given_pin
set_dont_touch_instance true | false -all_registers
```

Don’t-Touch Cells

To avoid fixing on any specific library cells, use the `set_dont_touch_cell` command. The library cells that match your don’t-touch specifications will become incapable of getting sized or moved. In terms of selecting don’t-touch library cells, any library cells defined with the don’t-touch attribute in the .lib file are automatically honored as don’t-touch cells by Tweaker. Enter `man set_dont_touch_cell` into the Tweaker command prompt for more usage details.

Example 82 Set don’t-touch Cells

```
set_dont_touch_cell true | false cell_list | cell_pattern
set_dont_touch_cell true | false -instance instance_list -quite
```

Don’t-Touch Pins

To avoid fixing on any specific pins, use the `set_dont_touch_pin` command. The instance pins along the tracing route that match your don’t-touch specifications will be locked from fixing. Enter `man set_dont_touch_pin` into the Tweaker command prompt for more usage details.

Example 83 Set don’t-touch Pins

```
set_dont_touch_pin true|false instance_pin_list
set_dont_touch_pin true | false -file file_name
set_dont_touch_pin true | false -from | -through | -to given_pin
set_dont_touch_pin true | false -netlist_assign
```

Feedback

Don’t-Touch Nets

To avoid fixing on any specific nets, use the `set_dont_touch_net` command. The nets that match your don’t-touch specifications will be locked from fixing. Enter `man set_dont_touch_net` into the Tweaker command prompt for more usage details.

Example 84 Set don’t-touch Nets

```
set_dont_touch_net true | false hierarchy_net_list
set_dont_touch_net true | false -file file_name
```

Don’t-Touch Nets by Pin

To avoid fixing on any specific nets according to any related pins, use the `set_dont_touch_net_by_pin` command. The nets that match your don’t-touch specifications will be locked from fixing. Enter `man set_dont_touch_net_by_pin` into the Tweaker command prompt for more usage details.

Example 85 Set don’t-touch Nets by Pin

```
set_dont_touch_net_by_pin true | false instance_pin_list
set_dont_touch_net true | false -file file_name
```

Don’t-Use Cells

To avoid using any specific cells while fixing, use the `set_dont_use_cell` command. The library cells that match your don’t-touch specifications will not be selected for use during cell-sizing. In terms of selecting don’t-use library cells, any library cells defined with the don’t-use attribute in the .lib file are automatically honored as don’t-use cells by Tweaker. Enter `man set_dont_use_cell` into the Tweaker command prompt for more usage details.

Example 86 Set don’t-use Cells

```
set_dont_use_cell true | false cell_list | cell_pattern
```

Ignore DEF Fixed Instances

By default, any instances defined with the “Fixed” attribute in DEF will be labeled as don’t-touch in Tweaker. Still, you can use the `slk_dont_touch_def_fixed_inst` variable to decide whether or not to honor the DEF attribute. “Move” or “Sizing” attributes do not apply to “Fixed” instances; however, buffer insertion is allowed.

Feedback

Ignore DEF Blockage

By default, any blockages defined in the DEF will be honored in Tweaker. To release the blockage space in Tweaker during the ECO process, you can set the `ignore_def_placement_blockages` variable to `true`.

Enter `man ignore_def_placement_blockages` into the Tweaker command prompt for more usage details.

For blockages with the “Soft Blockage” attribute, Tweaker treats these as free space since in general, soft blockages are placed to reserve space for ECO.

For blockages with the “Partial Blockage” attribute, Tweaker will first check if the local density of the blockage area is above or below the threshold defined in DEF. Based on that information, Tweaker will either treat the area as free space or as blocked area. For example, a partial blockage is given with 50% density threshold (PARTIAL 0.5). If Tweaker finds the local density is 45%, then the area will be considered as free space. On the other hand, if the local density is 55%, then the area will be blocked for ECO process.

Library Cells to be Ignored

If for any reason, there are library cells to be ignored during `libin` or `lefin`, you can specify them using the `library_cells_to_be_ignored` variable. If this variable is set before `libin` and `lefin`, neither of the lib or LEF of the cells will be read in. If the variable is set after `libin` but before `lefin`, then the lib of the cells will still be read in but the LEF of the cells won’t. Enter `man library_cells_to_be_ignored` into the Tweaker command prompt for more usage details.

Physical Components to be Ignored

Physical components, such as filler cells in DEF, are normally associated with the “DIST” attribute. Tweaker honors the “DIST” attribute in DEF, and provides some useful relevant variables. Set the `ignore_def_physical_cells` variable to `true` before `defin` to read in DEF with physical components ignored. This variable is usually used when a DEF is found without free space due to filler cells not being removed before the DEF was output from the third party P&R tool. With the physical components ignored during `defin`, you will not need to go back to the P&R tool to output another DEF without physical components. To ignore only certain types of the physical components, you can specify the physical components with the `def_physical_cells_to_be_ignored` variable.

Example 87 Set Physical Cells to Ignore

```
set ignore_def_physical_cells true
set def_physical_cells_to_be_ignored { FILLER* }
```

Feedback

```
set ignore_def_physical_cell_excluding_def_fixed true
defin ...
```

By default, the `ignore_def_physical_cell_excluding_def_fixed` variable will not touch any of physical cell types with the “FIXED” attribute, regardless of the cell types listed with the `def_physical_cells_to_be_ignored` variable. In this setting, only those without the “FIXED” attribute will be ignored and treated as free space.

12

Relay Fixing

During ECO, Tweaker will maintain a record of netlist updates in the nlcmd file in addition to the log file. By using an nlcmd file, you can resume a previously performed ECO job without restarting the entire ECO process.

The following topics are covered:

- Netlist Command File
- Restore nlcmd With Accurate Timing Updates
- Sourcing an nlcmd File
- Command Usage: save_session and restore_session

Netlist Command File

The netlifst command file (nlcmd) is found in the working directory, accompanied by its corresponding log file. All netlist update history is recorded in the nlcmd file, which can be used to “replay” the previous netlist editing job.

Example 88 Record of Instance A’s Sizing and Post-Sizing Legal Placement

```
#sizing: (DCKBFS3)->(DCKBFS8) inst (A)
twf -begin_sizing { top A }
rep -master { top A -n DCKBFS8}
place {A} 1115.820000 1215.520000 0 y PLACED
twf -end_sizing

#corner(wc_cwosrt) mode (func)
#slack (-14.65)->(-14.19) sub twf (999.000:-17.306)->(998.539:-16.933)
```

Information relevant to timing fixes – like in Example 88 – is considered useful and therefore recorded for necessary debugging purposes. In a new Tweaker run, if you want to restore the previous efforts you can source the nlcmd file after loading design data and the netlist will be updated exactly the same way as before.

Feedback

Restore nlcmd With Accurate Timing Updates

When sourcing an nlcmd file, you must consider which procedure to use to restore the previous state of a design or job. To restore only physical data and netlist updates, you simply need to load in the design and source the nlcmd. If you want to source an nlcmd to restore previous timing fix results (or any jobs such as “sdfout”, which requires correct timing updates) add the additional `slackin -nlcmd nlcmd_file` command to build extra nlcmd-based ECO domains in advance. For instance, a setup timing fix job is divided into three sub-tasks so that three engineers can work in parallel to save time. Each engineer’s results (fixsetup.1.nlcmd, fixsetup.2.nlcmd, fixsetup.3.nlcmd) can be merged and followed by a hold time fixing later. In this case, you should follow Example 89 and use the `slackin -nlcmd nlcmd_file` command in order to build the correct ECO Domain, then source the nlcmd files to restore the netlist change made by *fixsetup*. If the corresponding “ECO Domain” of *fixsetup* is not built in advance, Tweaker-T1 will not have correct database to update timing when the fixsetup.*.nlcmd files are sourced.

Example 89 Sourcing an nlcmd File to Restore Previous Timing Fix Results

```
begin_corner wcl_cworst
...

# Build the "ECO Path Domain" of previous timing fix result
slackin -nlcmd fixsetup.1.nlcmd
slackin -nlcmd fixsetup.2.nlcmd
slackin -nlcmd fixsetup.3.nlcmd
end_corner wcl_cworst

begin_corner bc_cbest
slackin -scan … "rpt_path/scan.hold.rpt.gz"
end_corner bc_cbest

slkdc -check_slack_consistency
...

# Batch netlist editing for restoring the previous timing fix result
source fixsetup.1.nlcmd -skip_conflict
source fixsetup.2.nlcmd -skip_conflict
source fixsetup.3.nlcmd -skip_conflict

slkfix -hold -all
```

Sourcing an nlcmd File

To combine previous fixing efforts with a current fixing job, source the old nlcmd files so that Tweaker will merge the old and new nlcmds’ contents and output the combined nlcmd in the same round. As a result, the latest combined nlcmd file will include all the sourced nlcmd’s and the most recently edits in the same round.

Feedback

Command Usage: save_session and restore_session

Tweaker also provides the `save_session` and `restore_session` commands to save your Tweaker database. Any runs in Tweaker can be saved using the `save_session` command. All environment settings and ECO jobs that were previously done will also be recorded into the saved session. Additionally, any previous sessions can be continued using the `restore_session` command. This command enables you to resume you Tweaker ECO database and continue where you left off in your ECO job. You can also write out entire reports and results (this includes information from the current session, as well al the session that was restored).

13

Frequently Seen Warning Messages

The following topics shows frequently seen warning messages:

TWF Setup and Hold Impact and DRV Checking Failures

During manual fixing, you might encounter warning messages regarding timing impacts through TWF updates.

Figure 69 TWF Impact Warning Message Window

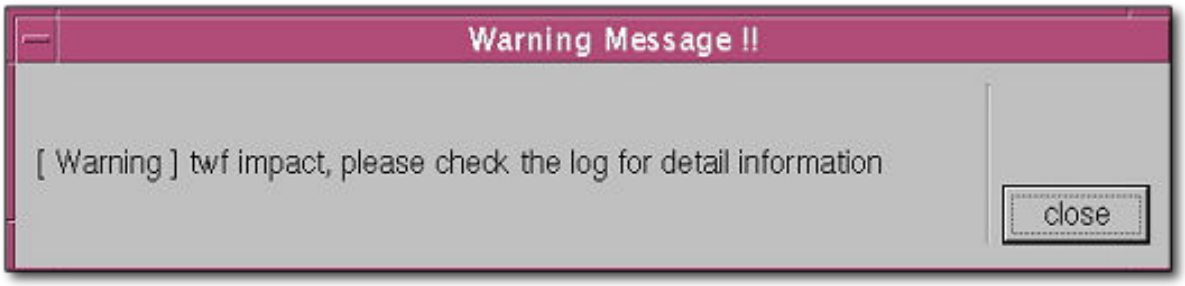
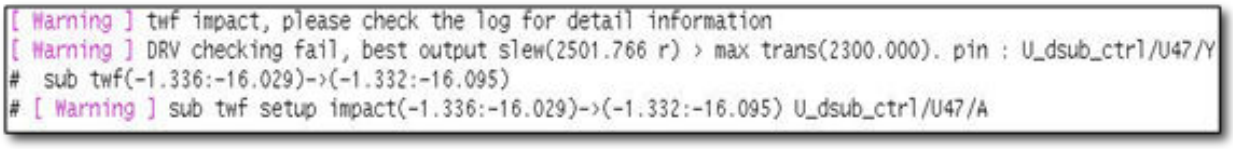


Figure 70 Log Message



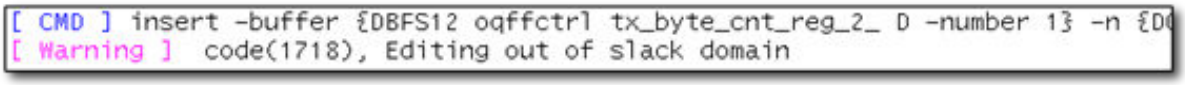
Therefore, it is important to watch the log message for detailed TWF updates with each manual edit, whether the updates are from DRV checking failure or from TWF setup and hold impact.

Editing Outside the ECO Domain

When manually modifying a netlist that is outside of the ECO domain, a warning window shown in Figure 71 can appear. In addition, the corresponding timing updates are not guaranteed.

Figure 71 Editing Outside the ECO Domain Warning Message

Figure 72 Log Message



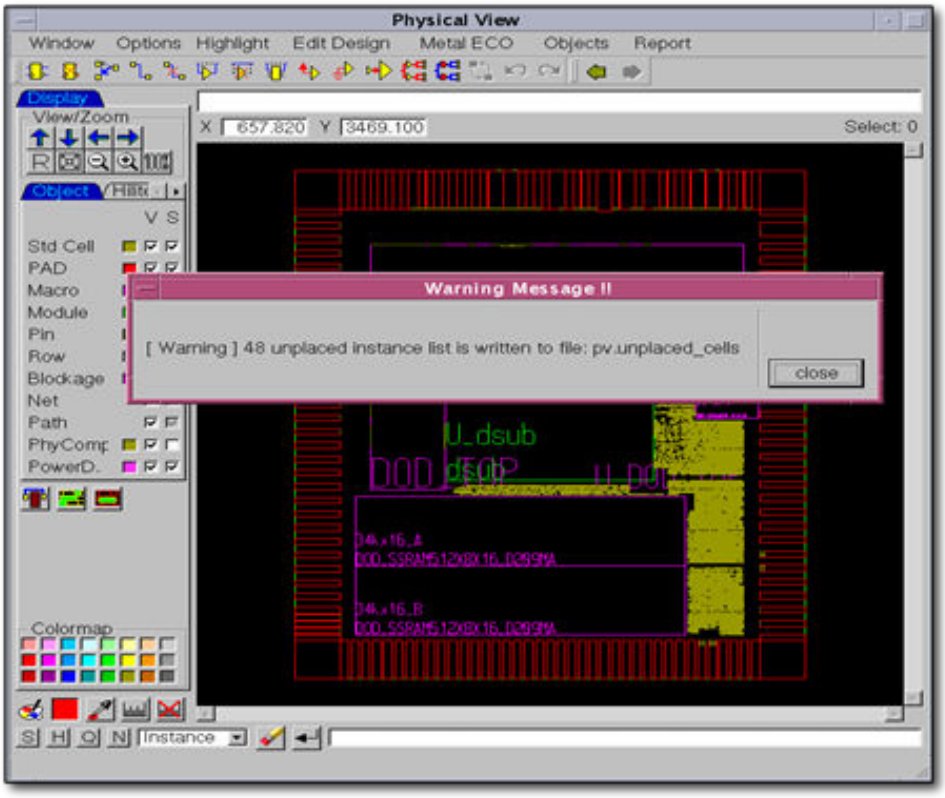
Any unwanted edits can be undone with the **Undo** button, which returns you to the previous step.

Physical View: Unplaced Cell Warning

When the physical view is opened for the first time, the following warning message can appear if there exists unplaced cells recognized by Tweaker. All unplaced cells are recorded in the pv.unplaced_cells file within your working directory.

Feedback

Figure 73 Unplaced Cells Warning Message

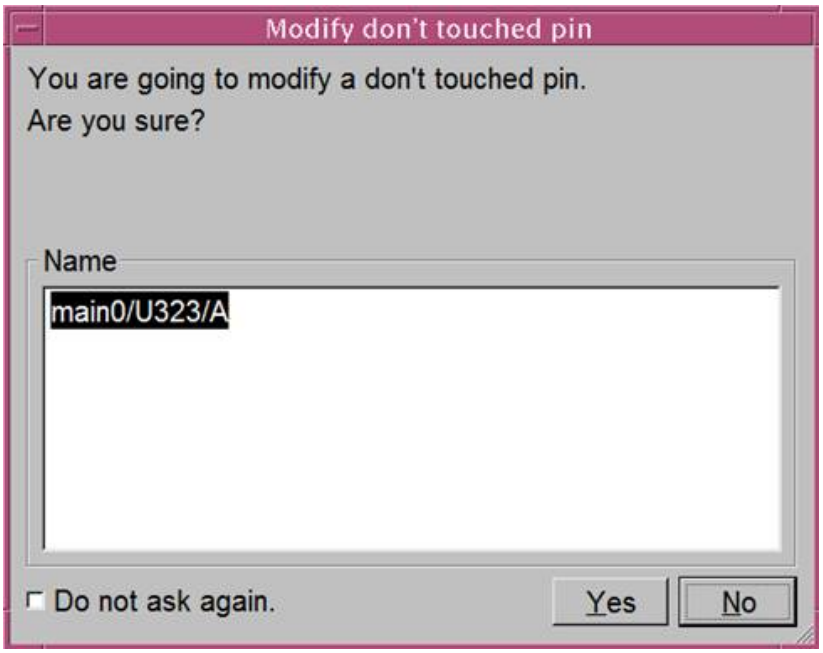


Editing Don’t-Touch Pins

When doing buffer insertion on locked pins or sizing/moving on an instance whose pin is locked, Tweaker will show the Modify don’t touched pin warning message. You can decide whether to continue or abort editing.

Feedback

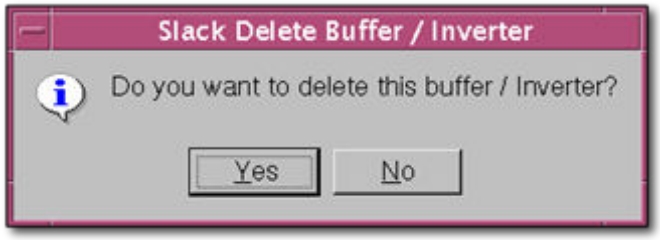
Figure 74 Modify don't touch Pin Window



Deleting Buffers or Inverters

When a buffer or inverter is deleted, the window shown in Figure 75 appears to confirm the deletion.

Figure 75 Delete Buffer/Inverter Confirmation Window



14

The Job Monitor Window

The Job Monitor window provides a user interface to quickly review important information from the log file, without having to open the log file. You can view the task information, such as the count, quality, and the ECO step. Basic and machine condition information are available. You can also schedule job status e-mail reminders.

The following topics are covered:

Benefits

The benefits of the job monitor window include:

- Ability to quickly review the quality of input data
- View quality of results (QoR) with line charts that display fixing trends
- View the QoR summary histogram and the top five highlighted blocks
- Schedule e-mails to provide job updates
- Record machine and job task status
- The job tracking system creates flowchart to manage sequence of databases and ECO strategies

Feedback

Launching Job Monitor

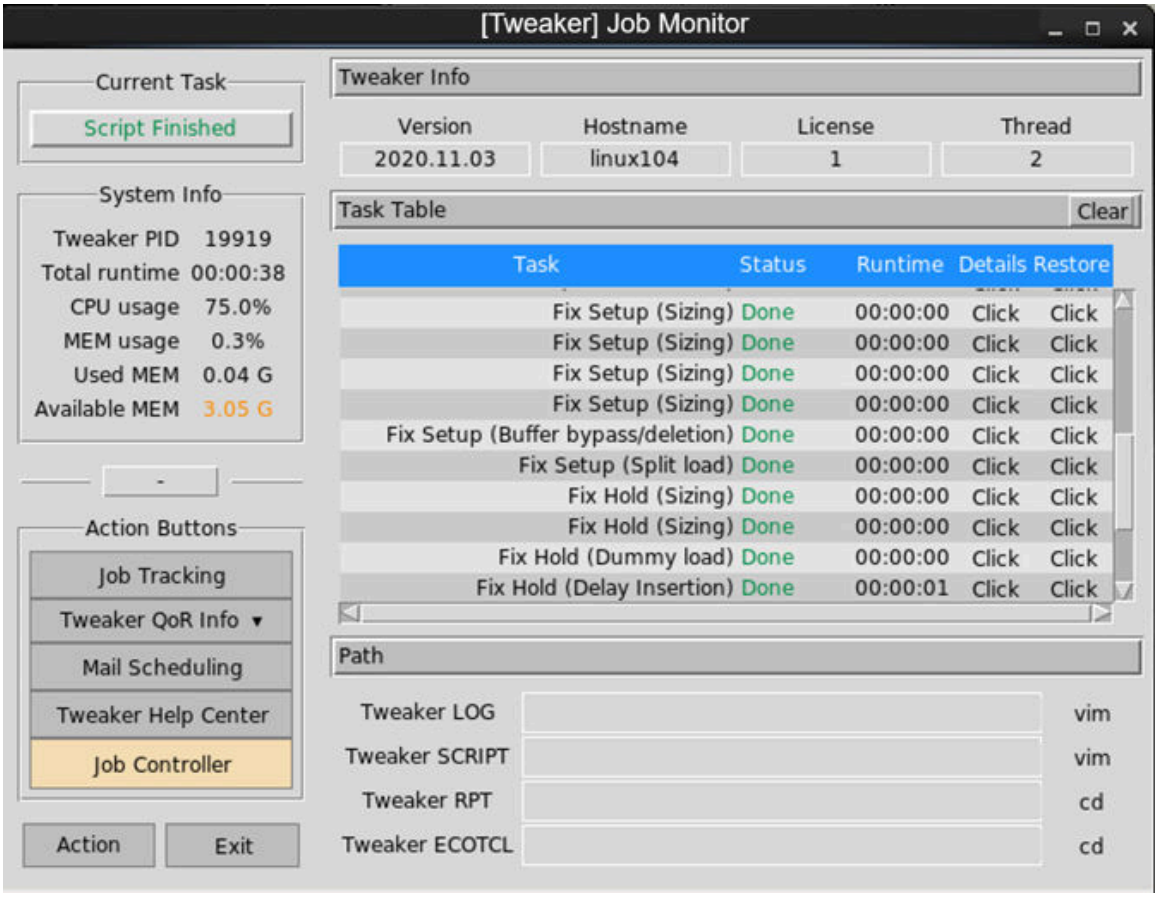
To launch the job monitor window, do one of the following:

- For each Tweaker run,
  - Add the following at the beginning of your script: job_monitor::begin
  - Add the following at the end of your script: job_monitor::end
- Open a Tweaker run and use the following command to parse information from the log file:

```
tweaker_shell> job_monitor::begin $LogFile_Path
```

Figure 76 shows the Job Monitor window.

Figure 76 The Job Monitor Window



Job Monitor GUI Functions

Each of the GUI functions are described as follows:

- Current Task

  Click the Script Finished button to view a line chart of the ECO progress. Click each node to view the fixing summary, blocking codes, and settings.

- System Info

  The following information is provided:

  - Tweaker PID

    Click to attach a GDB file

  - Total runtime

    The GUI runtime is used rather than the Tweaker runtime

  - CPU and Memory usage

    Available MEM displays in orange when the available memory is lower than 5G and displays in red when it is lower than 3G

- Tweaker Info

  The Tweaker version, host name, license, and thread information are displayed

- Action Buttons

  The following actions are available:

  - Job Tracking

    Click the Job Tracking button to check in all jobs and create the flowchart to manage the sequence of the databases and ECO strategies. Each job refers to a job monitor data file. You can open the job monitor to view detailed information. You can compare two jobs with an endpoint violation report and the QoR summary.

  - Tweaker QoR Info

    Click the Tweaker QoR Info button to display the QoR summary, runtime and memory information, and input file details.

  - Mail Scheduling

    Click the Mail Scheduling button to display the Mail System Settings window. You can schedule e-mails to provide status job status information.

  - Tweaker Help Center

Click the Tweaker Help Center button to display the Tweaker Manual File Query System window. You can search for commands, variables, and rules in the Pattern field. You can use the wildcard (*) in your search.

- Job Controller

  You can pause Tweaker to apply commands. Use this when you want to report ECO Tcl during autofix that takes a large amount of time to finish. Default commands are available to add in the text file. Click Continue to apply the commands and continue with Tweaker. Click Apply to apply the commands without continuing to Tweaker. Click Cancel to cancel the commands.

- Task Table

  This section displays the task status, reports errors, and displays autofix summary and settings. The task status is defined as follows:

  - Yellow highlight: Ongoing
  - Green font: Normal
  - Orange font: Degradation. Click the details to view the fixing summary.
  - Red font: Error. Click the details to view the error information.

- Path

  This section displays the following information:

  - Log: The file path is received when the job monitor is launched.
  - Script: The file path is found from the first appearance of “[CMD] source” in the log.
  - Report: The folder path is found from the last appearance of “[CMD] slkrpeort -autofix” in the log.
  - Ecotcl: The folder path is found from the last appearance of any ecotcl outputting message in the log.