

# What is Low Power Design?

[Home \(/\)](#) / [Glossary \(/glossary.html\)](#) / [Low Power Design \(/glossary/what-is-low-power-design.html\)](#)

## Table of Contents

[The Need for Low Power Design](#)

[Low Power Design Techniques](#)

[Low Power Design Methodology](#)

[Low Power Design Tools](#)

[Low Power Design and Synopsys](#)

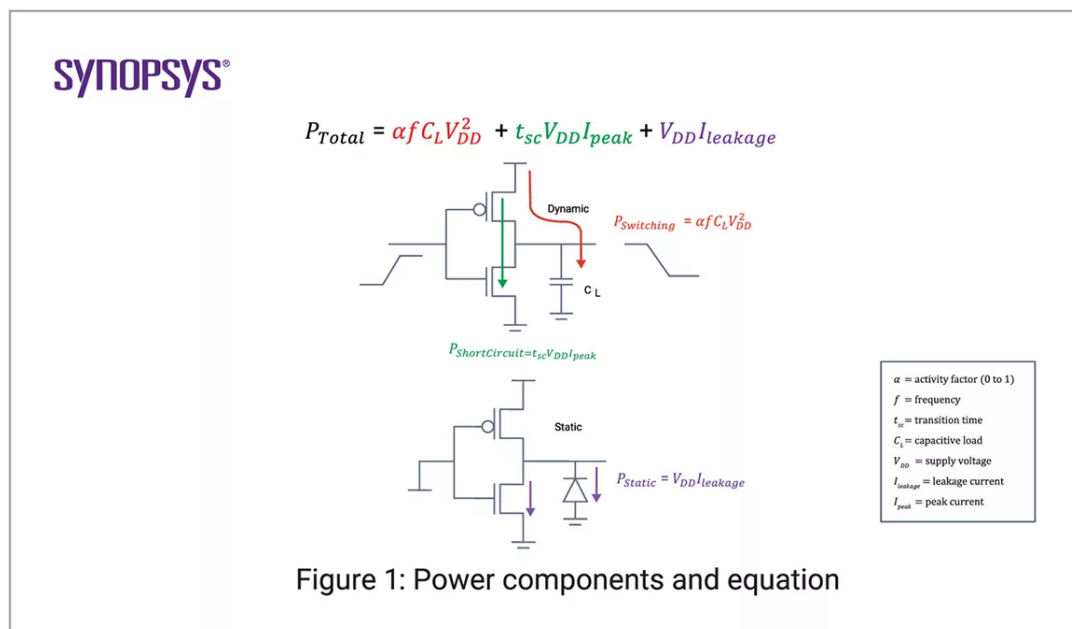
## Definition

Low power design is a collection of techniques and methodologies aimed at reducing the overall dynamic and static power consumption of an integrated circuit (IC).

Looking at the individual components of power as illustrated by the equation in **Figure 1**, the goal of low power design is to reduce the individual components of power as much as possible, thereby reducing the overall power consumption. The power equation contains components for dynamic and static power. Dynamic power is comprised of switching and short-circuit power; whereas static power is comprised of leakage, or current that flows through the transistor when there is no activity. The value of each power component is related to any of the following factors:

- Activity
- Frequency
- Transition time
- Capacitive load
- Voltage
- Leakage current
- Peak current

For example, the higher the voltage, the higher the power consumed by each component, resulting in higher overall power. Conversely, the lower the voltage, the lower the overall power. To achieve the best performance with the lowest power consumption, tradeoffs for each of these different factors are tried and tested via various low power techniques and methodologies.



Click to see the detail

## The Need for Low Power Design

Companies are continuing to push the boundaries on new features and functionality, all packed into portable, handheld, and battery powered devices. For such products, improving the battery life by minimizing power consumption is a huge differentiator and extremely important to their end users' applications. Improving the time it takes for a device to go from OFF/SLEEP state to ON/ACTIVE state is just as important, as the end user wants to have a seamless experience along with longer battery life.

For "plug-in" products, power consumption is also important because it can affect the overall cost of systems by requiring heat sinks and elaborate cooling systems, increasing electricity costs, etc. For example, in server farms, where massively parallel systems are used, a reduction in power for a single component can result in significant power savings because it is used throughout the system. The power and cost savings by upgrading these systems with newer and more power efficient ICs can be significant.

## Energy-Efficient SoCs

End-to-end solution for low power design, verification and IP from silicon to software.

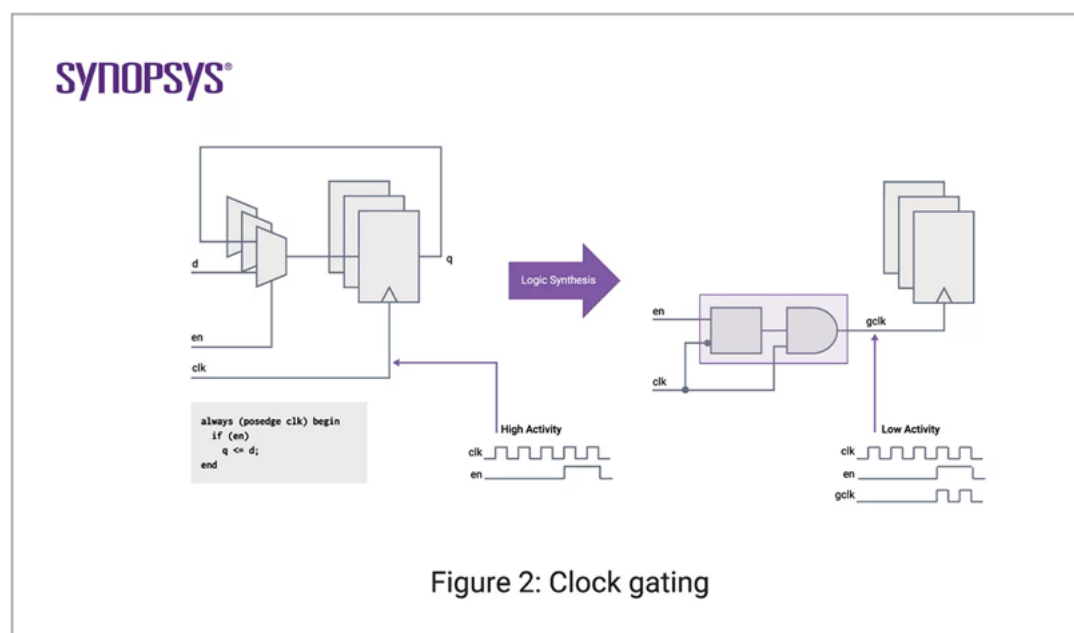
Learn More (<https://www.synopsys.com/energy-efficient.html>)

## Low Power Design Techniques

There are many low power design techniques available, some of which are very simple to use while others are more involved and complex.

### Clock Gating

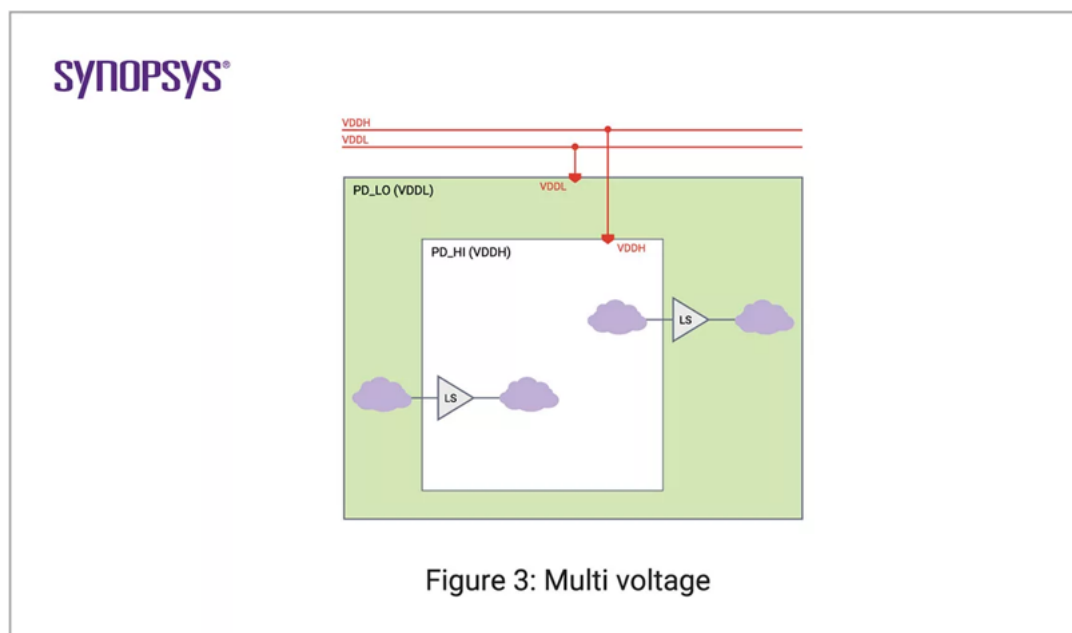
This technique is typically performed during logic synthesis where enable flops are optimized into a clock gating structure, thereby saving mux area and reducing the overall switching activity of the clock net (refer to **Figure 2**). With respect to the power equation, the goal is to reduce capacitive load (via area reduction) and activity factors which reduces the switching power component of dynamic power. This is a very simple and readily available technique to reduce power and area. However, it does rely on the logic synthesis to perform this optimization. Fortunately, this technique is well-known and well supported in most tools and flows.



🔍 Click to see the detail

### Multi Voltage

This is a technique where functions of a chip are partitioned via performance characteristics – perhaps one block is high performance, while the rest of the chip is lower performance as shown in **Figure 3**. To achieve the goals for the high-performance block, a higher voltage is typically required; while to save power on the lower performance blocks, a lower voltage can be used. This is in lieu of designing the entire block at the higher voltage, which is simpler but more power intensive. In the power equation, voltage is reduced which decreases every static and dynamic power component. With multi-voltage designs, there is the complication of designing in separate voltage islands where voltage crossings between islands may require “Level Shifter (LS) cells with the need to implement and analyze the blocks at their different voltage characteristics.



[Click to see the detail](#)

## Power Gating

This is a technique where functions on an IC are also partitioned, much like multi voltage, but this time the power supplies for the power domains are connected to power switches as shown in **Figure 4**. Power gating effectively shuts off the power completely for a block. In the power equation, this zeros out the voltage and shuts off power, resulting in both static and dynamic savings for the time that the block is turned off. Power gating typically offers the most aggressive power savings, and thus it's an ideal goal to shut off as many domains as possible, as often as possible, while maintaining functionality. In order to achieve this power savings with power gating, power switches must be implemented in the design, which requires isolation gates that clamp the boundaries of the power domain to known values when off. The power states of the design and what combination of ON/OFF states for given voltages must be considered. Lastly, a power management unit (PMU) that controls the power switch and isolation enable signals must be implemented. It is essential that the order of these signals are correct during power down and power up, such that the values during shutdown are clamped to the right values at the right time.

SYNOPSYS®

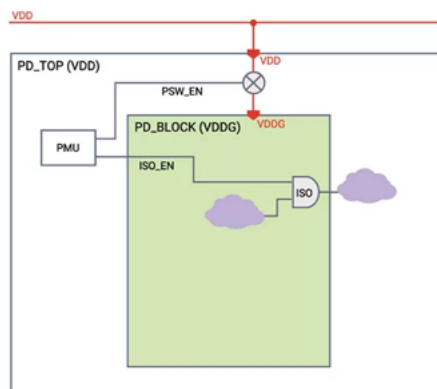


Figure 4: Power gating

[Click to see the detail](#)

## Retention with Power Gating

Retention (or register retention) is a technique used along with power gating. Here in each shutdown block when the block is OFF, either a subset of the flops or all the flops in the block have their previous values saved. When the block powers on, then the previously saved values will be restored. It is important to save the state of the block at the time it is powered off so that the block can quickly restore its previous state instead having to cycle through from an INIT state to the current state. This saves power by reducing the time and steps necessary to get the saved state, as well as improves the overall ramp up time to restore previous functionality of the block. In addition to everything needed for power gating, retention flops must exist in the library that can map to the desired registers in the RTL. SAVE/RESTORE signals will need to be added to the PMU along with the control the sequence of these signals in addition to those done for power gating (refer to **Figure 5**).

SYNOPSYS®

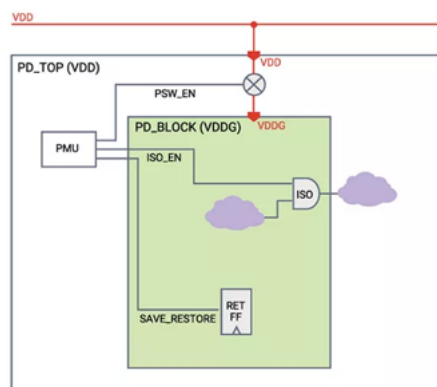



Figure 5: Retention with power gating

 Click to see the detail

## Advanced Techniques

There are many more advanced techniques for low power design, including the combination of the techniques previously mentioned. Blocks that use a lower voltage with power gating and have isolation, retention, and level shifters are commonly seen in many modern-day chips. Well biasing, zero-pin retention flops, specialized low power library cells, dynamic voltage and frequency scaling (DVFS), adaptive voltage and frequency scaling (AVFS), and custom design are just some of the other advanced low power techniques also used in the industry.

## Low Power Design Methodology

Assuming an example where the system has been specified, system simulations have been performed, microarchitecture is completed, low power choices for technology node, IP, etc., have been made, and coding of RTL and UPF are done. Given this, there are five main phases for low power design and verification methodology to be used to design the IC.

### Static Power Verification and Exploration

In static verification, the first step is to ensure the inputs to the design flow (RTL, UPF, and SDC) are structurally and syntactically correct. By definition, static verification doesn't use test vectors, so this is a very efficient way to review inputs before going into simulation or implementation flows. Lint and CDC checks are important in general to ensure your RTL is clean. UPF checks can be done either independently or with the corresponding RTL to ensure they are clean and SDC can also be statically checked along with RTL as well. In power exploration, early estimates for power for the RTL can be driven, either with estimated switching or actual waveforms from simulation. Choices can be made early on to improve the overall architecture of the design by performing early RTL power analysis.

### Dynamic Power Verification and Analysis

In dynamic power verification, there are several important aspects to check. First off, does the sequence of the PMU control signals work correctly to shut down, clamp for isolation, save, restore, remove isolation clamp, and power up. This is an extremely important check with the design RTL and UPF together to measure the design is functioning properly. Next, what type of waveforms and toggle activity are seen in the design? This will determine the dynamic power used since it depends on activity factor. The higher the activity factor, the more power is being used. Hence, the waveforms produced are very important to accurately estimate power both early and late in the process.

### Software Driven Power Analysis

For emulation-based low power flows, it's important to be able to capture the right peak windows for the design's power profile. Emulation allows review of a much wider set of data, enabling one to choose the windows that would be most valuable to generate waveforms to estimate power.

## Power Implementation

RTL-based predictive power estimation, logical synthesis, DFT insertion and physical implementation all have important low power specific roles to play. RTL-based predictive power estimation allows, very early on, to make RTL modifications with early power estimates. In logic synthesis, the RTL, SDC, and UPF, not fully verified both statically and dynamically, are mapped to technology gates. Power-specific isolation, I shifter, and retention cells are mapped to gates as well, where timing, area and power are all part of the function for generating a Netlist and associated UPF'. DFT insertion occurs as well, often simultaneously during this time. Once the Netlist and UPF' are complete, another round of checks is done statically and dynamically at this level – once clean, the results are input to physical Implementation. In physical implementation, floorplanning is done with macro placement and power routing in mind. Then placement is performed where power switches are physically inserted and placed; and iterations of placement, routing estimation, logical optimizations, and clock tree synthesis are performed to once again trade off for timing, area, and power. Finally, the routing step occurs, where pre-route of the priority signals (clock, power enables, switch connections) is done followed by detailed routing of the rest of the design – all with emphasis on reducing power more granularly, while still trying to meet the timing and area targets.

## Signoff

UPF consistency should once again be checked during signoff. However, this time with the Netlist and UPF from logic synthesis and PGNetlist and UPF' from physical implementation. This will ensure that the connections and changes made to the netlist and UPF are consistent and clean, and the power intent is preserved. Logical equivalence checks comparing RTL and UPF vs. Gates and UPF' vs. PGNetlist and UPF ensures the logical functionality is preserved. Finally, static timing analysis (/glossary/what-is-static-timing-analysis.html) should be performed with UPF to ensure the design meets timing; and power analysis with detailed waveform behaviors to give accurate power estimation results.

## Low Power Design Tools

The low power design tools needed for each phase of the methodology are:

1. **Static Power Verification and Exploration.** Static verification requires tools for Lint and CDC, to ensure the RTL is clean. For UPF, a UPF checker is necessary to ensure the UPF is clean. For power exploration, a RTL power estimation tool is required.
2. **Dynamic Power Verification and Analysis.** A power-aware simulator is necessary to validate the power sequence and generate waveforms. A power-aware debug tool is also necessary, along with tools that can leverage RTL simulation data and translate to Gate level simulation data, as this is vital for saving time and providing very accurate waveforms for power analysis.
3. **Software Driven Power Analysis.** With emulation, cycle-based power can be generated, observing much longer behaviors, and the right power/timing windows for power analysis can be chosen.
4. **Power Implementation.** RTL-based predictive power estimation, logic synthesis, DFT insertion (Scan Insertion, MBIST, Logic BIST, etc.) and physical implementation (Floor planning, Placement, Clock Tree Synthesis, Routing) are necessary to implement the low power design.

5. **Signoff.** UPF checking for Netlist and UPF' and PGNNetlist and UPF'' is critical to ensure the power is correct and preserved through the flow. Logical equivalence checking with power-aware capabilities also critical to ensure logical functionality is preserved through the flow. Static timing analysis and power analysis tools are critical to verify timing and give accurate power estimation results.

## Low Power Design and Synopsys

Synopsys offers a proven low power flow and methodology solution that covers all aspects of the low power design and verification flow, as shown below in **Figure 6**.

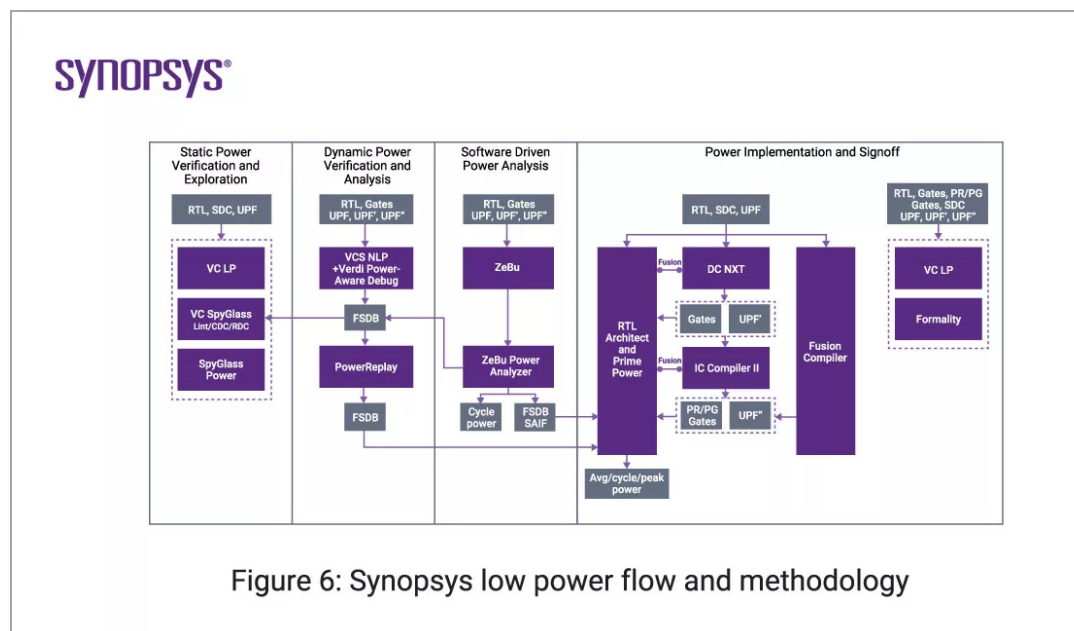


Figure 6: Synopsys low power flow and methodology

🔍 Click to see the detail

### Static Power Verification and Exploration

- **VC SpyGlass Lint/CDC/RDC** (</verification/static-and-formal-verification/vc-spyglass/vc-spyglass-lint.html>) provides Lint and power-aware CDC/RDC checking at the RTL level
- **VC LP** (</verification/static-and-formal-verification/vc-lp.html>) provides UPF checking at RTL+UPF
- **SpyGlass Power** (</verification/static-and-formal-verification/spyglass/spyglass-power.html>) provides RTL Power Estimation

### Functional Verification

- **VCS** (</verification/simulation/vcs.html>) provides power-aware simulation
- **Verdi Power-Aware Debug** (</verification/debug/verdi-power-aware-debug.html>) provides a power-aware debug environment



- **PowerReplay** (</verification/simulation/powerreplay.html>) provides mapping from RTL FSDB (Waveform) to Gate/PG FSDB (Waveform) without having to run Gate/PG simulations

### Software Driven Power Analysis

- **ZeBu** (</verification/emulation.html>) provides power-aware Emulation
- **ZeBu Power Analyzer** (</verification/emulation/use-cases-fast-emulation.html>) provides cycle-by-cycle power estimates and the ability to generate waveforms used for detailed power analysis

### Power Implementation

- **RTL Architect** (</implementation-and-signoff/rtl-synthesis-test/rtl-architect.html>) provides predictive RTL Design Closure
- **Design Compiler** (</implementation-and-signoff/rtl-synthesis-test/design-compiler-graphical.html>) family of products provides power-aware Logical Synthesis
- **TestMAX** (</implementation-and-signoff/test-automation.html>) family of products provides power-aware test insertion, BIST and ATPG solutions
- **IC Compiler II** (</implementation-and-signoff/physical-implementation/ic-compiler.html>) provides power-aware Physical Implementation
- **Fusion Compiler** (</implementation-and-signoff/physical-implementation/fusion-compiler.html>) provides power-aware RTL-to-GDSII Implementation

### Signoff

- **VC LP** (</verification/static-and-formal-verification/vc-lp.html>) provides UPF checking at Netlist and PGNetlist and UPF
- **Formality Equivalence Checking** (</implementation-and-signoff/signoff/formality-equivalence-checking.html>) provides power-aware Logical Equivalence Checking
- **PrimeTime** (</implementation-and-signoff/signoff/primetime.html>) provides power-aware Static Timing Analysis
- **PrimePower** (</implementation-and-signoff/signoff/primepower.html>) provides Power Analysis and Estimation

## Continue Reading

#### BLOG

### AI Requires a New Approach to Low Power Design

Learn More →  
(<https://blogs.synopsys.com/from-silicon-to-software/2020/09/10/low-power-design/>)

#### WHITE PAPER

### Low Power Methodology Manual

Guide to managing power in system-on-chip designs.

Learn More →  
(<https://www.synopsys.com/company/resources/press/low-power-methodology->

#### RELATED SOLUTION

### Energy-Efficient SoCs

Learn More → (</energy-efficient.html>)

(<https://www.synopsys.com/>)

Corporate Headquarters

675 Almanor Ave  
Sunnyvale, CA 94085

(<https://www.google.com/maps/search/?api=1&query=675+Almanor+Ave+>)

Customer Support

650-584-5000  
800-541-7737

Worldwide Location

View our office locations (</company/contact-synopsys/office-locations.html>)

©2023 Synopsys, Inc. All Rights Reserved

Products

- Application Security  
(</software-integrity.html>)
- Semiconductor IP  
(</designware-ip.html>)
- Verification  
(</verification.html>)
- Design (</implementation-and-signoff.html>)
- Silicon Engineering  
(</silicon.html>)

Resources

- Solutions (</solutions.html>)
- Services (</services.html>)
- Support (</support.html>)
- Community  
(</community.html>)
- Academic & Research  
Alliances (SARA)  
(</academic-research.html>)
- Manage Subscriptions  
(<https://online.synopsys.com/contact-us/form-subscription-center.html>)

Corporate

- About Us (</company.html>)
- Careers (</careers.html>)
- ESG  
(</company/environmental-social-governance.html>)
- Inclusion & Diversity  
(</careers/inclusion-diversity.html#present>)
- Investor Relations  
(<https://investor.synopsys.com/overview/default.asp>)
- Contact Us  
(</company/contact-synopsys.html>)

Legal

- Privacy  
(</company/legal/privacy-policy.html>)
- Trademarks & Brands  
(</company/legal/trademarks-brands.html>)
- Software Integrity  
Agreements  
(</company/legal/software-integrity-agreements.html>)

Follow

(<https://twitter.com/synopsys>)