The Quartus® II software offers power-driven compilation to fully optimize device power consumption. Power-driven compilation focuses on reducing your design’s total power consumption using power-driven synthesis and power-driven place-and-route. This chapter describes the power-driven compilation feature and flow in detail, as well as low power design techniques that can further reduce power consumption in your design. The techniques primarily target Arria® GX, Stratix® and Cyclone® series of devices. These devices utilize a low-k dielectric material that dramatically reduces dynamic power and improves performance. Arria series, Stratix II, Stratix III, Stratix IV, and Stratix V device families include efficient logic structures called adaptive logic modules (ALMs) that obtain maximum performance while minimizing power consumption. Cyclone device families offer the optimal blend of high performance and low power in a low-cost FPGA.

For more information about a device-specific architecture, refer to the device handbook, available from the Literature and Technical Documentation page on the Altera website.

Altera provides the Quartus II PowerPlay Power Analyzer to aid you during the design process by delivering fast and accurate estimations of power consumption. You can minimize power consumption, while taking advantage of the industry’s leading FPGA performance, by using the tools and techniques described in this chapter.

For more information about the PowerPlay Power Analyzer, refer to the PowerPlay Power Analysis chapter in volume 3 of the Quartus II Handbook.

Total FPGA power consumption is comprised of I/O power, core static power, and core dynamic power. This chapter focuses on design optimization options and techniques that help reduce core dynamic power and I/O power. In addition to these techniques, there are additional power optimization techniques available for Stratix III and Stratix IV devices. These techniques include:

- Selectable Core Voltage (available only for Stratix III devices)
- Programmable Power Technology
- Device Speed Grade Selection

For more information about power optimization techniques available for Stratix III devices, refer to AN 437: Power Optimization in Stratix III FPGAs. For more information about power optimization techniques available for Stratix IV devices, refer to AN 514: Power Optimization in Stratix IV FPGAs.
Power Dissipation

This section describes the sources of power dissipation in Stratix III and Cyclone III devices. You can refine techniques that reduce power consumption in your design by understanding the sources of power dissipation.

Figure 13–1 shows the power dissipation of Stratix III and Cyclone III devices in different designs. All designs were analyzed at a fixed clock rate of 100 MHz and exhibited varied logic resource utilization across available resources.

Figure 13–1. Average Core Dynamic Power Dissipation

Notes to Figure 13–1:
(1) 103 different designs were used to obtain these results.
(2) 96 different designs were used to obtain these results.
(3) In designs using DSP blocks, DSPs consumed 5% of core dynamic power.

As shown in Figure 13–1, a significant amount of the total power is dissipated in routing for both Stratix III and Cyclone III devices, with the remaining power dissipated in logic, clock, and RAM blocks.

In Stratix and Cyclone device families, a series of column and row interconnect wires of varying lengths provide signal interconnections between logic array blocks (LABs), memory block structures, and digital signal processing (DSP) blocks or multiplier blocks. These interconnects dissipate the largest component of device power.

FPGA combinational logic is another source of power consumption. The basic building block of logic in the latest Stratix series devices is the ALM, and in Cyclone II, Cyclone III and Cyclone IV GX devices, it is the logic element (LE).

For more information about ALMs and LEs in Cyclone II, Cyclone III, Cyclone IV GX, Stratix II, Stratix III, Stratix IV, and Stratix V, devices, refer to the respective device handbook.
Memory and clock resources are other major consumers of power in FPGAs. Stratix II devices feature the TriMatrix memory architecture. TriMatrix memory includes 512-bit M512 blocks, 4-Kbit M4K blocks, and 512-Kbit M-RAM blocks, which are configurable to support many features. Stratix IV and Stratix III TriMatrix on-chip memory is an enhancement based upon the Stratix II FPGA TriMatrix memory and includes three sizes of memory blocks: MLAB blocks, M9K blocks, and M144K blocks. Stratix III, Stratix IV, and Stratix V devices feature Programmable Power Technology, an advanced architecture that enables a smooth trade-off between speed and power. The core of each Stratix III, Stratix IV, and Stratix V device is divided into tiles, each of which may be put into a high-speed or low-power mode. The primary benefit of Programmable Power Technology is to reduce static power, with a secondary benefit being a small reduction in dynamic power. Cyclone II devices have 4-Kbit M4K memory blocks, and Cyclone III and Cyclone IV GX devices have 9-Kbit M9K memory blocks.

**Design Space Explorer**

Design Space Explorer (DSE) is a simple, easy-to-use, design optimization utility that is included in the Quartus II software. DSE explores and reports optimal Quartus II software options for your design, targeting either power optimization, design performance, or area utilization improvements. You can use DSE to implement the techniques described in this chapter.

Figure 13–2 shows the DSE user interface. The Settings tab is divided into Project Settings and Exploration Settings.

**Figure 13–2. Design Space Explorer User Interface**
The **Search for Lowest Power** option, under **Exploration Settings**, uses a predefined exploration space that targets overall design power improvements. This setting focuses on applying different options that specifically reduce total design thermal power.

By default, the Quartus II PowerPlay Power Analyzer is run for every exploration performed by the DSE when the **Search for Lowest Power** option is selected. This helps you debug your design and determine trade-offs between power requirements and performance optimization.

For more information about the DSE, refer to *About Design Space Explorer* in Quartus II Help.

### Power-Driven Compilation

The standard Quartus II compilation flow consists of Analysis and Synthesis, placement and routing, Assembly, and Timing Analysis. Power-driven compilation takes place at the Analysis and Synthesis and Place-and-Route stages. Quartus II software settings that control power-driven compilation are located in the **PowerPlay power optimization** list on the **Analysis & Synthesis Settings** page, and the **PowerPlay power optimization** list on the **Fitter Settings** page. The following sections describes these power optimization options at the Analysis and Synthesis and Fitter levels.

#### Power-Driven Synthesis

Synthesis netlist optimization occurs during the synthesis stage of the compilation flow. The optimization technique makes changes to the synthesis netlist to optimize your design according to the selection of area, speed, or power optimization. This section describes power optimization techniques at the synthesis level.
The **Analysis & Synthesis Settings** page allows you to specify logic synthesis options. The **PowerPlay power optimization** option is available for all devices supported by the Quartus II software except MAX® 3000 and MAX 7000 devices. (Figure 13–3).

![Figure 13–3. Analysis & Synthesis Settings Page](image)

Table 13–1 shows the settings in the **PowerPlay power optimization** list. You can apply these settings on a project or entity level.

**Table 13–1. Optimize Power During Synthesis Options**

<table>
<thead>
<tr>
<th>Settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>No netlist, placement, or routing optimizations are performed to minimize power.</td>
</tr>
<tr>
<td>Normal compilation (Default)</td>
<td>Low compute effort algorithms are applied to minimize power through netlist optimizations as long as they are not expected to reduce design performance.</td>
</tr>
<tr>
<td>Extra effort</td>
<td>High compute effort algorithms are applied to minimize power through netlist optimizations. Max performance might be impacted.</td>
</tr>
</tbody>
</table>

The **Normal compilation** setting is turned on by default. This setting performs memory optimization and power-aware logic mapping during synthesis.
Memory blocks can represent a large fraction of total design dynamic power as described in “Reducing Memory Power Consumption” on page 13–14. Minimizing the number of memory blocks accessed during each clock cycle can significantly reduce memory power. Memory optimization involves effective movement of user-defined read/write enable signals to associated read-and-write clock enable signals for all memory types (Figure 13–4).

**Figure 13–4. Memory Transformation**

<table>
<thead>
<tr>
<th>Data</th>
<th>Q</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wr Clk Enable</td>
<td>Read Enable</td>
</tr>
<tr>
<td>Write Address</td>
<td>Read Address</td>
</tr>
<tr>
<td>Clock</td>
<td>Switch</td>
</tr>
</tbody>
</table>

In Stratix III, Cyclone III, Cyclone IV GX, and Stratix III devices, the specified read-during-write behavior can significantly impact the power of single-port and bidirectional dual-port RAMs. It is best to set the read-during-write parameter to “Don’t care” (at the HDL level), as it allows an optimization whereby the read-enable signal can be set to the inversion of the existing write-enable signal (if one exists). This allows the core of the RAM to shut down (that is, not toggle), which saves a significant amount of power.

The other type of power optimization that takes place with the Normal compilation setting is power-aware logic mapping. The power-aware logic mapping reduces power by rearranging the logic during synthesis to eliminate nets with high toggle rates.

The Extra effort setting performs the functions of the Normal compilation setting and other memory optimizations to further reduce memory power by shutting down memory blocks that are not accessed. This level of memory optimization can require extra logic, which can reduce design performance.
The **Extra effort** setting also performs power-aware memory balancing. Power-aware memory balancing automatically chooses the best memory configuration for your memory implementation and provides optimal power saving by determining the number of memory blocks, decoder, and multiplexer circuits required. If you have not previously specified target-embedded memory blocks for your design’s memory functions, the power-aware balancer automatically selects them during memory implementation.

Figure 13–5 shows an example of a 4k × 4 (4k deep and 4 bits wide) memory implementation in two different configurations using M4K memory blocks available in Stratix II devices. The minimum logic area implementation uses M4K blocks configured as 4k × 1. This implementation is the default in the Quartus II software because it has the minimum logic area (0 logic cells) and the highest speed. However, all four M4K blocks are active on each memory access in this implementation, which increases RAM power. The minimum RAM power implementation is created by selecting **Extra effort** in the **PowerPlay power optimization** list. This implementation automatically uses four M4K blocks configured as 1k × 4 for optimal power saving. An address decoder is implemented by the RAM megafunction to select which of the four M4K blocks should be activated on a given cycle, based on the state of the top two user address bits. The RAM megafunction automatically implements a multiplexer to feed the downstream logic by choosing the appropriate M4K output. This implementation reduces RAM power because only one M4K block is active on any cycle, but it requires extra logic cells, costing logic area and potentially impacting design performance.

There is a trade-off between power saved by accessing fewer memories and power consumed by the extra decoder and multiplexor logic. The Quartus II software automatically balances the power savings against the costs to choose the lowest power configuration for each logical RAM. The benchmark data shows that the power-driven synthesis can reduce memory power consumption by as much as 60% in Stratix devices.

Figure 13–5. 4K × 4 Memory Implementation Using Multiple M4K Blocks
Memory optimization options can also be controlled by the Low_Power_Mode parameter in the Default Parameters page of the Settings dialog box. The settings for this parameter are None, Auto, and ALL. None corresponds to the Off setting in the PowerPlay power optimization list. Auto corresponds to the Normal compilation setting and ALL corresponds to the Extra effort setting, respectively. You can apply PowerPlay power optimization either on a compiler basis or on individual entities. The Low_Power_Mode parameter always takes precedence over the Optimize Power for Synthesis option for power optimization on memory.

You can also set the MAXIMUM_DEPTH parameter manually to configure the memory for low power optimization. This technique is the same as the power-aware memory balancer, but it is manual rather than automatic like the Extra effort setting in the PowerPlay power optimization list. You can set the MAXIMUM_DEPTH parameter for memory modules manually in the megafunction instantiation or in the MegaWizard™ Plug-In Manager for power optimization as described in “Reducing Memory Power Consumption” on page 13–14. The MAXIMUM_DEPTH parameter always takes precedence over the Optimize Power for Synthesis options for power optimization on memory optimization.

For step-by-step instructions on how to perform power-driven synthesis, refer to Running a Power-Optimized Compilation in Quartus II Help.
Power-Driven Fitter

The Fitter Settings page enables you to specify options for fitting (Figure 13–6). The PowerPlay power optimization option is available for Arria GX, Arria II GX, Cyclone II, Cyclone III, Cyclone IV, Stratix II, Stratix II GX, Stratix III, Stratix IV, and Stratix V devices.

Table 13–2 lists the settings in the PowerPlay power optimization list. These settings can only be applied on a project-wide basis. The Extra effort setting for the Fitter requires extensive effort to optimize the design for power and can increase the compilation time.

<table>
<thead>
<tr>
<th>Settings</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Off</td>
<td>No netlist, placement, or routing optimizations are performed to minimize power.</td>
</tr>
<tr>
<td>Normal compilation (Default)</td>
<td>Low compute effort algorithms are applied to minimize power through placement and routing optimizations as long as they are not expected to reduce design performance.</td>
</tr>
<tr>
<td>Extra effort</td>
<td>High compute effort algorithms are applied to minimize power through placement and routing optimizations. Max performance might be impacted.</td>
</tr>
</tbody>
</table>
The **Normal compilation** setting is selected by default and performs DSP optimization by creating power-efficient DSP block configurations for your DSP functions. For Stratix III, Stratix IV, and Stratix V devices, this setting, which is based on timing constraints entered for the design, enables the Programmable Power Technology to configure tiles as high-speed mode or low-power mode. Programmable Power Technology is always turned **ON** even when the **OFF** setting is selected for the Fitter **PowerPlay power optimization** option. Tiles are the combination of LAB and MLAB pairs (including the adjacent routing associated with LAB and MLAB), which can be configured to operate in high-speed or low-power mode. This level of power optimization does not have any affect on the fitting, timing results, or compile time. Also, for Stratix III devices, this setting enables the memory transformation as described in “Power-Driven Synthesis” on page 13–4.

For more information about Stratix III power optimization, refer to *AN 437: Power Optimization in Stratix III FPGAs*. For more information about Stratix IV power optimization, refer to *AN 514: Power Optimization in Stratix IV FPGAs*.

The **Extra effort** setting performs the functions of the **Normal compilation** setting and other place-and-route optimizations during fitting to fully optimize the design for power. The Fitter applies an extra effort to minimize power even after timing requirements have been met by effectively moving the logic closer during placement to localize high-toggling nets, and using routes with low capacitance. However, this effort can increase the compilation time.

The **Extra effort** setting uses a Value Change Dump File (.vcd) that guides the Fitter to fully optimize the design for power, based on the signal activity of the design. The best power optimization during fitting results from using the most accurate signal activity information. Signal activities from full post-fit netlist (timing) simulation provide the highest accuracy because all node activities reflect the actual design behavior, provided that supplied input vectors are representative of typical design operation. If you do not have a .vcd file, the Quartus II software uses assignments, clock assignments, and vectorless estimation values (PowerPlay Power Analyzer Tool settings) to estimate the signal activities. This information is used to optimize your design for power during fitting. The benchmark data shows that the power-driven Fitter technique can reduce power consumption by as much as 19% in Stratix devices. On average, you can reduce core dynamic power by 16% with the Extra effort synthesis and Extra effort fitting settings, as compared to the **Off** settings in both synthesis and Fitter options for power-driven compilation.

Only the **Extra effort** setting in the **PowerPlay power optimization** list for the Fitter option uses the signal activities (from .vcd files) during fitting. The settings made in the **PowerPlay Power Analyzer Settings** page in the **Settings** dialog box are used to calculate the signal activity of your design.

For more information about .vcd files and how to create them, refer to the **PowerPlay Power Analysis** chapter in volume 3 of the *Quartus II Handbook*.

For step-by-step instructions on how to perform power-driven fitting, refer to *Running a Power-Optimized Compilation* in Quartus II Help.
Area-Driven Synthesis

Using area optimization rather than timing or delay optimization during synthesis saves power because you use fewer logic blocks. Using less logic usually means less switching activity. The Quartus II integrated synthesis tool provides **Speed**, **Balanced**, or **Area** for the **Optimization Technique** option. You can also specify this logic option for specific modules in your design with the Assignment Editor in cases where you want to reduce area using the **Area** setting (potentially at the expense of register-to-register timing performance) while leaving the default **Optimization Technique** setting at **Balanced** (for the best trade-off between area and speed for certain device families). The **Speed Optimization Technique** can increase the resource usage of your design if the constraints are too aggressive, and can also result in increased power consumption.

The benchmark data shows that the area-driven technique can reduce power consumption by as much as 31% in Stratix devices and as much as 15% in Cyclone devices.

Gate-Level Register Retiming

You can also use gate-level register retiming to reduce circuit switching activity. Retiming shuffles registers across combinational blocks without changing design functionality. The **Perform gate-level register retiming** option in the Quartus II software enables the movement of registers across combinational logic to balance timing, allowing the software to trade off the delay between timing critical and noncritical timing paths.

Retiming uses fewer registers than pipelining. **Figure 13–7** shows an example of gate-level register retiming, where the 10 ns critical delay is reduced by moving the register relative to the combinational logic, resulting in the reduction of data depth and switching activity.

**Figure 13–7. Gate-Level Register Retiming**
Gate-level register retiming makes changes at the gate level. If you are using an atom netlist from a third-party synthesis tool, you must also select the Perform WYSIWYG primitive resynthesis option to undo the atom primitives to gates mapping (so that register retiming can be performed), and then to remap gates to Altera primitives. When using Quartus II integrated synthesis, retiming occurs during synthesis before the design is mapped to Altera primitives. The benchmark data shows that the combination of WYSIWYG remapping and gate-level register retiming techniques can reduce power consumption by as much as 6% in Stratix devices and as much as 21% in Cyclone devices.

For more information about register retiming, refer to the Netlist Optimizations and Physical Synthesis chapter in volume 2 of the Quartus II Handbook.

Design Guidelines

Several low-power design techniques can reduce power consumption when applied during FPGA design implementation. This section provides detailed design techniques for Cyclone II, Cyclone III, Cyclone IV GX, Stratix II, and Stratix III devices that affect overall design power. The results of these techniques might be different from design to design.

Clock Power Management

Clocks represent a significant portion of dynamic power consumption due to their high switching activity and long paths. Figure 13–1 on page 13–2 shows a 14% average contribution to power consumption for global clock routing in Stratix III devices and 16% in Cyclone III devices. Actual clock-related power consumption is higher than this because the power consumed by local clock distribution within logic, memory, and DSP or multiplier blocks is included in the power consumption for the respective blocks.

Clock routing power is automatically optimized by the Quartus II software, which enables only those portions of the clock network that are required to feed downstream registers. Power can be further reduced by gating clocks when they are not required. It is possible to build clock-gating logic, but this approach is not recommended because it is difficult to generate a glitch free clock in FPGAs using ALMs or LEs.

Arria GX, Arria II GX, Cyclone III, Cyclone IV, Stratix II, Stratix III, Stratix IV, and Stratix V devices use clock control blocks that include an enable signal. A clock control block is a clock buffer that lets you dynamically enable or disable the clock network and dynamically switch between multiple sources to drive the clock network. You can use the Quartus II MegaWizard Plug-In Manager to create this clock control block with the ALTCLKCTRL megafunction. Arria GX, Arria II GX, Cyclone III, Cyclone IV, Stratix II, Stratix III, Stratix IV, and Stratix V devices provide clock control blocks for global clock networks. In addition, Stratix II, Stratix III,
Stratix IV, and Stratix V devices have clock control blocks for regional clock networks. The dynamic clock enable feature lets internal logic control the clock network. When a clock network is powered down, all the logic fed by that clock network does not toggle, thereby reducing the overall power consumption of the device. Figure 13–8 shows a 4-input clock control block diagram.

**Figure 13–8. Clock Control Block Diagram**

The enable signal is applied to the clock signal before being distributed to global routing. Therefore, the enable signal can either have a significant timing slack (at least as large as the global routing delay) or it can reduce the $f_{\text{MAX}}$ of the clock signal.

For more information about using clock control blocks, refer to the *Clock Control Block Megafunction User Guide (ALTCLKCTRL)*.

Another contributor to clock power consumption is the LAB clock that distributes a clock to the registers within a LAB. LAB clock power can be the dominant contributor to overall clock power. For example, in Cyclone III devices, each LAB can use two clocks and two clock enable signals, as shown in Figure 13–9. Each LAB’s clock signal and clock enable signal are linked. For example, an LE in a particular LAB using the labclk1 signal also uses the labclkena1 signal.

**Figure 13–9. LAB-Wide Control Signals**
To reduce LAB-wide clock power consumption without disabling the entire clock tree, use the LAB-wide clock enable to gate the LAB-wide clock. The Quartus II software automatically promotes register-level clock enable signals to the LAB-level. All registers within an LAB that share a common clock and clock enable are controlled by a shared gated clock. To take advantage of these clock enables, use a clock enable construct in the relevant HDL code for the registered logic.

**LAB-Wide Clock Enable Example**

The VHDL code in Example 13–1 makes use of a LAB-wide clock enable. This clock-gating logic is automatically turned into an LAB-level clock enable signal.

**Example 13–1.**

```vhdl
IF clk'event AND clock = '1' THEN
    IF logic_is_enabled = '1' THEN
        reg <= value;
    ELSE
        reg <= reg;
    END IF;
END IF;
```

For more information about LAB-wide control signals, refer to the *Stratix II Architecture*, *Cyclone III Device Family Overview*, or *Cyclone II Architecture* chapters in the respective device handbook.

**Reducing Memory Power Consumption**

The memory blocks in FPGA devices can represent a large fraction of typical core dynamic power. Memory consumes approximately 20% of the core dynamic power in typical Cyclone III and Stratix III device designs. Memory blocks are unlike most other blocks in the device because most of their power is tied to the clock rate, and is insensitive to the toggle rate on the data and address lines.

When a memory block is clocked, there is a sequence of timed events that occur within the block to execute a read or write. The circuitry controlled by the clock consumes the same amount of power regardless of whether or not the address or data has changed from one cycle to the next. Thus, the toggle rate of input data and the address bus have no impact on memory power consumption.

The key to reducing memory power consumption is to reduce the number of memory clocking events. You can achieve this through clock network-wide gating described in “Clock Power Management” on page 13–12, or on a per-memory basis through use of the clock enable signals on the memory ports. Figure 13–10 shows the logical view of the internal clock of the memory block. Use the appropriate enable signals on the memory to make use of the clock enable signal instead of gating the clock.

**Figure 13–10. Memory Clock Enable Signal**

![Internal Memory Ck]

Enable

**Clk**
Using the clock enable signal enables the memory only when necessary and shuts it down for the rest of the time, reducing the overall memory power consumption. You can use the MegaWizard Plug-In Manager to create these enable signals by selecting the **Clock enable signal** option for the appropriate port when generating the memory block function (Figure 13–11).

For example, consider a design that contains a 32-bit-wide M4K memory block in ROM mode that is running at 200 MHz. Assuming that the output of this block is only required approximately every four cycles, this memory block will consume 8.45 mW of dynamic power according to the demands of the downstream logic. By adding a small amount of control logic to generate a read clock enable signal for the memory block only on the relevant cycles, the power can be cut 75% to 2.15 mW.

You can also use the **MAXIMUM_DEPTH** parameter in your memory megafunction to save power in Cyclone II, Cyclone III, Cyclone IV GX, Stratix II, Stratix III, Stratix IV, and Stratix V devices; however, this approach might increase the number of LEs required to implement the memory and affect design performance.

You can set the **MAXIMUM_DEPTH** parameter for memory modules manually in the megafunction instantiation or in the MegaWizard Plug-In Manager (Figure 13–12). The Quartus II software automatically chooses the best design memory configuration for optimal power, as described in “Power-Driven Compilation” on page 13–4.
Memory Power Reduction Example

Table 13–3 shows power usage measurements for a 4K × 36 simple dual-port memory implemented using multiple M4K blocks in a Stratix II EP2S15 device. For each implementation, the M4K blocks are configured with a different memory depth.

Table 13–3. 4K × 36 Simple Dual-Port Memory Implemented Using Multiple M4K Blocks

<table>
<thead>
<tr>
<th>M4K Configuration</th>
<th>Number of M4K Blocks</th>
<th>ALUTs</th>
</tr>
</thead>
<tbody>
<tr>
<td>4K × 1 (Default setting)</td>
<td>36</td>
<td>0</td>
</tr>
<tr>
<td>2K × 2</td>
<td>36</td>
<td>40</td>
</tr>
<tr>
<td>1K × 4</td>
<td>36</td>
<td>62</td>
</tr>
<tr>
<td>512 × 9</td>
<td>32</td>
<td>143</td>
</tr>
<tr>
<td>256 × 18</td>
<td>32</td>
<td>302</td>
</tr>
<tr>
<td>128 × 36</td>
<td>32</td>
<td>633</td>
</tr>
</tbody>
</table>
Figure 13–13 shows the amount of power saved using the MAXIMUM_DEPTH parameter. For all implementations, a user-provided read enable signal is present to indicate when read data is required. Using this power-saving technique can reduce power consumption by as much as 60%.

![Power Savings Using the MAXIMUM_DEPTH Parameter](image)

As the memory depth becomes more shallow, memory dynamic power decreases because unaddressed M4K blocks can be shut off using a decoded combination of address bits and the read enable signal. For a 128-deep memory block, power used by the extra LEs starts to outweigh the power gain achieved by using a more shallow memory block depth. The power consumption of the memory blocks and associated LEs depends on the memory configuration.

The SOPC Builder and Qsys system do not offer specific power savings control for on-chip memory block. There is no read enable, write enable, or clock enable that you can enable in the on-chip RAM megafunction to shut down the RAM block in the SOPC Builder and Qsys system.

### Pipelining and Retiming

Designs with many glitches consume more power because of faster switching activity. Glitches cause unnecessary and unpredictable temporary logic switches at the output of combinational logic. A glitch usually occurs when there is a mismatch in input signal timing leading to unequal propagation delay.

For example, consider an input change on one input of a 2-input XOR gate from 1 to 0, followed a few moments later by an input change from 0 to 1 on the other input. For a moment, both inputs become 1 (high) during the state transition, resulting in 0 (low) at the output of the XOR gate. Subsequently, when the second input transition takes place, the XOR gate output becomes 1 (high). During signal transition, a glitch is
produced before the output becomes stable, as shown in Figure 13–14. This glitch can propagate to subsequent logic and create unnecessary switching activity, increasing power consumption. Circuits with many XOR functions, such as arithmetic circuits or cyclic redundancy check (CRC) circuits, tend to have many glitches if there are several levels of combinational logic between registers.

Figure 13–14. XOR Gate Showing Glitch at the Output

Pipelining can reduce design glitches by inserting flipflops into long combinational paths. Flipflops do not allow glitches to propagate through combinational paths. Therefore, a pipelined circuit tends to have less glitching. Pipelining has the additional benefit of generally allowing higher clock speed operations, although it does increase the latency of a circuit (in terms of the number of clock cycles to a first result). Figure 13–15 shows an example where pipelining is applied to break up a long combinational path.

Figure 13–15. Pipelining Example

Pipelining is very effective for glitch-prone arithmetic systems because it reduces switching activity, resulting in reduced power dissipation in combinational logic. Additionally, pipelining allows higher-speed operation by reducing logic-level numbers between registers. The disadvantage of this technique is that if there are not many glitches in your design, pipelining can increase power consumption by adding unnecessary registers. Pipelining can also increase resource utilization. The benchmark data shows that pipelining can reduce dynamic power consumption by as much as 30% in Cyclone and Stratix devices.
Architectural Optimization

You can use design-level architectural optimization by taking advantage of specific device architecture features. These features include dedicated memory and DSP or multiplier blocks available in FPGA devices to perform memory or arithmetic-related functions. You can use these blocks in place of LUTs to reduce power consumption. For example, you can build large shift registers from RAM-based FIFO buffers instead of building the shift registers from the LE registers.

The Stratix device family allows you to efficiently target small, medium, and large memories with the TriMatrix memory architecture. Each TriMatrix memory block is optimized for a specific function. The M512 memory blocks available in Stratix II devices are useful for implementing small FIFO buffers, DSP, and clock domain transfer applications. M512 memory blocks are more power-efficient than the distributed memory structures in some competing FPGAs. The M4K memory blocks are used to implement buffers for a wide variety of applications, including processor code storage, large look-up table implementation, and large memory applications. The M-RAM blocks are useful in applications where a large volume of data must be stored on-chip. Effective utilization of these memory blocks can have a significant impact on power reduction in your design.

The latest Stratix and Cyclone device families have configurable M9K memory blocks that provide various memory functions such as RAM, FIFO buffers, and ROM.

For more information about using DSP and memory blocks efficiently, refer to the Area and Timing Optimization chapter in volume 2 of the Quartus II Handbook.

I/O Power Guidelines

Nonterminated I/O standards such as LVTTL and LVCMOS have a rail-to-rail output swing. The voltage difference between logic-high and logic-low signals at the output pin is equal to the $V_{CCIO}$ supply voltage. If the capacitive loading at the output pin is known, the dynamic power consumed in the I/O buffer can be calculated as shown in Equation 13–1:

\[
P = 0.5 \times F \times C \times V^2
\]

In this equation, $F$ is the output transition frequency and $C$ is the total load capacitance being switched. $V$ is equal to $V_{CCIO}$ supply voltage. Because of the quadratic dependence on $V_{CCIO}$, lower voltage standards consume significantly less dynamic power.

Transistor-to-transistor logic (TTL) I/O buffers consume very little static power. As a result, the total power consumed by a LVTTL or LVCMOS output is highly dependent on load and switching frequency.

When using resistively terminated I/O standards like SSTL and HSTL, the output load voltage swings by a small amount around some bias point. The same dynamic power equation is used, where $V$ is the actual load voltage swing. Because this is much smaller than $V_{CCIO}$, dynamic power is lower than for nonterminated I/O under similar conditions. These resistively terminated I/O standards dissipate significant static (frequency-independent) power, because the I/O buffer is constantly driving
current into the resistive termination network. However, the lower dynamic power of these I/O standards means they often have lower total power than LVCMOS or LVTTL for high-frequency applications. Use the lowest drive strength I/O setting that meets your speed and waveform requirements to minimize I/O power when using resistively terminated standards.

You can save a small amount of static power by connecting unused I/O banks to the lowest possible $V_{CCIO}$ voltage of 1.2 V.

Table 13–4 shows the total supply and thermal power consumed by outputs using different I/O standards for Stratix II devices. The numbers are for an I/O pin transmitting random data clocked at 200 MHz with a 10 pF capacitive load.

For this configuration, nonterminated standards generally use less power, but this is not always the case. If the frequency or the capacitive load is increased, the power consumed by nonterminated outputs increases faster than the power of terminated outputs.

Table 13–4. I/O Power for Different I/O Standards in Stratix II Devices

<table>
<thead>
<tr>
<th>Standard</th>
<th>Total Supply Current Drawn from $V_{CCIO}$ Supply (mA)</th>
<th>Total On-Chip Thermal Power Dissipation (mW)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3-V LVTTL</td>
<td>2.42</td>
<td>9.87</td>
</tr>
<tr>
<td>2.5-V LVCMOS</td>
<td>1.9</td>
<td>6.69</td>
</tr>
<tr>
<td>1.8-V LVCMOS</td>
<td>1.34</td>
<td>4.18</td>
</tr>
<tr>
<td>1.5-V LVCMOS</td>
<td>1.18</td>
<td>3.58</td>
</tr>
<tr>
<td>3.3-V PCI</td>
<td>2.47</td>
<td>10.23</td>
</tr>
<tr>
<td>SSTL-2 class I</td>
<td>6.07</td>
<td>4.42</td>
</tr>
<tr>
<td>SSTL-2 class II</td>
<td>10.72</td>
<td>5.1</td>
</tr>
<tr>
<td>SSTL-18 class I</td>
<td>5.33</td>
<td>3.28</td>
</tr>
<tr>
<td>SSTL-18 class II</td>
<td>8.56</td>
<td>4.06</td>
</tr>
<tr>
<td>HSTL-15 class I</td>
<td>6.06</td>
<td>3.49</td>
</tr>
<tr>
<td>HSTL-15 class II</td>
<td>11.08</td>
<td>4.87</td>
</tr>
<tr>
<td>HSTL-18 class I</td>
<td>6.87</td>
<td>4.09</td>
</tr>
<tr>
<td>HSTL-18 class II</td>
<td>12.33</td>
<td>5.82</td>
</tr>
</tbody>
</table>

When calculating I/O power, the PowerPlay Power Analyzer uses the default capacitive load set for the I/O standard in the Capacitive Loading page of the Device and Pin Options dialog box. For Stratix II devices, if Enable Advanced I/O Timing is turned on, I/O power is measured using an equivalent load calculated as the sum of the near capacitance, the transmission line distributed capacitance, and the far-end capacitance as defined in the Board Trace Model page of the Device and Pin Options dialog box or the Board Trace Model view in the Pin Planner. Any other components defined in the board trace model are not taken into account for the power measurement.

For Cyclone III, Cyclone IV GX, Stratix III, Stratix IV, and Stratix V, devices, Advanced I/O Timing, which uses the full board trace model, is always used.

For information about using Advanced I/O Timing and configuring a board trace model, refer to the I/O Management chapter in volume 2 of the Quartus II Handbook.

**Dynamically Controlled On-Chip Terminations**

Stratix V, Stratix IV and Stratix III FPGAs offer dynamic on-chip termination (OCT). Dynamic OCT enables series termination (RS) and parallel termination (RT) to dynamically turn on/off during the data transfer. This feature is especially useful when Stratix V, Stratix IV and Stratix III FPGAs are used with external memory interfaces, such as interfacing with DDR memories.

Compared to conventional termination, dynamic OCT reduces power consumption significantly as it eliminates the constant DC power consumed by parallel termination when transmitting data. Parallel termination is extremely useful for applications that interface with external memories where I/O standards, such as HSTL and SSTL, are used. Parallel termination supports dynamic OCT, which is useful for bidirectional interfaces (see Figure 13–16).

**Figure 13–16. Stratix III On-Chip Parallel Termination**

The following is an example of power saving for a DDR3 interface using on-chip parallel termination.
The static current consumed by parallel OCT is equal to the V_{CCIO} voltage divided by 100 Ω. For DDR3 interfaces that use SSTL-15, the static current is 1.5 V/100 Ω = 15 mA per pin. Therefore, the static power is 1.5 V × 15 mA = 22.5 mW. For an interface with 72 DQ and 18 DQS pins, the static power is 90 pins × 22.5 mW = 2.025 W. Dynamic parallel OCT disables parallel termination during write operations, so if writing occurs 50% of the time, the power saved by dynamic parallel OCT is 50% × 2.025 W = 1.0125 W.

For more information about dynamic OCT in Stratix IV and Stratix III devices, refer to the *Stratix III Device I/O Features* chapter in the *Stratix III Device Handbook* and the *Stratix IV Device I/O Features* chapter in the *Stratix IV Device Handbook*, respectively.

**Power Optimization Advisor**

The Quartus II software includes the Power Optimization Advisor, which provides specific power optimization advice and recommendations based on the current design project settings and assignments. The advisor covers many of the suggestions listed in this chapter. The following example shows how to reduce your design power with the Power Optimization Advisor.

**Power Optimization Advisor Example**

After compiling your design, run the PowerPlay Power Analyzer to determine your design power and to see where power is dissipated in your design. Based on this information, you can run the Power Optimization Advisor to implement recommendations that can reduce design power. Figure 13–17 shows the Power Optimization Advisor after compiling a design that is not fully optimized for power.

![Power Optimization Advisor](image)

The Power Optimization Advisor shows the recommendations that can reduce power in your design. The recommendations are split into stages to show the order in which you should apply the recommended settings. The first stage shows mostly CAD setting options that are easy to implement and highly effective in reducing design power. An icon indicates whether each recommended setting is made in the current
project. In Figure 13–17, the checkmark icons for Stage 1 shows the recommendations that are already implemented. The warning icons indicate recommendations that are not followed for this compilation. The information icon shows the general suggestions. Each recommendation includes the description, summary of the effect of the recommendation, and the action required to make the appropriate setting.

There is a link from each recommendation to the appropriate location in the Quartus II user interface where you can change the setting. You can change the Power-Driven Synthesis setting by clicking Open Settings dialog box - Analysis & Synthesis Settings page. The Settings dialog box is shown with the Analysis & Synthesis Settings page selected, where you can change the PowerPlay power optimization settings.

After making the recommended changes, recompile your design. The Power Optimization Advisor indicates with green check marks that the recommendations were implemented successfully (Figure 13–18). You can use the PowerPlay Power Analyzer to verify your design power results.

The recommendations listed in Stage 2 generally involve design changes, rather than CAD settings changes as in Stage 1. You can use these recommendations to further reduce your design power consumption. Altera recommends that you implement Stage 1 recommendations first, then the Stage 2 recommendations.

**Conclusion**

The combination of a smaller process technology, the use of low-k dielectric material, and reduced supply voltage significantly reduces dynamic power consumption in the latest FPGAs. To further reduce your dynamic power, use the design recommendations presented in this chapter to optimize resource utilization and minimize power consumption.
## Document Revision History

Table 13–5 shows the revision history for this chapter.

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>May 2013</td>
<td>13.0.0</td>
<td>Added a note to “Memory Power Reduction Example” on page 13–16 on Qsys and SOPC Builder power savings limitation for on-chip memory block.</td>
</tr>
<tr>
<td>June 2012</td>
<td>12.0.0</td>
<td>Removed survey link.</td>
</tr>
<tr>
<td>November 2011</td>
<td>10.0.2</td>
<td>Template update.</td>
</tr>
<tr>
<td>December 2010</td>
<td>10.0.1</td>
<td>Template update.</td>
</tr>
<tr>
<td>July 2010</td>
<td>10.0.0</td>
<td>■ Was chapter 11 in the 9.1.0 release</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated Figures 14-2, 14-3, 14-6, 14-18, 14-19, and 14-20</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated device support</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Minor editorial updates</td>
</tr>
<tr>
<td>November 2009</td>
<td>9.1.0</td>
<td>■ Updated Figure 11-1 and associated references</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated device support</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Minor editorial update</td>
</tr>
<tr>
<td>March 2009</td>
<td>9.0.0</td>
<td>■ Was chapter 9 in the 8.1.0 release</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated for the Quartus II software release</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Added benchmark results</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Removed several sections</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated Figure 13–1, Figure 13–17, and Figure 13–18</td>
</tr>
<tr>
<td>November 2008</td>
<td>8.1.0</td>
<td>■ Changed to 8½“ × 11” page size</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Changed references to altsyncram to RAM</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Minor editorial updates</td>
</tr>
<tr>
<td>May 2008</td>
<td>8.0.0</td>
<td>■ Added support for Stratix IV devices</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated Table 9–1 and 9–9</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated “Architectural Optimization” on page 9–22</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Added “Dynamically-Controlled On-Chip Terminations” on page 9–26</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated “Referenced Documents” on page 9–29</td>
</tr>
<tr>
<td></td>
<td></td>
<td>■ Updated references</td>
</tr>
</tbody>
</table>

For previous versions of the Quartus II Handbook, refer to the [Quartus II Handbook Archive](#).