

Dual threshold voltages and power-gating design flows offer good results

DUAL THRESHOLD VOLTAGES AND POWER-GATING DESIGN FLOWS MANAGE BOTH LEAKAGE POWER AND PERFORMANCE WITH LITTLE EFFORT.

Design-optimization methodologies and flows that use gates with two threshold voltages (V_{TH}) can achieve excellent results for both power and timing with a high degree of automation. This dual- V_{TH} approach has become crucial for VDSM (very-deep-submicron) chips, in which reduced V_{TH} not only improves performance, but also increases static (leakage) power.

In fact, leakage power increases exponentially with the technology scaling and reaches 50% of chip power at 65 nm. This dramatic increase in leakage power is unacceptable for most designs, whether or not they run from battery power. As a result, most designs can benefit from design-optimization flows that balance the trade-offs between performance and leakage power.

There are three popular flows for optimizing performance and leakage power based on design requirements. These flows target minimum leakage, best performance, and optimum chip area and tool runtime for a design in power-on mode. Because the design still consumes leakage power in standby mode, the flows also include methods for minimizing standby leakage power.

THREE FLOWS FOR MANAGING LEAKAGE POWER

Dual- V_{TH} methodologies rely on the use of two cell libraries—one with low- V_{TH} cells that have smaller propagation delay and higher leakage power, and the other with higher V_{TH} cells that have larger delay and lower leakage. Optimizing a design with the low- V_{TH} cells in critical timing paths and high- V_{TH} cells in noncritical paths can maximize speed and minimize leakage power.

The quality of this optimization depends significantly on the identification of true timing-critical paths and accurate calculation of the two libraries' timing impact on a path. To achieve the necessary timing accuracy, path-delay calculations need to include interconnect delays based on cell placement and net-routing information. Thus, physical synthesis is highly recommended for the second-pass mixed- V_{TH} design optimization in the following three flows:

- The min-cut flow achieves the lowest leakage power of the three flows but has higher cell count and dynamic power as well as lower performance.
- The max-cut flow results in the highest performance and

lowest cell count and dynamic power but has the highest leakage power of the three flows.

- The max-cut II flow provides a compromise of the first two approaches with a good balance between leakage power and chip area. This flow also minimizes tool runtime and capacity issues.

The first flow uses an iterative min-cut algorithm, in which all cells in a combinational circuit are initially assigned a high threshold voltage. Due to the performance degradation of the high- V_{TH} transistors, the design usually violates delay constraints. But the initial design has the lowest leakage power. Next, the algorithm identifies a minimum subset of edges, in which changing threshold voltages to low improves the performance and meets the delay constraints. A min-cut graph algorithm based on minimum-weight cut identifies these edges.

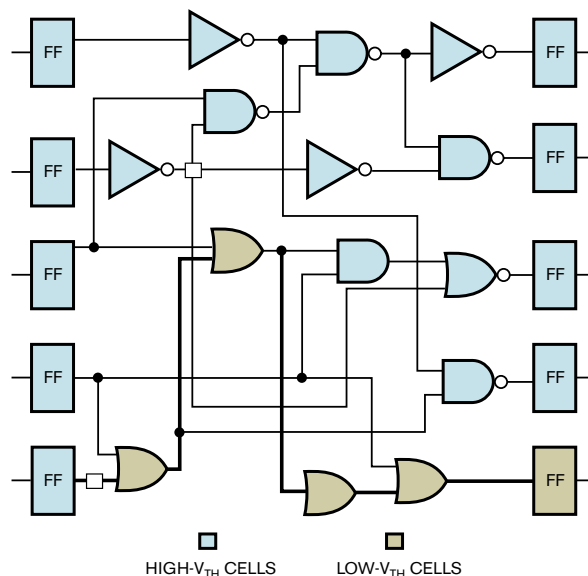


Figure 1 In this example of the min-cut approach to reducing leakage-power consumption, the synthesis tool has swapped logic gates in the critical paths to low- V_{TH} gates to meet timing constraints.

This cut corresponds to the smallest power increase as a result of changing to low V_{TH} . **Figure 1** shows an example of the dual- V_{TH} assignment using the min-cut algorithm.

Implemented as a leakage-centric design-optimization methodology and flow, the min-cut approach generates a design with minimum leakage-power consumption. The right half of **Figure 2** depicts the flow. After the first optimization using the high- V_{TH} library, the design usually has timing violations. Incremental optimization using both low- and high- V_{TH} cell libraries identifies all timing-critical paths and replaces cells in these paths with low- V_{TH} cells. The synthesis tool also performs local design optimizations to fix timing violations in paths that still fail after low- V_{TH} -cell swapping.

Unlike the iterative min-cut algorithm, the iterative max-cut algorithm initially assigns all low- V_{TH} cells in a combinational circuit. Because low- V_{TH} cells are fast, this implementation meets the defined delay constraints with a positive margin but at the cost of high leakage-power consumption. Next, the algorithm identifies a maximum subset of edges, in which changing to high V_{TH} reduces leakage power without causing delay-constraint violations. The max-cut graph algorithm identifies these edges based on maximum-weight cut, which corresponds to the maximum-leakage-power reduction as the result of changing to high V_{TH} . The implementation of the algorithm, which the left half of **Figure 1** depicts, works in essentially the same way as the previous flow but with low- and high- V_{TH} libraries used in reverse order.

The third approach is a variation of the iterative max-cut algorithm (referred to as max-cut II) that improves on runtime and capacity. The max-cut II algorithm starts by assigning all high- V_{TH} cells to the design and then identifies critical subcircuits that violate timing constraints. All cells in the critical subcircuits are changed to low V_{TH} to meet the timing constraints. Next, the regular max-cut algorithm identifies edges that can tolerate a change back to high V_{TH} without causing delay violations. As a result, max-cut II essentially applies the max-cut algorithm only to critical subcircuits—greatly reducing the size of the optimized circuit.

This size reduction allows the flow implementation of the max-cut II algorithm to reduce chip area and optimization runtime compared with the regular max-cut approach (**Figure 3**). The max-cut II flow optimizes the design using the high- V_{TH} library for minimum leakage-power consumption. By lowering the clock frequency, however, you get an optimization with relaxed timing constraints to avoid working on timing-critical paths that cannot meet timing with only high- V_{TH} cells. After this initial optimization, you adjust the clock frequency up to its true target value to restore the actual timing constraints and then incrementally optimize the design using both low- and high- V_{TH} libraries.

POWER GATING

In standby mode, your design continues to consume leakage power—whatever amount you achieved with one of these optimization methods. One way to reduce leakage-power consumption in standby mode is to shut off the power supply to sections of logic. Because you can power down some logic while keeping other logic active, you must partition the design into two or more power islands. Then, you can gate

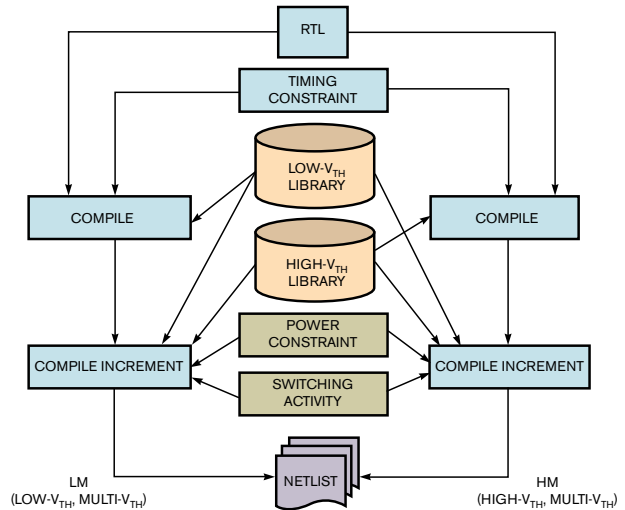


Figure 2 The options for the min- and max-cut flows appear on this flow chart's right and left sides, respectively. The only difference is that one begins with minimum leakage (high- V_{TH} cells) and the other begins with minimum delay (low- V_{TH} cells).

the power to the appropriate islands for powering down.

Although this power-gating approach achieves ultralow leakage power, the necessary methodologies are complex. They require the following special tasks in a design flow:

- Adding sleep transistors to shut off power supplies to idle circuits;
- Distributing power-gating control signals;
- Synthesizing power-island-aware clock trees;
- Retaining states of a subdesign in power-down periods;
- Isolating power-island interface signals; and
- Optimizing the multi-power-island physical design.

You can implement sleep transistors in a power-gating design based on gates, clusters, or power domains. In the gate-based implementation, you insert the sleep transistors into a logic gate to control power supplies to the gate. This approach offers the advantage that each gate can have the optimal-sized sleep transistor based on the gate's switching current and power-noise margin. Additionally, the virtual power nets are short and hidden in the gate, and you can implement the gates using standard-cell-based synthesis and physical-design tools. On the other hand, every gate has a sleep transistor, which increases area more than the cluster- and power-domain-based implementations. Distributing the global sleep-control signal to every gate is challenging in physical design.

In the cluster-based sleep-transistor implementation, you group the gates of a design into clusters. Choose a minimal number of clusters and minimal simultaneous switching current. The gates in a cluster belong to the same power domain and are close to each other in the layout. Each cluster connects to a virtual power net through a sleep transistor. The cluster-based implementation usually requires less area than the gate-based implementation, but obtaining realistic current estimates for each cluster to enable accurate sleep-transistor sizing is difficult. Moreover, the IR-drop variations among these individually powered

clusters result in performance variations.

In the power-domain-based sleep-transistor implementation, you connect gates in a power domain to a virtual power network that connects to a real power network through a number of sleep transistors. You can optimize the virtual power network in the same way as the real power network using normal power planning, analysis, and optimization methods. This sleep-transistor implementation usually requires less area than the gate- and cluster-based implementations because all gates in a power domain receive power via the distributed-sleep-transistor network in which the transistors share and balance the current discharge. Be aware that the virtual ground net tends to reduce the design's noise margin and can cause functional failure, so you have to carefully analyze the power needs of a power-gating design to ensure power integrity.

Whatever sleep-transistor approach you use, take care in routing the power-gating control signals to the transistors so that powering down a power island does not block the control-signal propagation to other power islands. A special buffer-tree-generation method can constrain creation and placement to ensure buffers are under the always-on power and ground nets. For similar reasons, constrain clock-tree synthesis to ensure that no clock branch passes through a power island, the shutdown of which would prevent the clock from getting to downstream logic.

STATE RETENTION AND RESTORATION

Designs that resume operation after wake-up need a method for saving a power island's logic state before going idle and restoring the state at wake-up. You can choose among three types of

MORE AT EDN.COM

+ We encourage your comments!
Go to www.edn.com/ms4154 and click on Feedback Loop to post a comment on this article.

state-retention and -restoration methods.

The first method relies on application software that writes specified register and memory values to disk storage before shutting down power supplies. At wake-up, the software writes the saved states back into the design. Although this method requires no special hardware

design, the retention and restoration processes usually take too long to be useful in real applications. Moreover, the disk reads and writes consume a lot of power—possibly enough to cancel the leakage-power savings of short-term standby.

The second state-retention and -restoration method uses a design's scan chains to shift out register states into an always-powered memory before power-down and shift-in of the states at wake-up. Although much faster than the first method, this second method is still too slow for many leading-edge applications, and memory access still consumes too much power.

In the third method, you implement retention registers and latches that can efficiently store and restore states. Among the various types of retention circuits is the balloon-style circuit that is common in low-power designs. To implement such a circuit, you add a shadow latch to a normal register or a latch to save the register's state in power-down. The shadow latch comprises high- V_{TH} transistors for low leakage and connects to the normal register or latch through a single point like a tied balloon.

The balloon-style retention circuit introduces two retention-control signals, B1 and B2. You need to distribute these global signals in a similar way to the power-gating control signal to ensure valid signals at the retention circuits when you power down a power island in the signal path.

ISLAND ISOLATION

When a power island powers down, its output signals float. These floating signals can affect other parts of the design that remain active. In the case of a handshake signal, a malfunction may occur. In any case, a floating signal may cause large short-circuit current in a receiving logic gate, resulting in wasted power and even device damage.

To avoid these problems, you need to add isolation logic at the interfaces between power-down and active-power islands. You can implement this isolation logic either in a power-down island to control output signals or in an active-power island to control input signals.

Isolating output signals usually requires fewer isolation cells than the input-signal-isolation method because outputs often drive multiple inputs of other power islands. Additionally, always-on power islands require no isolation cells at their outputs. Distribution of the output isolation control signal is also simpler.

On the other hand, output-signal isolation cells usually consume more power in standby mode because they must drive a large signal net, in contrast to the short local nets needed for input isolation cells. Moreover, you can implement the input-signal isolation logic with standard cells, but the output-signal isolation must rely on custom cells.

The choice of methods also depends on a design's power-management structure. For a design with one or two gating power islands, an input-signal-isolation method usually makes the best

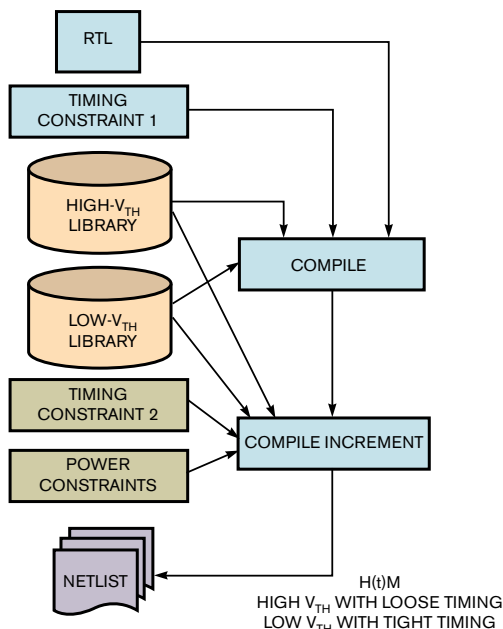


Figure 3 The max-cut II approach is the same as the max-cut flow in Figure 2, but it modifies only critical subcircuits. This approach reduces tool runtime.

sense. For designs with more complex power-management structures, the output-signal-isolation method is usually a better choice.

Various combinations of gates and transistors are suitable for achieving the necessary pullup or pulldown state under control of the power-gating signal. During placement and physical synthesis, make sure that the input isolation cells are inside the power island and close to the power-island boundary. And check the results of physical synthesis to make sure that the tool never inserts a buffer between the isolation cells and the inputs of a power island. Any such buffer defies the purpose of isolation cells.

POWER-ISLAND-AWARE DESIGN OPTIMIZATION

Sleep transistors, state-retention cells, and interface-signal isolations make design optimization a complex task. These guidelines can help achieve good results:

- Use strict exclusive region constraints during placement and physical synthesis to ensure that cells in a power island are placed inside the island. This placement also requires design specifications of logical hierarchy in association with the power island.
- Assign new cells created during restructuring logic or local buffering in a power island to the power island and apply constraints to ensure that they are inside the power island.
- Place all special cells that require always-on power supplies, such as state-retention cells, under always-on power and

ground nets. Some tools automatically handle this task.

- Place a power island's signal-isolation cells at the island's boundary to ensure that cross-power-island signals are isolated.
- Create global signal distributions and manage scan-chain stitching so that powering down a power island does not block signals or scan chains in the rest of the design.

Applying all of these constraints increases runtime, so implementing power-down islands is a useful strategy only for applications that demand the lowest possible leakage-power consumption. For most designs, the automated dual- V_{TH} methodologies provide good results with little effort. **EDN**

AUTHOR'S BIOGRAPHY

Kaijian Shi is a principal consultant with Synopsys Professional Services. He holds a bachelor's degree in physics, master's degree in computer science (Shanghai Teachers University, China), master's degree in electrical engineering (University of Birmingham, Birmingham, UK), and doctorate in electrical engineering (University of Kent, Canterbury, UK). Shi has served or is currently serving as chairman of the IEEE Dallas Section (2006), chairman of the IEEE Circuits and System Society Dallas Chapter (2004), track chairman of the IEEE SOC Conference (2005, 2006), track organizer of IEC DesignCon (2003 to 2006), and member of the technical-program committees of IEEE ASP-DAC (2005) and IEEE ISVLSI (2006). He has published nine journals and 33 international-conference papers.