A Utility for Leakage Power Recovery within PrimeTime\(^1\) SI

Bruce Zahn

LSI Corporation

Bruce.Zahn@lsi.com

ABSTRACT

This paper describes a utility which is run within the PrimeTime SI signoff environment that recovers leakage power and achieves optimal leakage results while preserving a design’s timing performance. This utility analyzes the design’s timing and does cell swapping of high leakage cells to lower leakage cells on paths with positive timing margins. The lower leakage cells are inherently slower, and the utility determines how many can be used while still achieving the performance target.

This utility takes into account crosstalk effects on design timing and supports multiple libraries of varying voltage thresholds. It minimizes runtime by reducing the number of timing iterations performed by estimating delays from potential cell swaps, and only implementing beneficial exchanges. Using postlayout signoff parasitics and highly accurate timing models, the utility achieves much greater leakage recovery over current optimization tools.

\(^1\) PrimeTime is a registered trademark of Synopsys, Inc.
Table of Contents

1.0 Introduction ...................................................................................................................... 4
2.0 Methodology and Flow ..................................................................................................... 5
  2.1 Flow Inputs ..................................................................................................................... 6
  2.2 Flow ................................................................................................................................ 6
  2.2.1 Initial Power Recovery Phase ..................................................................................... 7
  2.2.2 Speed Recovery Phase ............................................................................................... 11
  2.2.3 Capacitance and Transition Recovery Phase ............................................................. 15
  2.3 Flow Outputs ................................................................................................................. 15
  2.4 Distributed Multiscenario Analysis (DMSA) Flow .......................................................... 15
  3.0 Features .......................................................................................................................... 17
  3.1 Hierarchical Design ....................................................................................................... 17
  3.2 Multiple Library and Vth Cell Support ......................................................................... 17
  3.3 Interface Timing Paths ................................................................................................... 17
  3.4 Controlling Speed-Recovery Paths to Process .............................................................. 17
  3.5 Intermediate Files ......................................................................................................... 18
  3.6 Distributed Multiscenario Analysis (DMSA) ................................................................ 18
  4.0 Limitations ..................................................................................................................... 18
  4.1 Unconstrained Cells ...................................................................................................... 18
  4.2 Identifying Clock Network Cells .................................................................................... 18
  4.3 Incremental Timing Update Runtimes ......................................................................... 19
  5.0 Experiments .................................................................................................................... 19
  5.1 Internal Test Block ........................................................................................................ 19
  5.2 DSP Block ..................................................................................................................... 20
  5.3 Storage Read Channel Block ....................................................................................... 23
  6.0 Conclusions and Recommendations .......................................................................... 23
  6.1 Future Work ................................................................................................................... 24
  7.0 Acknowledgements ........................................................................................................ 24
  8.0 References ..................................................................................................................... 24

Table of Figures

Figure 1. Power Recovery Utility Flow Diagram ................................................................. 5
Figure 2. Example Vth Map File ......................................................................................... 6
Figure 3. Initial Power Recovery Flow Diagram ................................................................. 8
Figure 4. Example pwr_rec_slack Attributes .................................................................... 9
Figure 5. Complex Example of pwr_rec_slack Attribute Adjustment .................................. 10
Figure 6. Speed Recovery Flow Diagram .......................................................................... 12
Figure 7. Example Circuit for Speed Recovery ................................................................. 13
Figure 8. DMSA Flow Diagram ......................................................................................... 16
Figure 9. Generated Clock Issue Example ......................................................................... 18
List of Tables

Table 1: Internal test block Power Recovery experiment Data .......................................................... 20
Table 2. DSP 65 nm Power Recovery experiment Data ..................................................................... 21
Table 3. Quality of Results Data for various iterations of DSP experiment ................................. 22
Table 4. Storage 65 nm Power Recovery experiment Data ............................................................... 23
1.0 Introduction

One way to reduce leakage power is to use multithreshold voltage cells during the design flow. High-voltage threshold cells leak less than standard and low-voltage threshold cells, but are slower. Conversely, low voltage threshold cells leak more than standard and high-voltage threshold cells, but are faster. These various cells are footprint-equivalent to each other, meaning that they have an identical physical footprint and can be swapped without disturbing the layout routing.

At smaller technologies such as 65 nm and 40 nm, library cells with different channel lengths are available. For example, at LSI we provide minimum and extended channel length library cells. These cells all have the same physical footprint and can be swapped without disturbing layout routing. Thus, for a given technology, there may be many variants of footprint-compatible Vth cells.

The limitations of leakage optimization in non-signoff environments were discussed in 2004 SNUG [1] as:

- Delay calculation algorithms are less accurate.
- Static timing analysis does not reflect true signoff timing.

In addition, greater pessimism is needed in the optimization tools to reduce the likelihood of timing violations seen in the signoff tool for two reasons:

- At smaller technologies, the spread between signoff and implementation parasitics is increasing.
- Signoff tools use more accurate timing windows and crosstalk timing calculation

For these reasons performing leakage power recovery in the signoff environment has the following advantages:

- It produces signoff quality timing to achieve best quality of results
- It uses most accurate data (signoff parasitics, models, crosstalk and delay)
- It uses signoff constraints

This paper describes a utility run within PrimeTime SI that performs footprint-equivalent Vth cell swaps to achieve optimal leakage recovery while still achieving the performance target. The utility is run when the design is timing closed or during the final iterations of timing closure. Since the cell changes produced by the utility are footprint-equivalent, the timing predictions observed in PrimeTime SI can be realized after the changes are implemented in the layout.
2.0 Methodology and Flow

The leakage power recovery utility is run in the signoff environment from within PrimeTime SI. The utility analyzes the timing of a design and does cell swapping of high leakage cells to lower leakage cells on paths with positive timing margins. The lower leakage cells are inherently slower, and the utility determines how many can be used while still achieving the performance target. The utility is typically run on a design late in the design flow after the design timing is closed, and after the implementation tool has exhausted its own leakage recovery methods. The cells swapped are footprint-equivalent to each other meaning that they have the identical physical footprint and can be swapped without disturbing the layout routing at all.

The leakage power recovery flow is shown in Figure 1.

![Flow Diagram](image-url)
2.1 Flow Inputs

The inputs to the leakage recovery utility are a voltage threshold (Vth) map file and a slack limit value. The Vth map file lists the different Vth library names and Vth suffixes in order from largest leakage (fastest timing) to smallest leakage (slowest timing). The slack limit value specifies a timing limit for the design. The utility performs leakage recovery while ensuring that timing performance does not fall below the slack limit.

The format of the Vth map file is one set of libraries and Vth suffixes per line. Each line contains a space-separated list of library_name:Vth_suffix ordered from the fastest, most leakage cell type to the slowest, least leakage cell type. An example of a Vth map file is shown below:

<table>
<thead>
<tr>
<th>g65fp:FP</th>
<th>g65fxp:FXP</th>
<th>g65xp:XP</th>
<th>g65lxp:LXP</th>
</tr>
</thead>
<tbody>
<tr>
<td>g65fpipo:FP</td>
<td>g65fxpipo:FXP</td>
<td>g65xpi:XP</td>
<td>g65lxpi:XP</td>
</tr>
</tbody>
</table>

Figure 2. Example Vth Map File

For example the first line in figure 2 confirms the footprint equivalent swaps for the following buffer cells; g65fp/BUFM1FP, g65fxp/BUFM1FXP, g65xp/BUFM1XP and g65lxp/BUFM1LXP, and any other cells which match the defined pattern.

The power recovery utility supports multiple Vth cells and multiple libraries. In this example, there are two sets of Vth libraries. Each specifies libraries that contain footprint-equivalent Vth cells. The g65fp, g65fxp, g65xp and g65lxp libraries correspond to low Vth minimum channel length, low Vth extended channel length, standard Vth, and high Vth libraries respectfully. The corresponding Vth naming suffixes are specified for each library. The second line specifies a different set of Vth libraries.

The utility uses the order specified in this file to define the order of cells from the fastest, highest leakage cell type to the slowest, least leakage cell type. This information can be determined by the leakage values in the libraries; however, these values may not be sufficient because the utility uses the worst-case process libraries. For leakage values, the best case, high temperature libraries should be used. At smaller technologies such as 40 nm, at worst case, low temperature, the order of leakage values may not match those of best case, high temperature. This is because the leakage values are so small at this particular corner that the relative differences are negligible.

2.2 Flow

The leakage power recovery flow consists of four parts, as shown in Figure 1.

- Initial power recovery
- Speed recovery
- Transition recovery
- Capacitance recovery

The initial power recovery process identifies all clock cells and cells that have timing below the slack limit provided and marks these as dont_swap. All remaining constrained cells are looped.
on to determine if they should be swapped to achieve better leakage optimization. The utility optimizes runtime by estimating the decrease in cell delay resulting from the cell swap rather than incurring a timing update. After all the cells are processed, the cell swaps are applied and a timing update occurs. After a timing update, timing failures, transition violations, and capacitance violations will be seen. These timing failures occur due to a number of reasons, such as the following:

- Timing estimates based on limited factors (input transition, output load)
- Swapped cells with different pin capacitance and drive capability
- Crosstalk effects that are not accounted for during delay estimation

At this stage, the utility performs multiple iterations of speed recovery to repair any timing that is below the user-specified limit. Each iteration loops through the failing timing paths, swapping back the minimum amount of cells required to repair the timing while preserving the best leakage power.

After the speed recovery, the utility loops through any transition and capacitance violations that may have been introduced during the initial power recovery phase.

The utility then generates a list of information that it feeds into other PVT corners or analysis modes for speed recovery. The result is a list of cell swaps that can provide better leakage optimization while meeting the performance target.

### 2.2.1 Initial Power Recovery Phase

During the initial power recovery phase, the utility examines cells in the design and their timing slack to determine whether lower leakage cells can replace them without reducing the timing slack below the user-specified limit. The utility examines the max_rise_slack and max_fall_slack attributes on each pin in the design. These attributes contain the worst timing slack value (rise or fall) that any timing path through that pin will see. These attributes exist by default in SI analysis in releases prior to 2007.12. In 2007.12, these attributes exist if the variable timing_save_pin_arrival_and_slack is set to true. Because the utility adjusts slack on each pin based on timing estimates to avoid timing updates, a user attribute named pwr_rec_slack is used. This user attribute is defined as the minimum of the max_fall_slack and max_rise_slack attributes.
Figure 3 shows the initial power recovery flow:

Initial power recovery flow diagram:

1. Initialize "pwr_rec_slack" attribute on all pins.
2. Identify clock network cells & cells with cap/tran violations and mark dont_swap.
3. Loop on each Vth celltype for each Vth_cell_type [lv1, svt1, lv2, ...] {}.
4. Get all cells that match current Vth with slack > slack limit.
5. Sort cells by worst slack.
6. Process each of the Vth cells:
   - Get cell parameters (close/load).
   - Estimate delays/slack for next level Vth cell.
7. If slack estimate > slack limit, yes.
   - Swap cell and update "pwr_slack_attribute" on fan-out/fan-in of cell.
   - Finished processing Vth cells, yes.
8. If slack estimate ≤ slack limit, no.
   - Next Vth celltype in loop.

Figure 3. Initial Power Recovery Flow Diagram
The utility initializes every pin in the design with an attribute called pwr_rec_slack. This attribute contains the worst timing slack value (rise or fall) that any timing path through that pin will incur. For example, in Figure 4 below, there are two timing paths that go through the output pin of instance U1. One is from F1 to F2 with a timing slack of 0.180 ns, and another is from F1 to F3 with a timing slack of 0.320 ns. Because the worst timing slack through the output pin of U1 is 0.180 ns, its pwr_rec_slack attribute is set to 0.180 ns. The output pin of U3 has a worst timing slack set to 0.320 ns.

![Figure 4. Example pwr_rec_slack Attributes](image)

After the utility initializes the design with these pwr_rec_slack attributes, it processes every cell that is not marked dont_swap. Cells marked dont_swap are cells that it does not alter, such as clock network cells, cells that have an initial starting timing slack below the specified slack limit, or cells that are unconstrained. An unconstrained cell does not contain a timing slack value because it is constrained in another mode of analysis.

The utility groups cells according to their voltage threshold type. It loops on each group of Vth cell types and sorts by worst slack before processing the cells. When each cell is processed, the utility identifies the cell type, input transition ramp time, and output load capacitance. Using these parameters the utility estimates how much the cell delay will increase if it is replaced with the next level lower leakage Vth cell. If the estimated delay increase results in a slack that is still above the user specified slack limit, the utility schedules the swap to be performed.

The utility then recursively processes all the pins in the transitive fan-out and fan-in of the current cell and updates the pwr_rec_slack attributes to reflect the estimated increase in delay. If the current cell is a sequential cell, only the fan-out pwr_rec_slack attributes are updated. Within PrimeTime SI an accurate update of the slack attributes incurs a costly timing update. Instead, the utility updates a user slack attribute in a way that achieves maximum power recovery and so does not incur the runtime of a timing update. For example consider that U3 in the previous figure is being swapped with a slower Vth cell and is being slowed down by 0.050 ns. It would not be correct to adjust the pwr_rec_slack value on the output pin of U1 because its worst path is still the path from F1 to F2 with a slack of 0.180 ns. Updating the slack attribute on instance U1 may limit the power recovery on U1. Any pessimistic assumptions made during the initial power recovery phase will be addressed in the subsequent speed recovery phase.
Consider a more complex example circuit as shown in Figure 5.

![Complex Example Circuit](image)

**Figure 5. Complex Example of pwr_rec_slack Attribute Adjustment**

The `pwr_rec_slack` attributes for all pins are shown in black and assume the slack margin is 0.0 ns (power recovery for all nonfailing paths). Consider that the utility is currently processing the U5 instance, and the delay increases by 0.020 ns due to swapping this cell. Because the adjusted slack value is still greater than the slack limit, the swap is scheduled. Now the `pwr_rec_slack` attributes are updated on the fan-in and fan-out of this cell. These updated `pwr_rec_slack` attributes are shown in blue below the original values.

The utility updates the attributes on U5, U6, and U7 for the fan-out pins, as well as the input pin of U9. However, it does not modify the output pin of U9 and any downstream pins from there. This is because a modification on U5 does not affect the worst path through the U9 output pin. Similarly, the utility modifies the slack attributes for the fan-in pins. Note that the utility does not modify the pins of U1. This is because the worst path through U1 does not pass through U5. Without performing a timing update, the utility cannot determine if the new worst path through U1 will pass through U5. Therefore, the utility changes cells to obtain the maximum leakage recovery. It repairs any changes resulting in timing violations in the subsequent speed recovery phase.

The initial power recovery phase results in a list of cell swaps that are implemented. The utility then updates the design timing. This update will cause timing violations, transition violations, and capacitance violations due to the following:

- Pessimistic `pwr_rec_slack` adjustments to allow maximum power recovery
- Timing estimates based on limited factors (input transition, output load)
- Swapped cells with different pin capacitance and drive capability
- Crosstalk effects that are not accounted for during delay estimation

At this stage, the utility performs multiple iterations of speed recovery to repair any timing that is below the user specified limit.
2.2.2  Speed Recovery Phase

The speed recovery phase processes failing paths to perform footprint-equivalent Vth cell swaps to repair the timing of the design while preserving the best leakage power. The utility processes failing paths for each clock group and sorts them by worst timing slack. It also takes into consideration cells that are crosstalk aggressors of crosstalk victim nets. These cells that are crosstalk aggressors need to be handled differently to minimize introducing additional crosstalk delay variation on victim nets that can deteriorate timing. During the speed recovery phase, a cell is swapped only if it was changed previously during the initial power recovery stage. This restriction is to ensure that hold violations are not introduced.
Figure 6 shows the speed recovery phase flow.

Figure 6. Speed Recovery Flow Diagram
Figure 7 shows an example circuit used to explain the speed recovery phase.

In this circuit, each cell type is labeled either H, S, or L:

- An H denotes a high-voltage threshold cell that is the slowest, least leakage cell type.
- An S denotes a standard-voltage threshold cell that is faster and has more leakage than the H type.
- An L denotes a low voltage threshold cell that is faster and has more leakage than the S type.

The utility can support any number of Vth cell types, but for this example we will limit it to these three. The cell types at the start of speed recovery are labeled inside the instances.

Prior to processing the failing paths of the circuit, the utility does an analysis to identify the largest crosstalk aggressors with victims that are involved in failing timing paths. It uses this analysis as a cost factor when processing paths to determine which cells to swap for faster cells with higher leakage. This limits swapping aggressor cells that affect many victim nets which would result in timing degradation on the paths these victims are involved in.

In the example circuit, the worst timing path is from FF1 to FF4, with a timing slack of -0.500 ns. The next-worst path is from FF1 to FF5, with a timing slack of -0.430 ns, and so on. Note that certain endpoint flops such as FF5 and FF6 have multiple timing paths from different starting points. For example, FF5 has two timing paths, one from FF1 and one from FF2. The utility loops on failing timing paths and sorts them by the worst timing slack. When processing the
worst timing path, the utility loops through each pair of Vth cell types. For this example, the pairs would be H to S and S to L. When processing the FF1 to FF4 path, the utility collects all the type H cells. It sorts these cells in descending order of cell delay. It puts any cells identified earlier as crosstalk aggressors to the bottom of the sorted list. For example, instance U2 is an aggressor to the net driven by instance X1, and it is therefore considered last for cell swapping to avoid increasing the aggression. The utility then processes each of these cells to get its input transition and output load. Based on these parameters, it obtains an estimated delay for the next faster, higher leakage cell type. It then adjusts the timing slack by the delay improvement of this swap. It processes additional cells unless the timing slack becomes greater than the slack limit. This processing ensures the minimum number of swaps to higher leakage cells to meet the timing performance target.

The utility stores the delay improvement estimate on the output pin of the cell scheduled to be swapped. Thus, if this cell is involved in other timing paths, the utility adjusts the slack before processing any new cells in the timing path. For example, while processing the FF1 to FF4 path, the utility marks U1 to be swapped, resulting in a 0.050 ns faster delay on U1. It stores this delay on the U1 output pin. When it processes the FF1 to FF5 path, the utility ascertains if any cells have been swapped from a previous path and adjusts the slack by the delay improvement. In this case, the utility adjusts the slack value by the 0.050 ns improvement from U1. When the utility is processing cells in a path, it swaps a cell only if that cell was changed previously during the initial power recovery stage. This is to ensure that hold violations are not introduced.

After the utility processes all the H cells in the path being examined, it processes all the next cell type (S cell type). The utility only changes cell types by one level. This is because it is working on delay estimates and requires a timing update to get an accurate assessment of the timing performance of the design. After the failing paths are processed, the scheduled swaps are performed and a timing update occurs.

After the initial iteration of speed recovery, the circuit may have the cell mix indicated by the labels above the cell instances in Figure 7. The new endpoint slacks are shown to the right of the arrows in the figure.

The utility has repaired the timing on many paths, such as that going to FF4. The path from FF1 to FF5 has improved but needs additional cell swaps to L type cells. This will occur in subsequent speed recovery iterations. Other paths may not have been processed because the limit of failing paths per endpoint has been exceeded for the current iteration.

The utility must run multiple iterations of the speed recovery routine in order to repair the entire timing of the design. In each iteration, it must carefully choose the number of failing paths processed. Processing all failing paths may consume too much runtime and lead to diminishing improvement if many of the cells in the failing paths have been processed earlier. This can also be design-specific as some designs may have deep combinational logic (such as multiplexing) to specific endpoints. The utility has two approaches for determining the number of failing paths to process. In one case, it collects failing paths based on a limit of maximum paths per clock group and a limit of paths per endpoint flop (these parameters can change based on type of design). In
another case, the utility collects failing paths based on the start-flop to end-flop pair connectivity. In this case, it obtains only one path per start-flop to end-flop pair. A large number of flops in a design can produce a large number of paths. The utility alternates between these two methods of collecting failing paths and adjusts the limits based on the timing state of the design. This allows a robust method of path collection to cover various design structures.

After all the speed recovery iterations are complete, the timing should be repaired to the slack limit or the original timing state of the design. In some cases, due to crosstalk timing window changes from the cell swaps, additional crosstalk delay variation may be seen on certain paths. For example, a path may have been put back to the identical cell types prior to the leakage power recovery utility being run but may fail timing due to additional crosstalk delay variation. To handle this situation, additional speed recovery iterations can be run ignoring the function of only changing cells that were swapped originally during the initial power recovery stage. This will repair any remaining timing issues, but hold timing must be checked to ensure a hold issue is not introduced.

2.2.3 Capacitance and Transition Recovery Phase

After every iteration of speed recovery is completed, the utility identifies any transition and capacitance violations introduced by cell swapping during the initial power recovery phase. It swaps the driver cells on transition violations back to cells that have sharper transition times. Similarly, it changes cells with maximum capacitance violations back to cells that can drive a larger load.

2.3 Flow Outputs

The output of the leakage power recovery utility run at a single corner is a list of cell swaps. This list is generated in heco format using the write_astro_changes command within PrimeTime SI. This heco file is converted into size_cell commands for subsequent speed recovery runs at other corners or modes.

The write_changes command in PrimeTime SI should not be used to generate the cell swap file. This command writes out every change done during the PrimeTime SI run, unlike the write_astro_changes command, which only generates commands for cells that are different from the start of the PrimeTime SI run. This results in a large difference in file size because many cells are swapped during the initial power recovery stage and then swapped back during the speed recovery phase.

2.4 Distributed Multiscenario Analysis (DMSA) Flow

One limitation of the flow described previously is that each PVT corner must be sequentially analyzed to ensure that there are no timing, transition, or capacitance violations in the specified corner. To address this issue, the utility can be run in a DMSA environment.
Figure 8 shows the current DMSA flow.

![DMSA Flow Diagram]

Figure 8. DMSA Flow Diagram
In DMSA, the utility runs the initial power recovery stage at a single corner, usually at the slowest timing corner. It then runs the speed recovery phase in parallel for each scenario, merging the cell swap results of each speed recovery and applying them to each scenario. This process runs for multiple iterations until no cell swaps remain in each scenario shown in Figure 8. After each iteration of speed recovery, transition and capacitance recovery is performed in a similar manor.

This approach provides the best runtime because each slave is processing the speed recovery phase in parallel. However, the power recovery may not be as good as in the sequential approach because changes in one scenario may reduce the number of changes needed in other scenarios. Future work will use the merged timing commands available in PrimeTime SI DMSA to run the speed recovery phase to achieve optimal power recovery results.

### 3.0 Features

#### 3.1 Hierarchical Design

The leakage power recovery utility can be used within a hierarchical design environment. A tcl list of block instance names can be specified, in which changes to cells are not allowed. The utility can thus run at top level without changing cells internal to hierarchical hard macros that are timing closed. In addition, the utility will not modify any instance that contains a `dont_touch` attribute. Therefore, users can apply a `dont_touch` attribute on any instance that should not be modified.

#### 3.2 Multiple Library and Vth Cell Support

The utility supports any number of libraries and Vth cells. The user must provide a Vth map file that lists the different Vth library names and Vth suffixes in order from largest leakage (fastest timing) to smallest leakage (slowest timing).

#### 3.3 Interface Timing Paths

The leakage power recovery utility contains a feature that does not allow changes to cells in IO paths. This feature is used at the block level where the interface timing should not be altered.

#### 3.4 Controlling Speed-Recovery Paths to Process

Users can specify options such as `–group`, `–max_paths`, and `–nworst` to specify a list of clock groups, number of timing paths, and number of paths per endpoint respectively for the speed recovery phase. By default, the utility works on all clock groups and varies the other options to maximize the timing repairs while minimizing runtime.

The `–slack_margin` option provides a slack limit. The utility will not modify cells where the change will result in timing slack less than the specified limit. The slack limit value will affect the number of speed recovery paths processed.
3.5 Intermediate Files

The utility contains options to generate intermediate files after each timing iteration. The following intermediate files can be generated:

- Quality of results report
- size_cell commands
- save_session

3.6 Distributed Multiscenario Analysis (DMSA)

The leakage power recovery utility can run in the PrimeTime SI DMSA environment across multiple PVT corners and multiple modes. Refer to Section 2.4 for details of the DMSA flow.

4.0 Limitations

4.1 Unconstrained Cells

The current implementation of the leakage power recovery utility only runs the initial power recovery phase at a single corner and mode. It does not process any unconstrained cells. An unconstrained cell does not contain a timing slack value because it is most likely constrained in another mode of analysis. Therefore, power recovery will not occur on these unconstrained cells.

4.2 Identifying Clock Network Cells

The utility uses the PrimeTime SI command `get_clock_network_objects` to identify clock network objects that should not be altered. There is a limitation with this command for clock objects that are between generated clocks on a cascaded divider. For example, consider the two divider flops shown in Figure 9. The generated clocks are defined on the output of FF1 and the output of the U2 MUX. PrimeTime SI does not include U1 pins or the input pin of U2 as clock network pins. STAR 9000239385 has been filed on this issue. The utility includes some workarounds for this issue using the `get_timing_paths` command. However, this may not get all the clock logic since it only provides access to the worst clock path.

![Figure 9. Generated Clock Issue Example](image-url)
4.3 Incremental Timing Update Runtimes

Based on the number of cell swaps in a given iteration, the tool may trigger a full timing update. At later iterations, fewer cell swaps occur and the tool performs a faster incremental timing update. However, it has been observed that when one of the cell swaps involves a sequential cell, the runtime of the timing updates increases by an order of magnitude. Due to this limitation and to benefit from the faster updates, it may be advantageous to process sequential cells in earlier iterations where full timing updates occur anyway.

5.0 Experiments

The power recovery utility was run on a number of blocks obtained from various designs. In addition, it was also run on a block used internally to verify and test the design flow. Some of the runs were at a single PVT corner. Others were run sequentially across multiple PVT corners to ensure that no timing violations or transition/capacitance violations occur at the completion of the power recovery run. Finally the DMSA version of the utility was run over multiple PVT corners.

The power recovery utility is run at slow process because this is the worst-case process for timing. Because of temperature inversion [2], cell delays may be slowest at the lowest temperature (-40 °C) or the highest temperature (125 °C) depending on the mixture of cell types in a particular timing path. For the data that follows in the subsequent sections “COLD” will refer to slow process -40°C and “HOT” will refer to slow process 125°C.

In the sections that follow, leakage power was measured using the report_power command in PrimeTime PX.

5.1 Internal Test Block

The first experiment was run on a 150K 65 nm test block containing multiple arithmetic and logical designware components. This block contains four types of footprint equivalent cells (high Vth, standard Vth, low Vth extended channel and low Vth min channel). The initial run was performed at the temperature inverted corner (COLD). A slack margin of 0.0 ns was used (work on all passing timing paths). The swapped cell info was then loaded into the HOT PVT corner, and additional speed recovery, transition, and capacitance recovery phases were run to repair any violations at this corner.

The starting netlist was optimized for power within ICC at all stages of the flow. This produced much lower leakage power compared to running leakage optimization only at the end because it enabled restructuring of the design, particularly within a high-effort place_opt. The high effort comes at a runtime expense.
An additional experiment was run using the DMSA version of the power recovery utility to run concurrently across these two PVT corners. The results of all runs are shown in table 1.

<table>
<thead>
<tr>
<th>Test 65 nm block</th>
<th>150K Placeable Elements</th>
<th>2007.12.sp2</th>
<th>Slack Margin 0 ns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Power data</td>
<td>Runtime</td>
<td># Timing Updates</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>PVT Stage</td>
</tr>
<tr>
<td>Original</td>
<td>HVt=73.2% SVt=13.7% LVt_ext_ch=5.6% LVt_min_ch=7.5%</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Static power=0.0951W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Recovery Utility</td>
<td>HVt=83.1% SVt=14.3% LVt_ext_ch=2.1% LVt_min_ch=0.47%</td>
<td>4 hrs 54 min</td>
<td>23</td>
</tr>
<tr>
<td></td>
<td>HVt=82.6% SVt=14.1% LVt_ext_ch=2.9% LVt_min_ch=0.5%</td>
<td>1 hrs 40 min</td>
<td>39</td>
</tr>
<tr>
<td></td>
<td>Static power=0.0551W percent reduction=42.06%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of cell swaps = 22974</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMSA Power Recovery Utility</td>
<td>HVt=82.9% SVt=13.8% LVt_ext_ch=2.6% LVt_min_ch=0.8%</td>
<td>5 hrs 37 min</td>
<td>44</td>
</tr>
<tr>
<td></td>
<td>Static power=0.0553W percent reduction=41.85%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of cell swaps = 23233</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Internal test block Power Recovery experiment Data

Both the sequential and DMSA runs achieved about a 42% leakage reduction. In both cases there are no timing, transition or capacitance violations at the completion of the run. The runtimes shown are runtimes of the power recover utility in the PrimeTime SI tool. There is additional runtime for the sequential run to load the second run at the HOT PVT corner.

This particular design uses many arithmetic and logical designware components. As a result, there are many paths to any particular endpoint, and many timing iterations of speed recovery are required. However, a majority of these complete within minutes because these are incremental timing updates.

5.2 DSP Block

The power recovery utility was next run on a 65 nm DSP block used on a networking standard product design. This block contains three types of footprint equivalent cells (high Vth, standard Vth, and low Vth). The utility was run both sequentially using the COLD and the HOT PVT corners and then concurrently with the DMSA version. A slack margin of 0.0 ns was used (work on all passing timing paths).
The results of these runs are shown in table 2.

<table>
<thead>
<tr>
<th>DSP 65 nm LP Block</th>
<th>595K Placeable Elements</th>
<th>2007.12.sp2</th>
<th>Slack Margin = 0 ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Original</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power data</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HVT = 54.233% SVT = 7.338% LVT = 38.429%</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Static power = 0.3330 W</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Recovery Utility</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HVT = 62.486% SVT = 23.616% LVT = 13.897%</td>
<td>19 hrs 52 min</td>
<td>32</td>
<td>COLD</td>
</tr>
<tr>
<td>HVT = 61.359% SVT = 22.144% LVT = 16.497%</td>
<td>8 hrs 51 min</td>
<td>26</td>
<td>HOT</td>
</tr>
<tr>
<td>HVT = 61.358% SVT = 22.144% LVT = 16.498%</td>
<td>0 hrs 50 min</td>
<td>2</td>
<td>COLD</td>
</tr>
<tr>
<td>Static power = 0.1951 W percent reduction = 41%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of cell swaps = 135576</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMSA Power Recovery Utility</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HVT = 61.187% SVT = 21.111% LVT = 17.702%</td>
<td>23 hrs 28 min</td>
<td>44</td>
<td>COLD &amp; HOT</td>
</tr>
<tr>
<td>Static power = 0.2053 W percent reduction = 38%</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Number of cell swaps = 128419</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 2. DSP 65 nm Power Recovery experiment Data

Notice that for the sequential run, timing violations had to be repaired at COLD, HOT and then again at COLD. This is because some of the cell swaps applied at the HOT corner caused timing violations at the COLD corner. These timing violations were caused by crosstalk delta delay changes resulting from the cell swaps.

Both the sequential and DMSA runs achieved about 40% leakage reduction. For this block, the sequential run achieved a better leakage reduction than the DMSA run. In both cases, the timing, transition and capacitance violations are less than at the beginning of the run. It is not clear why the sequential run obtained better results than the DMSA run. One thought is that the results are better because of the parallel runs that are done during DMSA fixing. It is possible that during the concurrent DMSA slave runs for each scenario, a path may get over-repaired for timing. In other words, the union of changes from each scenario may be greater than the optimal subset of changes. Future work will examine the use of merged timing in DMSA to overcome this issue.

...
During the power recovery run, the utility can generate intermediate files such as quality of result files to show the progress of the run. The following table shows the quality of result reports for the main clock for many of the iterations, including the reduction in worst negative slack (WNS), total negative slack (TNS), and number of failing paths at each iteration.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>Critical Path Slack</th>
<th>Total Negative Slack</th>
<th>No. of Violating Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-0.5260</td>
<td>-733.9975</td>
<td>6173</td>
</tr>
<tr>
<td>2</td>
<td>-0.3538</td>
<td>-175.8084</td>
<td>1601</td>
</tr>
<tr>
<td>3</td>
<td>-0.2895</td>
<td>-83.1979</td>
<td>1158</td>
</tr>
<tr>
<td>4</td>
<td>-0.2444</td>
<td>-41.5463</td>
<td>905</td>
</tr>
<tr>
<td>5</td>
<td>-0.2022</td>
<td>-20.0050</td>
<td>601</td>
</tr>
<tr>
<td>6</td>
<td>-0.1710</td>
<td>-12.9097</td>
<td>406</td>
</tr>
</tbody>
</table>

Table 3. Quality of Results Data for various iterations of DSP experiment
5.3 Storage Read Channel Block

The power recovery utility was next run on a 65 nm read channel block used on a storage SOC design. This block contains three types of footprint-equivalent cells (high Vth, standard Vth, and low Vth). The utility was run both sequentially using the COLD and the HOT PVT corners and concurrently with DMSA.

The results of these runs are shown in table 4.

<table>
<thead>
<tr>
<th>Storage 65 nm block</th>
<th>665K Placeable Elements</th>
<th>2007.06-SP3</th>
<th>Slack Margin = 0.0 ns</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Power data</td>
<td>Run Time</td>
<td># Timing Updates</td>
</tr>
<tr>
<td>Original</td>
<td>HVt = 24.31% SVt = 18.89% LVt = 56.80%</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td></td>
<td>Static power = 0.6459W</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Power Recovery Utility</td>
<td>HVt = 32.21% SVt = 42.91% LVt = 24.88%</td>
<td>39 hrs 21 min</td>
<td>27</td>
</tr>
<tr>
<td></td>
<td>HVt = 30.41% SVt = 42.51% LVt = 27.08%</td>
<td>23 hrs 48 min</td>
<td>20</td>
</tr>
<tr>
<td></td>
<td>Static power = 0.4260W percentage = 34.05%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of cell swaps = 196510</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DMSA Power Recovery Utility</td>
<td>HVt = 31.89% SVt = 39.02% LVt = 29.08%</td>
<td>31 hrs 35 min</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>Static power = 0.4403W percentage = 31.83%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Number of cell swaps = 190797</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 4. Storage 65 nm Power Recovery experiment Data

Both the sequential and DMSA runs achieved about a 32% leakage reduction. In both cases, the timing, transition and capacitance violations are less than at the beginning of the run. The runtimes shown are runtimes of the power recover utility in the PrimeTime SI tool. There is additional runtime for the sequential run to load the second run at the HOT PVT corner. This run was done using the 2007.06.sp3 version of Primetime SI. This is because that is the version used by the design team. The runtimes are larger for this block as opposed to previous ones. The 2007.12.sp2 version seems to run much faster compared to the 2007.06.sp3 version.

6.0 Conclusions and Recommendations

This paper has presented a utility to recover leakage power that runs in the PrimeTime SI signoff environment by performing footprint-equivalent cell swaps on cells with positive timing slack. A substantial improvement in leakage power has been obtained on the example blocks run through this utility.
It is recommended that the utility be used when a design or block is very close to being timing closed. Since the utility works on cells with positive timing slack and does not alter the routing, it can be used during the final ECO iterations of a design.

Based on the number of PVT corners and modes that must be analyzed, it is recommended to run the power recovery utility in PrimeTime SI DMSA. This avoids having to run sequentially at each PVT corner and mode. It also reduces the overall runtime. It is also recommended to use models such as Interface Logic Models (ILM) for each physical block when running at top level to reduce runtimes.

6.1 Future Work

The current implementation of the power recovery utility runs the initial power recovery stage at a single PVT corner and mode. This has a limitation of not recovering leakage power in other modes such as test. Future enhancements will examine all modes in DMSA at the initial power recovery stage.

In addition, merged timing will be examined in DMSA to see if better leakage recovery can be obtained. The issue with this is runtime. For optimal runtime in DMSA, the handshaking between the master and the slaves needs to be minimized.

In 2008.06 PrimeTime SI release a new command, estimate_eco was introduced to provide estimated slack values based on netlist changes. The utility will be modified in the future to use this new command.

7.0 Acknowledgements

The author would like to thank Gerard M Blair for assistance in the development of the leakage recovery utility and the internal document, Thomas Wilderotter and Chris Papademetrious for their excellent support on PrimeTime SI issues during the development of the utility.

8.0 References