Dynamic Power Reduction in Flash FPGAs

Table of Contents

Introduction

Due to the dramatic increase in portable and battery-operated applications, lower power consumption has become a necessity in order to prolong the battery life. Power consumption is an important part of the equation determining the end product's size, weight, and efficiency. Field programmable gate array (FPGA)s are becoming more attractive for these applications due to their shorter product life cycle. FPGAs are programmable, so they allow product differentiation. Selecting an appropriate FPGA architecture is critical in achieving the best static and dynamic power consumption.

Flash-based FPGAs by Microsemi are the low-power leaders in the industry. In addition to utilizing the low-power attributes of flash-based FPGAs, you can deploy several design techniques to further reduce overall power.

This application note describes various design techniques to help reduce dynamic power and accomplish this goal easily, presenting several coding scenarios with real-world results. This application note assumes that you are familiar with Microsemi flash FPGA architecture.

This document is organized in various sections:

- "Power Components and Dynamic Power"
- "Using SmartPower to Compute FPGA Power Profile"
- "Recommended Design Flow for Low-Power Designs"
- "Standard Techniques for Lowering Dynamic Power"
- "General Tips During Synthesis and Place-and-Route"
- "General Design Practice"

Power Components and Dynamic Power

The important FPGA power components to consider in the following sections:

- Power-up (inrush power): Inrush power is the amount of power drawn by the device during power-up.
- Configuration power: Configuration power is the amount of power required during the loading of the FPGA upon power-up (specific to SRAM-based programmable logic devices).
- Static (standby) power: Static power is the amount of power the device consumes when it is powered-up but not actively performing any operation.

- Dynamic (active) power: Dynamic power is the amount of power the device consumes when it is actively operating.
- Sleep power (low-power mode): Some FPGA devices offer low-power or sleep modes. In some cases, this may be different from static power.

This application note focuses on reducing the dynamic power. In general, the dynamic power is calculated using the formula shown in EQ 1:

$$P = \alpha * C * V^2 * F$$

EQ 1

Where α is the switching activity, C is the capacitive load, V is the supply voltage, and F is the frequency.

In flash FPGAs, the components that consume dynamic power are clock networks, logic blocks, routing resources (nets), I/Os, memory, PLLs, etc. These components have different α, C, V, and F values. For example, the dynamic power of a net depends on the average switching (α), the total capacitive loading of the net (C), the net's voltage swing (V), and the frequency (F).

To resolve the dynamic power problem, you must first calculate the dynamic power. The total dynamic power consumption (P_{DYN}) in flash FPGAs is approximated in EQ 2:

$$P_{DYN} = P_{CLOCK} + P_{S\text{-}CELL} + P_{C\text{-}CELL} + P_{NET} + P_{INPUTS} + P_{OUTPUTS} + P_{MEMORY} + P_{PLL}$$

EQ 2

Where:

P_{CLOCK}	=	Global Clock Contribution
$P_{S\text{-}CELL}$	=	Sequential Cells Contribution
$P_{C\text{-}CELL}$	=	Combinatorial Cells Contribution
P_{NET}	=	Routing Net Contribution
P_{INPUTS}	=	I/O Input Buffer Contribution
$P_{OUTPUTS}$	=	I/O Output Buffer Contribution
P_{MEMORY}	=	RAM Contribution
P_{PLL}	=	PLL Contribution

The FPGA handbooks give a detailed formula for estimating P_{CLOCK}, $P_{S\text{-}CELL}$, $P_{C\text{-}CELL}$, P_{NET}, P_{INPUTS}, $P_{OUTPUTS}$, P_{MEMORY}, and P_{PLL}. However, to get more accurate and detailed power estimations, use the SmartPower tool included in Microsemi Libero® System-on-Chip (SoC). The next section describes how to use SmartPower to generate an FPGA power profile and obtain various power components.

Using SmartPower to Compute FPGA Power Profile

SmartPower is a state-of-the-art power analysis tool by Microsemi that enables you to visualize power consumption and potential power consumption problems within your design so you can make adjustments to reduce power when possible. SmartPower allows you to find power consumption for individual blocks, nets, macros, and FPGA resources. It is very useful for identifying the component inflicting the largest power penalty and improving the power consumption by using the techniques discussed in this application note. Refer to the *Microsemi SmartPower v8.5 User's Guide* for details. SmartPower allows you to estimate power in two ways:

1. Vectorless post-layout power estimation
2. Simulation-based power estimation

Vectorless Post-Layout Power Estimation

This is a less accurate method that uses statistical switching rate to compute power for combinatorial and sequential logic. This method should be used when no simulation data is available and power estimates are required as soon as possible. You provide only the following information to the SmartPower tool:

- Operating clock frequencies of the design using SmartTime GUI or SDC constraint
- Estimated toggle rate using SmartPower or Tcl command

After you perform place-and-route on the design, SmartPower uses this information to calculate the power. Provide the toggle rate carefully because it directly impacts power calculation. Typical design toggle rates are 10-15%. However, you can reduce the toggling rate by applying design techniques discussed in this application note.

Simulation-Based Power Estimation

This is the recommended method for power estimation because it provides more accurate analysis. In this method, you import an industry-standard value change dump (VCD) or SAIF file to SmartPower. SmartPower uses the clock frequencies and switching activity information and generates the power estimates. It is more accurate than the vectorless approach because the switching activity information extracted from the VCD/SAIF file is more accurate.

Microsemi Libero SoC allows you to generate a VCD file during simulation and import it to SmartPower. Refer to Libero SoC online help for details on generating and using VCD files. After importing the VCD file, view the Summary tab in SmartPower for the total power computation (Figure 1 on page 3). The Summary tab also has an option that allows breakdown of the design's power usage by design hierarchy as well as the following types: Net, Gate, I/O, Memory, Core Static, and Banks Static. In addition, it can show power analysis by Type, by Rail, or by Clock Domain.

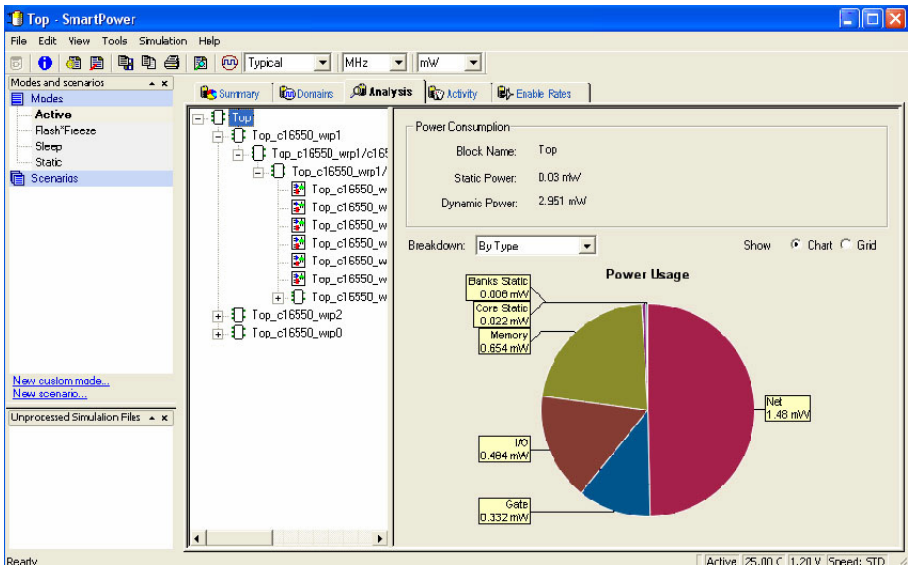

***Figure 1* • Power Profile Using SmartPower Power Analysis Tool**

SmartPower also generates the cycle-accurate power analysis, which reports a power value for each clock period instead of an average power over the entire simulation time. This feature allows you to identify the worst cycle in terms of power consumption. You can obtain the cycle number, the start and end times, and the power usage for the selected cycle (Figure 2 on page 4).

You can analyze bottlenecks and take appropriate action.

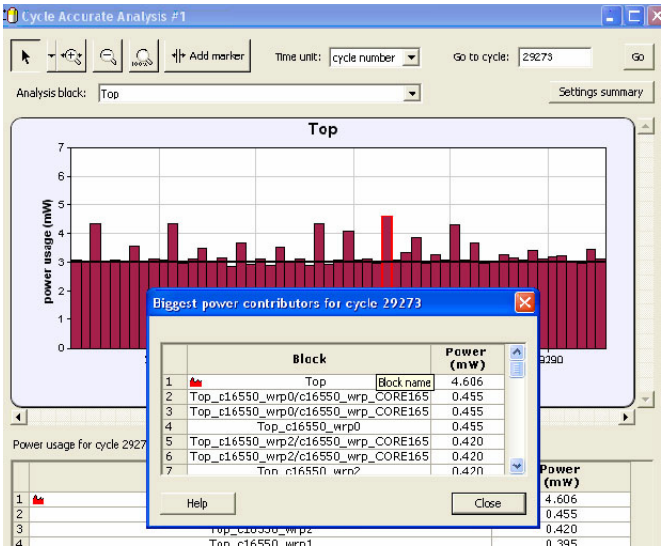

***Figure 2* • Power Profile Using SmartPower Power Analysis Tool**

Recommended Design Flow for Low-Power Designs

For most designs, the goal is to achieve timing requirements first and then minimize the power consumption. There are also low clock frequency designs that have sufficient timing margin and the primary goal with those is to minimize the average power consumption. Depending on the requirement, the design flow can change. However, the general design flow is shown in Figure 3. The primary goal of this flow is to reduce the dynamic power while maintaining the timing constraints. Here are the main steps:

- The flow starts with using regular RTL in timing-critical blocks and power-friendly RTL source code for non-timing-critical blocks.
- Go through synthesis and run timing-driven place-and-route. You may need to use advanced options if you do not meet timing with default options.
- After you meet the timing in place-and-route, run post layout simulation and import the VCD file.
- Run timing- and power-driven layout and perform power analysis using SmartPower.

- If the power consumption is not acceptable, analyze the power consumption for various block instances. Then go back to the RTL and make further modifications, following the iterative process.

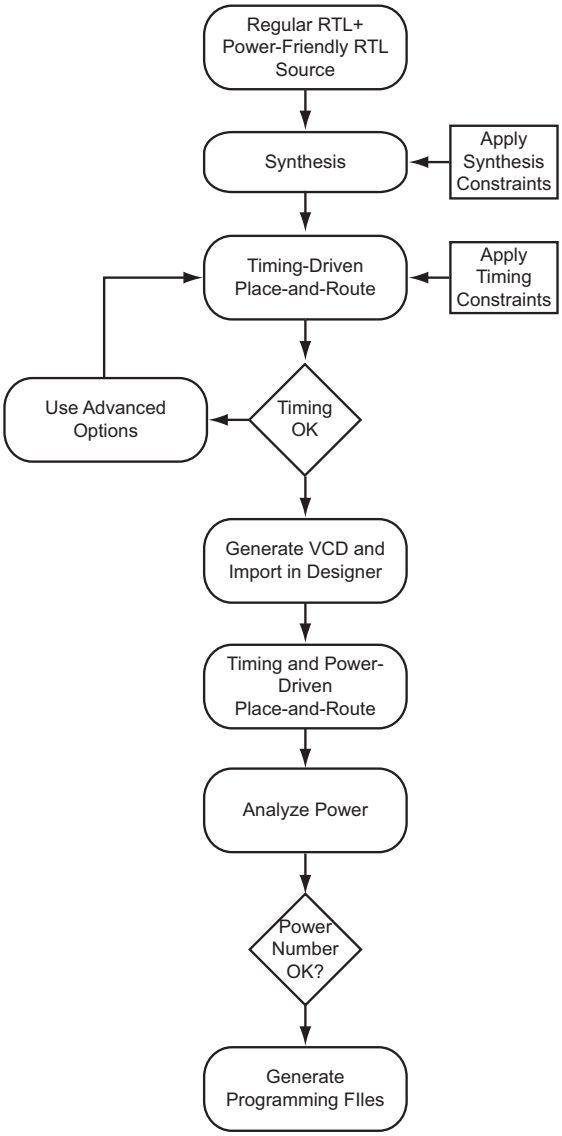

***Figure 3* • Low-Power Design Flow**

Standard Techniques for Lowering Dynamic Power

The following techniques will help you reduce dynamic power:

- Decrease the average logic-switching frequency.
- Reduce the amount of logic switching at each clock edge.
- Reduce the propagation of the switching activity.
- Lower the capacitance of the routing network, especially for high-frequency signals.
- Use low voltage I/O standards. Note that any reduction in voltage has a quadratic effect on power.

Follow the design flow and generate a power profile using SmartPower with a VCD file from post layout simulation, as discussed in the "Recommended Design Flow for Low-Power Designs" section. The dynamic power profile provides a clear picture of the consumption of each of the FPGA resources used.

Because FPGAs are flexible and allow many types of applications to be mapped on the same device, there is no recipe for efficiently tackling the dynamic power without a deep understanding of the actual design power profile.

Figure 4 provides examples of power profiles for various designs. A quick look at the power profile of Design 1 indicates you would avoid deploying efforts to lower dynamic power for the I/Os. In Design 2 and Design 3, however, you would focus power optimization and thermal management efforts on the I/Os.

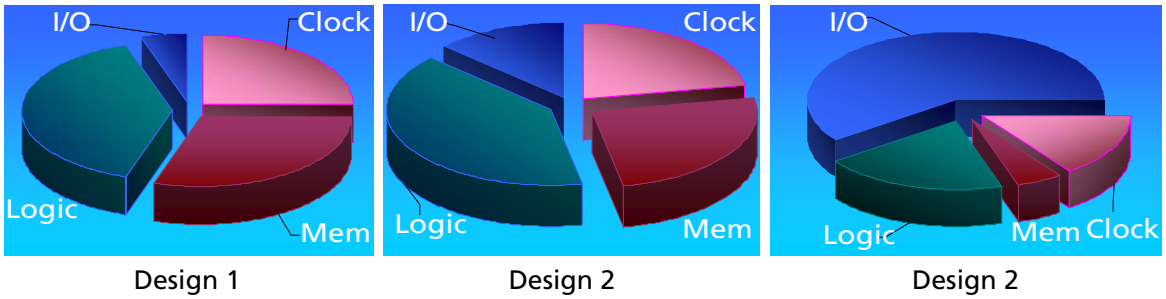

***Figure 4* • Power Profile for Various Designs**

Reducing Dynamic Power on Clock Scheme

Clock Gating at Chip Level

The easy and popular system-level clock-gating stops the clock for an entire FPGA, effectively disabling all functionality. It prevents the logic from switching. Flash-based FPGAs by Microsemi use Flash*Freeze mode, which is more flexible than system-level clock gating, controlling input and output states and freezing clocks. This technique is not discussed in this application note. Refer to the "Flash*Freeze Technology and Low-Power Modes in IGLOO, IGLOO PLUS, and ProASIC3L Devices" chapter of the *IGLOO FPGA Fabric User's Guide* for information on using Flash*Freeze mode.

Clock Gating at Design Level

Clock gating at RTL level is a commonly used power-saving technique. There are two types of clock gating: latch-based and latch-free clock gating types (Figure 5 on page 7). The latch-free clock gating style uses a simple AND gate. However, this imposes a requirement on the circuit that the gating signal be held constant from the active (rising) edge of the clock until the inactive (falling) edge of the clock to prevent clock glitches.

The latch-based clock gating style adds a level-sensitive latch to the design to hold the enable signal from the active edge of the clock until the inactive edge of the clock, making it unnecessary for the circuit itself to enforce that requirement.

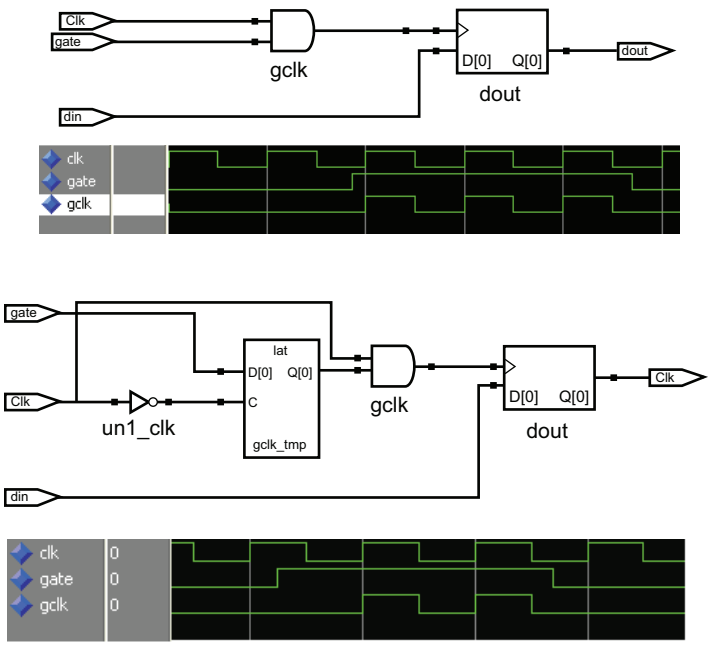

Figure 5 • **Clock Gating**

Clock gating is a straightforward substitution for RTL code. FPGA synthesis tools do not perform clock gating automatically. You must identify groups of flip-flops that share a common enable term to implement combinational clock gating. Therefore, if a bank of flip-flops which share a common enable term use RTL clock gating, the flip-flops will consume zero dynamic power as long as this enable term is false. However, you must be careful when implementing it because the skew between clock and enable signal could cause extra glitches. Microsemi recommends using a gating scheme with latch to eliminate the possibility of extra glitches at the AND gate output.

Global Resource Power Reduction

Both chip global and quadrant global networks in flash-based FPGAs are segmented into sections called spines. Refer to the "Global Resources in Microsemi Low-Power Flash Devices" chapter of the *ProASIC3 FPGA Fabric User's Guide* for detailed information on global architecture. Clock-tree power depends heavily on the frequency and fanout of each clock domain. In general, with a higher number of spines used and higher capacitive loading, the power consumption will be higher. So the key elements of reducing power consumption for clock networks are to reduce frequency, reduce the number of clock spines, and reduce the fanout and loading on the global networks.

The Microsemi Designer place-and-route tool has a Power-Driven Layout option that minimizes the number of spines and rows used in an effort to reduce clock tree power. Figure 6 on page 8 shows examples that allow the optimal exploitation of two clock networks. In some cases, however, because of the placement of RAM blocks or placement of the clock net driver or the flip-flops, power-driven layout might not result in the best optimal clock network usage.

You should review the clock domains and the interactions among major functional blocks, and then use manual floorplanning to switch off unused segments of clock trees.

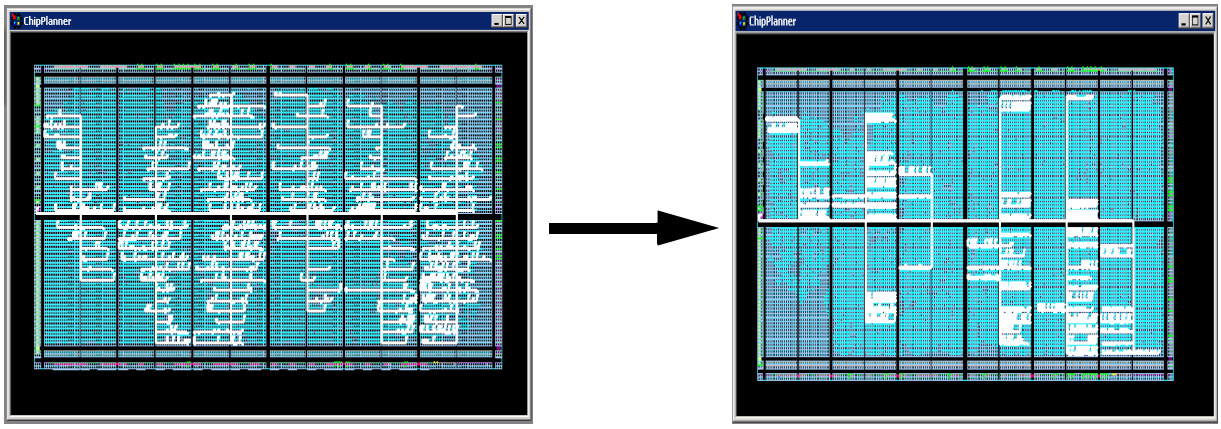

***Figure 6* • Power Reduction Using Clock Domain Floorplanning**

Here are some guidelines for clock domain floorplanning:

- Try to place the signal in a quadrant global network if possible. In general, the quadrant global has less capacitive load than the chip global.
- Check the fanout and determine whether you need to use the whole global network. If you do not need to use all spine locations, use region constraints to control the number of spines used, or use the clock aggregation feature.

Clock Conditioning Circuits (CCC) Power Reduction

Most flash-based FPGA families include one or more PLLs for frequency division, frequency multiplication, phase shifting, and delay operations. These PLL blocks can consume extra power. So for power-critical designs, avoid using PLLs whenever possible. For example, when you need to divide a clock, Microsemi recommends using a divider rather than PLL to generate a new clock frequency. Then, depending on the fanout, you can drive a clock network with the output of the divider. Figure 7 shows the dynamic power consumption comparison when using a PLL versus a simple divider circuit.

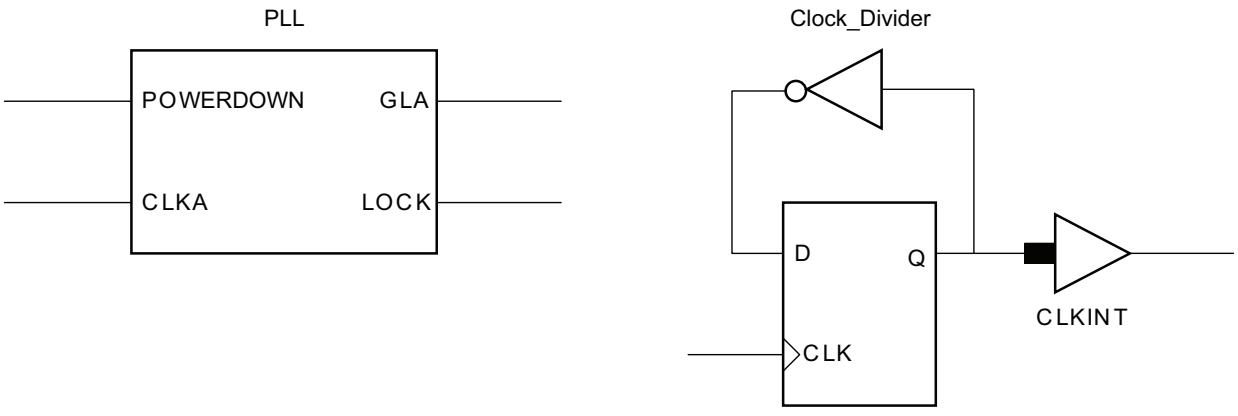

***Figure 7* • Power Comparison between PLL and Divider Circuit**

If you must use a PLL, the various multipliers and dividers around the PLL core usually allow more than one combination to synthesize one or several output clocks. Optimizing the choice of combinations that reduces the maximum PLL output frequency helps reduce power. In addition, the PLL in a flash-based FPGA has a POWERDOWN input pin to the PLL core. You can use this input to power down the PLL and clock networks when not needed.

Frequency Scaling Using Dynamic PLL

The dynamic reconfiguration feature of the PLL allows you to dynamically change PLL output frequencies. Refer to the "Dynamic PLL Configuration" section in the "Clock Conditioning Circuits in Low-Power Flash Devices and Mixed-Signal FPGAs" chapter of the *ProASIC3 FPGA Fabric User's Guide* for detailed information on dynamic PLL uses. The dynamic PLL offers additional power savings for applications that allow for clock frequencies to be reduced when the application is operating in certain modes. This feature must be controlled at the system level by a processor or other logic that can change the frequencies with respect to the operating mode.

Reducing Logic Power

Careful selection of appropriate arithmetic blocks is a source of large power savings in computation-oriented designs. This section goes over the power consumption of various arithmetic blocks and provides recommendations for reducing logic power.

Adder and Multiplier

The synthesis tool library offers a wide variety of arithmetic blocks with several architectures to better fit the area and performance needs of designs. Figure 8 and Figure 9 on page 10 show the power results for the various options of multipliers and adders. The following architectures are used for multiplier implementation: Carry-Save-Adders multiplier (CSA), Charge State Multiplier (CSM), Wallace and Non Booth Encoding Wallace (NBW) architectures.

For adder, the architectures are the Forward Carry Look Ahead (CLF), the Brent and Kung (BK), the Carry Look-Ahead (CLA), the CSM, and the Ripple (RPL) adders. These multipliers and adders are provided by DesignWare, the Synopsys® macro generator. However, they are also available through various online resources. In general, the power numbers in the arithmetic blocks are dominated by routing and correlate to a wide spread of the wire lengths. However, a reasonable selection rule consists of replacing all the adders in the critical path to infer the Brent and Kung architecture or multiplier-to-NBW architecture.

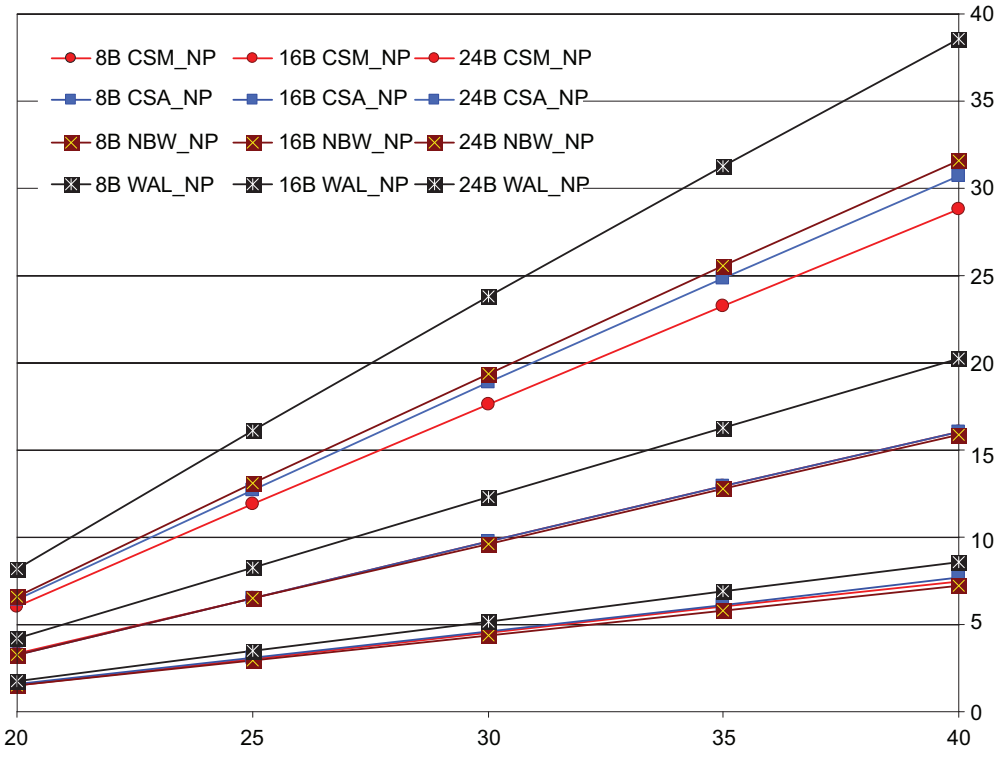

***Figure 8* • 8-, 16-, 24-Bit Multipliers Power Profiles**

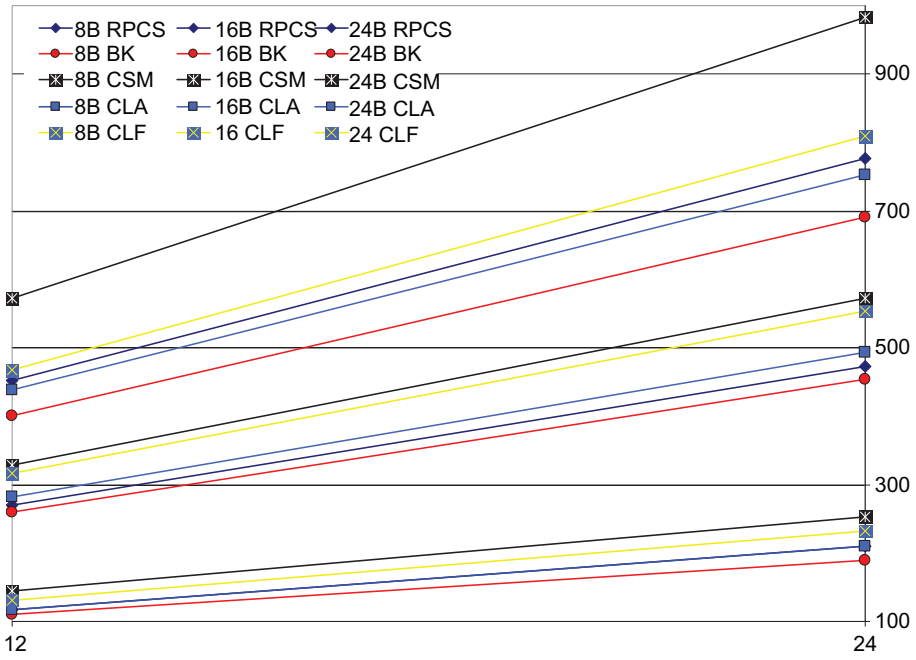

***Figure 9* • 8-, 16-, 24-Bit Adder Power Profiles**

Counter

Counters are commonly used throughout the designs for many different functions, such as keeping track of elapsed time, loading RAM address or data busses, controlling a state machine's next state, or output logic. Figure 10 on page 11 provides the power profiles for Gray, binary, and ring counters. It reveals that the binary counters offer the lowest power, followed by the Gray, while the composition of ring counters has the worst power figure, mainly because of the large load on the clock. A Gray counter consumes more power because it requires more logic to compute next count. This does not mean to systematically use sequential binary counters to reduce power. Gray counters driving a large load bus allow better power saving than binary because of lower switching rates at outputs. Moreover, both Gray and binary counters require more logic and will burn more power if a large number of the counter values need to be decoded.

Ring counters could be the best power-friendly solution if a uniform composition is used and when several counter values are decoded.

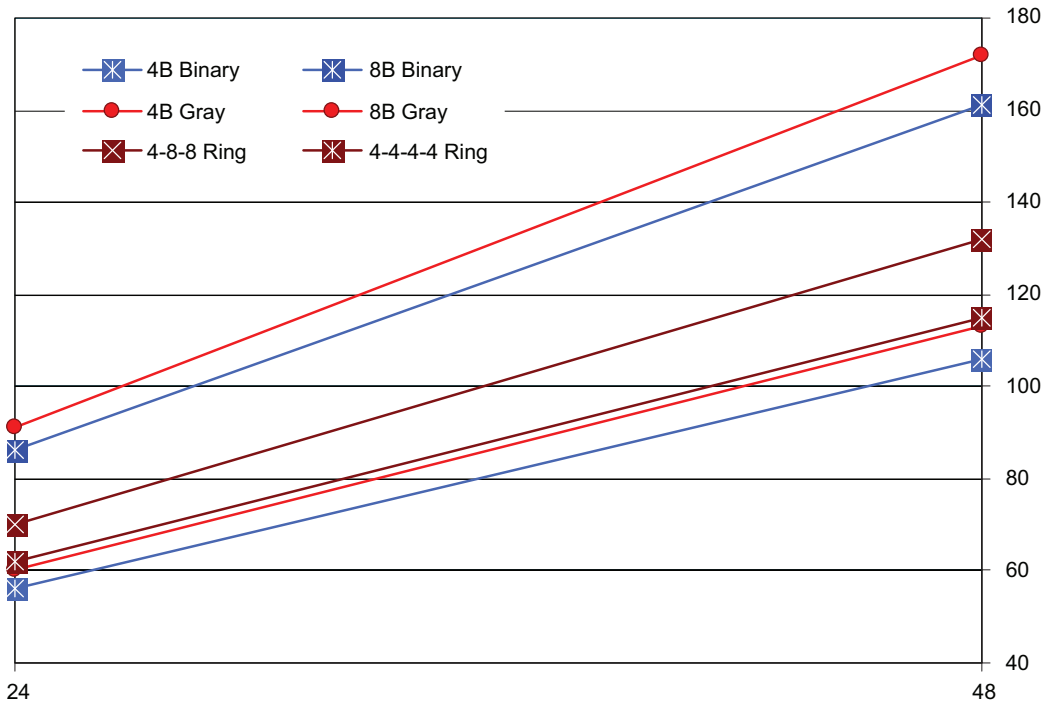

***Figure 10* • Binary, Gray, and Ring Counter Power Profiles**

Finite State Machine (FSM) and Counter Encoding

The selection of the state assignment depends on several parameters, such as the complexity of the state machine (the number of states), the number of paths and their lengths, the number of fork situations, and the complexity of the predicates on transitions between states. Depending on the type of state machine, the designer can choose among Gray, one-hot, or in-between styles. Most synthesis tools provide a way to select the encoding style during synthesis. Gray encoding uses fewer transitions than binary encoding. Sometimes it might not be possible to Gray encode all the states. In this case, you can increase the number of flip-flops in the state vector to minimize the number of flip-flops that are switching. Another alternative could be one-hot encoding. Though one-hot encoding uses many more flip-flops, it might reduce the use and the depth of combinatorial logic. This is especially the case for state machines with several outputs, when each output is a function of several states. In one-hot encoding, every state is already decoded, so this would use less logic for the outputs.

To summarize, you should control the encoding scheme for low-power design. The most effective way is to write the RTL directly into the intended encoding as opposed to letting synthesis decide the state encoding, because you can explicitly select encoding to minimize the number of bit changes per transition.

General Glitch Reduction Techniques on Logic

Glitches are unwanted switching activities that occur before a signal settles to its intended value. On every active clock edge, glitches can occur within combinatorial logic. This is because every node has a different delay, which means combinatorial logic may change states several times before settling down. Glitches on a node are dependent on the logic depth to that node—the number of logic gates from the node to the primary inputs (or sequential elements). The deeper and wider the logic cone behind a node, the more it will glitch. Unstable logic expressions, unbalanced sets of paths driving a sensitive combinatorial cell, or a MUX select line that can toggle several times within a clock period are examples of sources of glitches. Some of the glitches are absorbed by the cell delays and do not propagate. Some others, however, do propagate and can affect the power dissipation.

Some general glitch reduction techniques follow.

Glitch Reduction by Rearranging the Logic

In this technique, you rearrange the logic to move the glitch downstream. Consider the block diagram shown in Figure 11. The data A, B, C, and D are stable because they come from registers. But if the control signal for the multiplexers is oscillating, the operands for the adders are unstable and consume extra power. Moving the adders upstream and then multiplexing them consumes much less power because the adders see stable input.

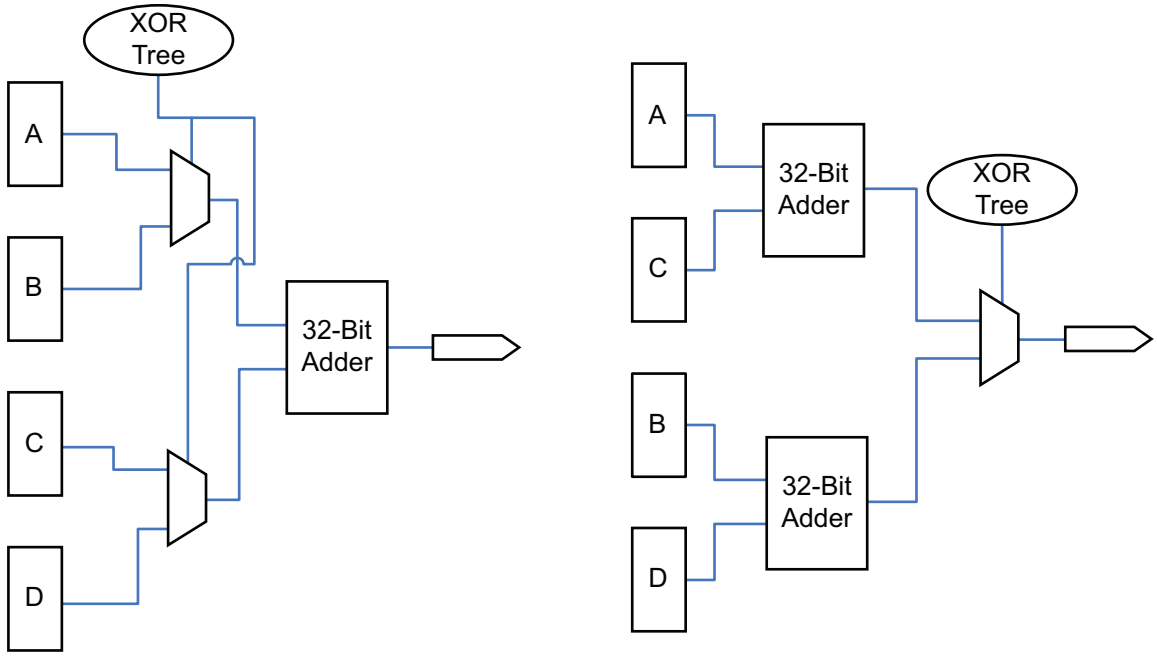

***Figure 11* • Glitch Reduction by Rearranging Logic**

Glitch Reduction by Partitioning and Using Different Optimization Level

A circuit synthesized for timing has more area but fewer levels of logic. This reduces the glitches on the nodes in the circuit, thereby helping with power. However, if the area penalty is too high, it may reduce the power advantages. In such cases, signals with a high switching rate should be mapped to reduce the levels of logic. Other paths with low switching rates can be optimized for area. Partition the design between high-switching and low-switching paths, and apply appropriate optimization.

Glitch Reduction by Pipelining

Pipelining is a technique that involves introducing the registers in the middle of long combinatorial paths. This adds latency but increases the speed and reduces the levels of logic. The introduction of extra registers consumes power but minimizes the glitches drastically. For example, a two-stage pipeline 16x16-bit unsigned multiplier generated from SmartGen consumes less power than its unpipelined counterpart.

Glitch Reduction by Inserting AND Gate on the Net Driver

Another method is to insert an AND gate on the net driver that is gated by the inactive clock level. This requires enough timing slack to allow the insertion of an AND gate driven by the "glitchy" net driver and the clock opposite edge, as depicted in Figure 12 on page 13. The only caveat is that each time the clock signal goes high, the output of the inserted AND gate will transition to low. This extra toggle is acceptable because it replaces a larger number of unnecessary toggles. There is no effect on functionality as long as the fanout of the AND gate gets into a logic cone that lands at a register input.

Inserting a transparent latch will also do the job, but timing analysis must be done carefully.

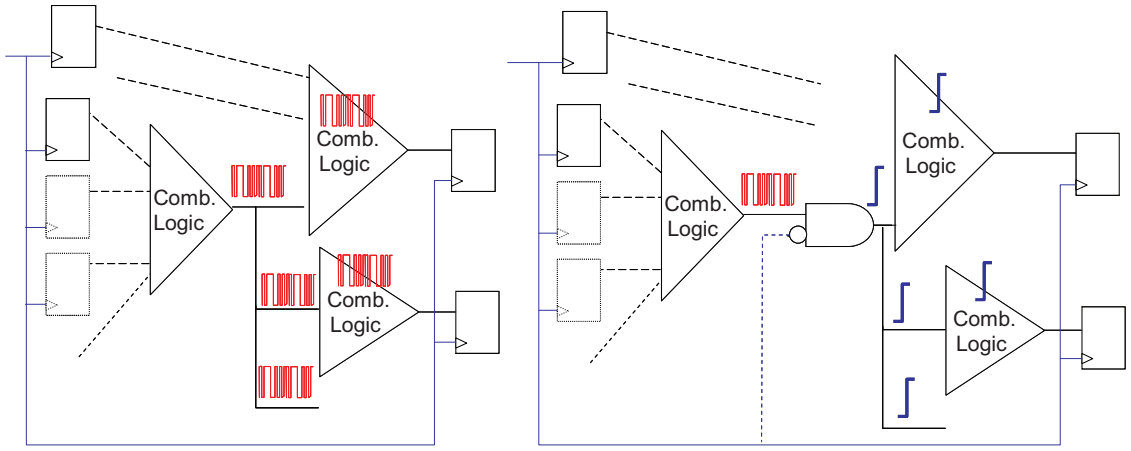

***Figure 12* • Glitch Reduction Using AND Gate on Net Driver**

Logic Depth Reduction for Frequently Switching Signals

You can move glitchy or fast-changing signals down the logic cone by reordering if-then-else constructs, as shown in Figure 13. This reduces switching activity propagation and power consumption. During synthesis, high-switching probability input signals to the combinatorial logic can be modeled as late-arriving signals. The synthesis tool tries to reduce the levels of logic from these inputs, thus minimizing switching activity propagation. If all inputs have the same switching probability, it is better to synthesize the logic as a balanced tree. Figure 14 shows an example of moving switching probability.

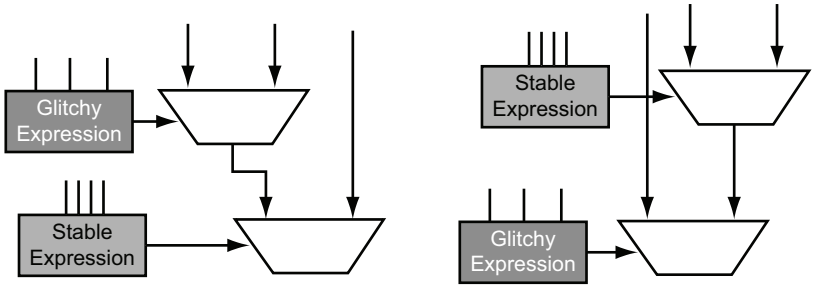

***Figure 13* • Power Reduction by Reducing Switching Activity Propagation**

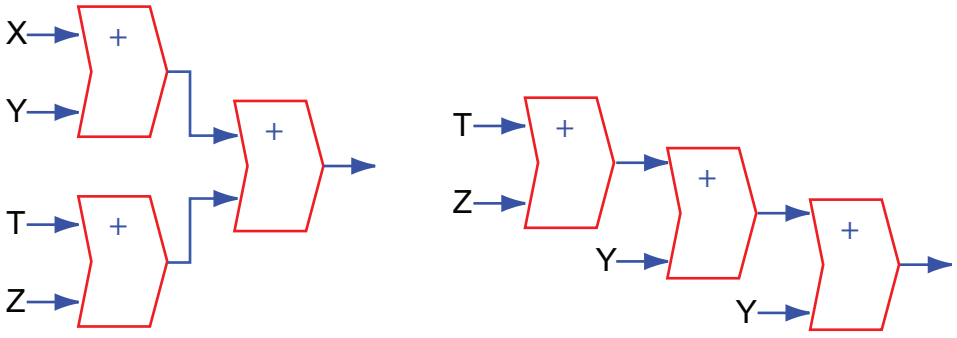

***Figure 14* • Power Reduction by Moving Switching Probability**

Using Enable to Stop Signal from Moving Downward Blocks

Decoder outputs are generally heavily loaded, so providing them as enable signals will stop unnecessary switching with outputs when the decoder is not in use, as shown in Figure 15.

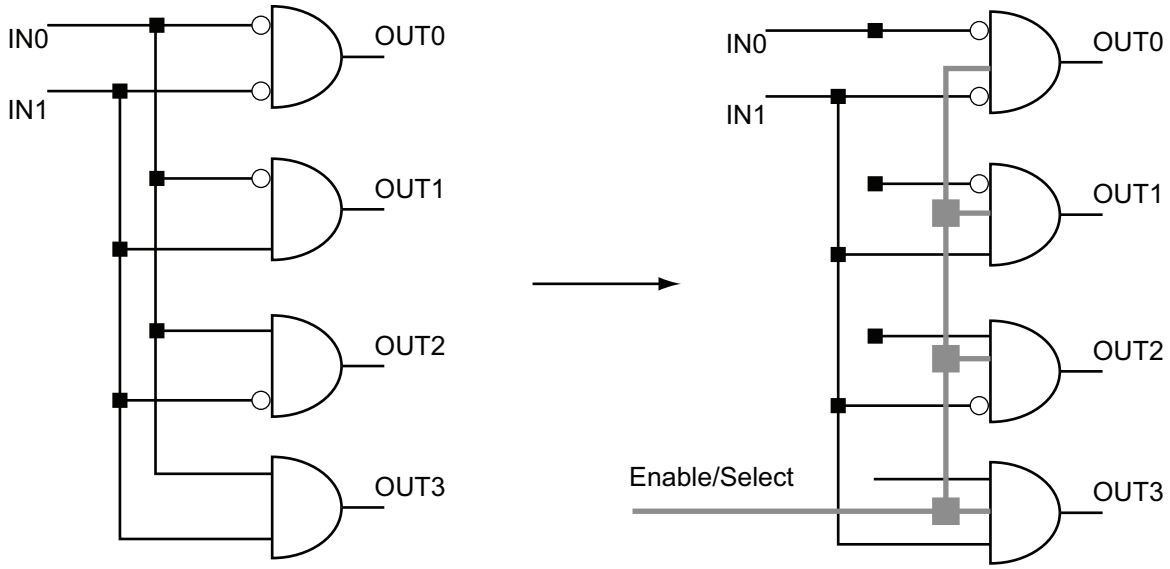

***Figure 15* • Using Signal Gating to Reduce Power**

Logic Toggling Activity Spread and Peak Power

Most designs use one edge of the clock to drive all registers in the design without an underlying timing need. When considering peak power, there is room for improving the peak consumption by revisiting the clocking. Figure 16 illustrates the concept of changing the clocking schemes of a portion of registers while remaining functionally equivalent. This change modifies the logic toggling profile and distributes power on both edges. Note that the overall energy dissipated is the same in both the schemes.

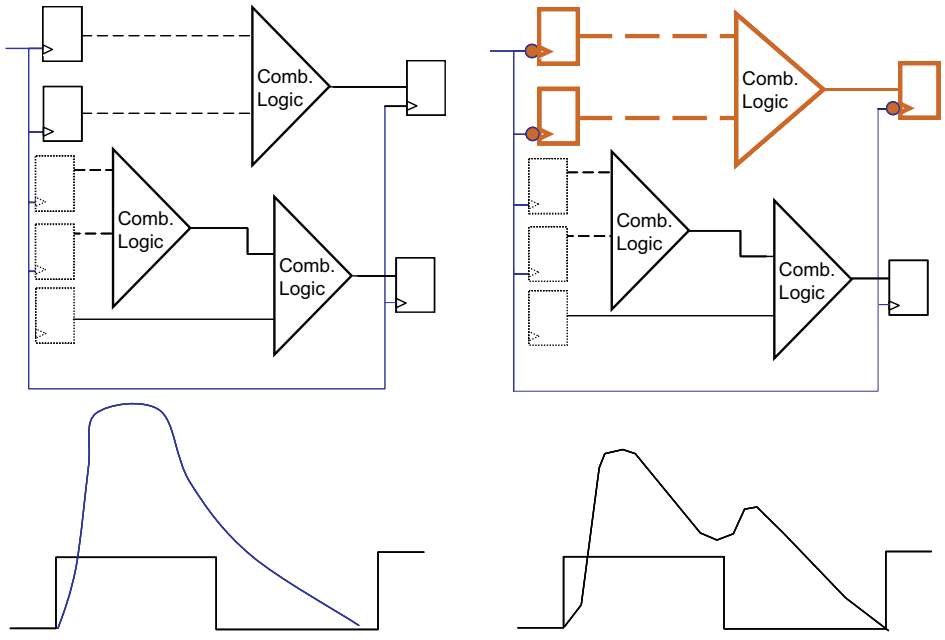

***Figure 16* • Clocking Schemes Changes and Logic Activity Spread**

Use Logic Floorplanning

For local nets, Microsemi recommends using minimum wire length to reduce power. Microsemi flash-based FPGAs have ultra-fast local resources, which are dedicated lines that allow the output of each VersaTile to connect directly to every input of the eight surrounding VersaTiles. So, when using a ripple adder, you should try to use these ultra-fast local lines. Figure 17 shows the placement of the 2-bit ripple adder. The layout tool can use ultra-fast local routing if logic elements are placed adjacent to one other. This floorplanning tip should be used only if additional power reduction is necessary after power-driven layout in Designer has been run.

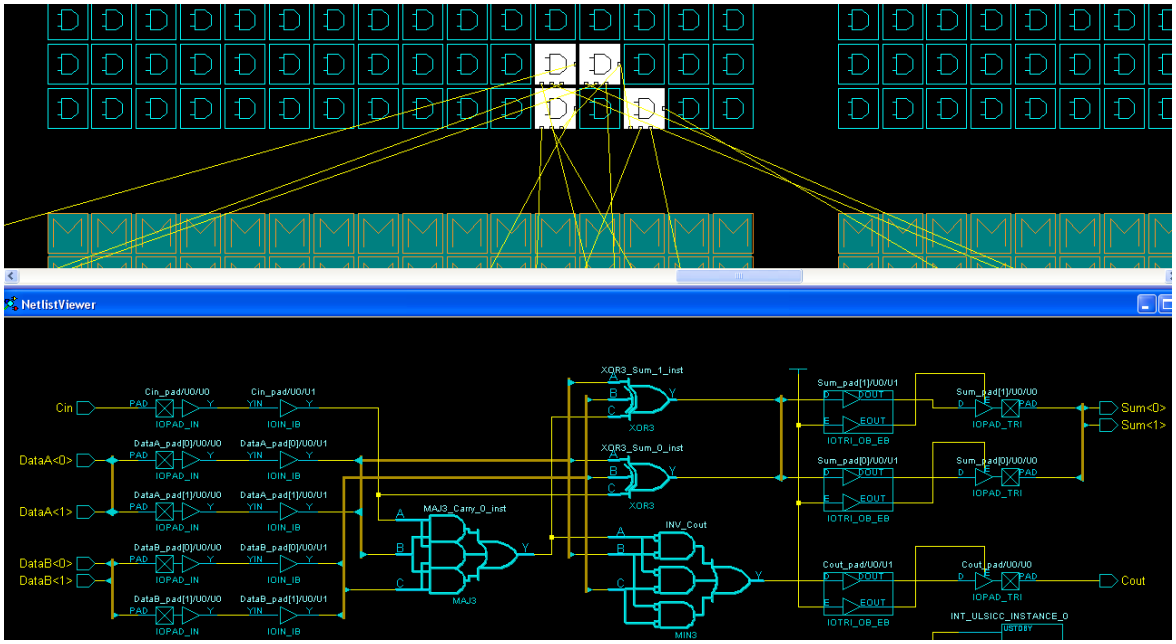

Figure 17 • **Use Local Floorplanning**

Reducing RAM Power

In flash FPGAs, the sources of RAM power consumption come from various circuits. For the read operation in RAM, the following circuits are involved:

1. Address and control latches
2. Row pre-decoder
3. Read column decoder
4. Row final decoder
5. Read column decoder control
6. Sense amplifier
7. Data output MUXes and latches
8. Sense enable logic
9. Read control logic
10. Bit-line precharge

Most of the current will be dissipated by the sense amplifier, bit-line precharge, and data output MUX/latches.

For the RAM write operation, the following circuits are involved:

1. Address and control latches
2. Row pre-decoder
3. Write column decoder
4. Row final decoder
5. Write column decoder control
6. Write driver
7. Bit-line pre-charge

Most of the current will be used by the write driver and bit-line pre-charge.

Analyzing power consumption during various operations brings to light several techniques for lowering the power contribution of the RAM blocks.

Read or Write Operation Impact on RAM Power

Figure 18 below provides silicon power measurements during the read and write operations for 12, 16, and 24 cascaded RAM blocks (with binary and Gray address schemes) in one of the flash FPGA devices. The write operation consumes slightly less power than the read access, so changing the RAM operation could reduce power consumption and reduce peak power.

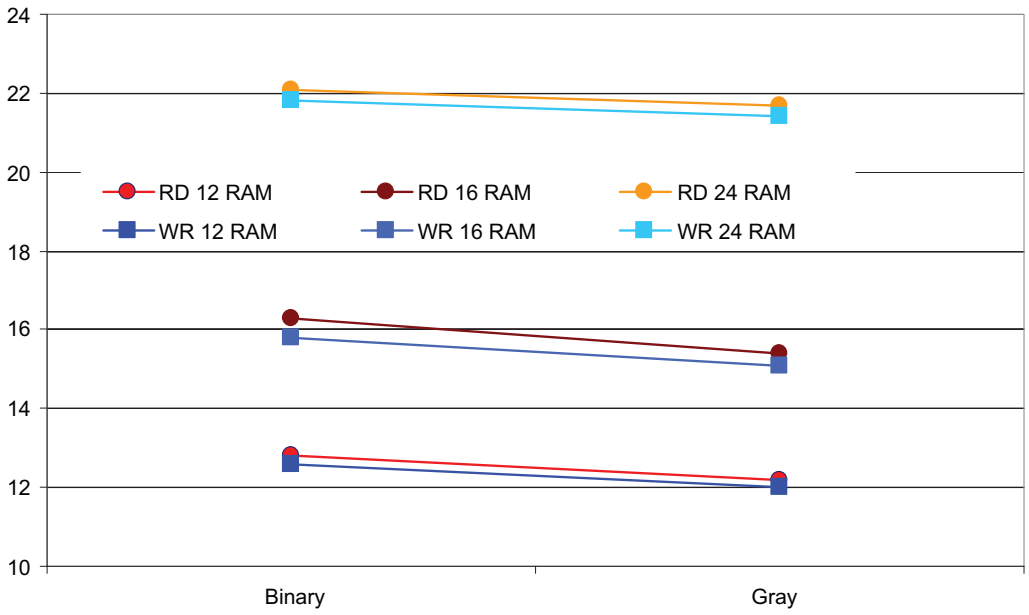

***Figure 18* • Read vs. Write Dissipation for Various RAM Sizes and Address Encoding**

Address Impact on RAM Power

Figure 19 shows the impact of the address changes for successive read or write accesses on the RAM power dissipation. The larger the Hamming distance between successive addresses, the larger the power consumption. So, Microsemi recommends reducing the Hamming distances between successive addresses in order to minimize the RAM power waste.

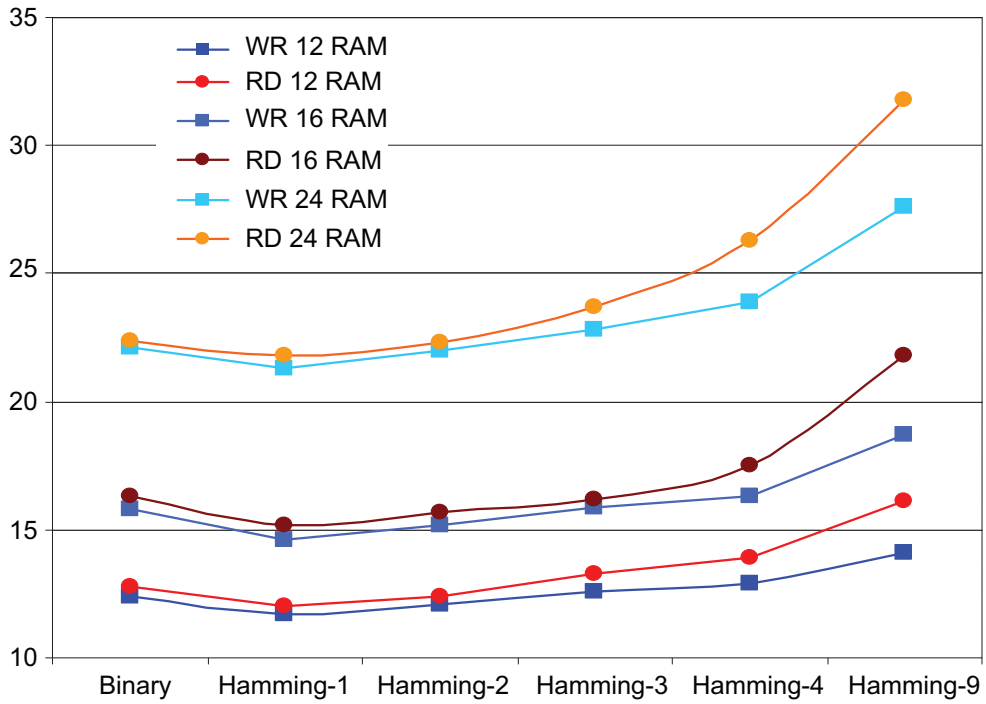

***Figure 19* • Successive Addresses Hamming Distance – Read and Write RAM Power**

Enable Signal Impact on RAM Power

Most designers write RTL without paying attention to the sequence of reads and writes. Figure 20 shows the difference between the power consumed when 1) a write is followed by a read and 2) three successive writes are followed by three successive read operations. Based on the analysis, Microsemi recommends doing consecutive writes or reads rather than switching between write and read more frequently.

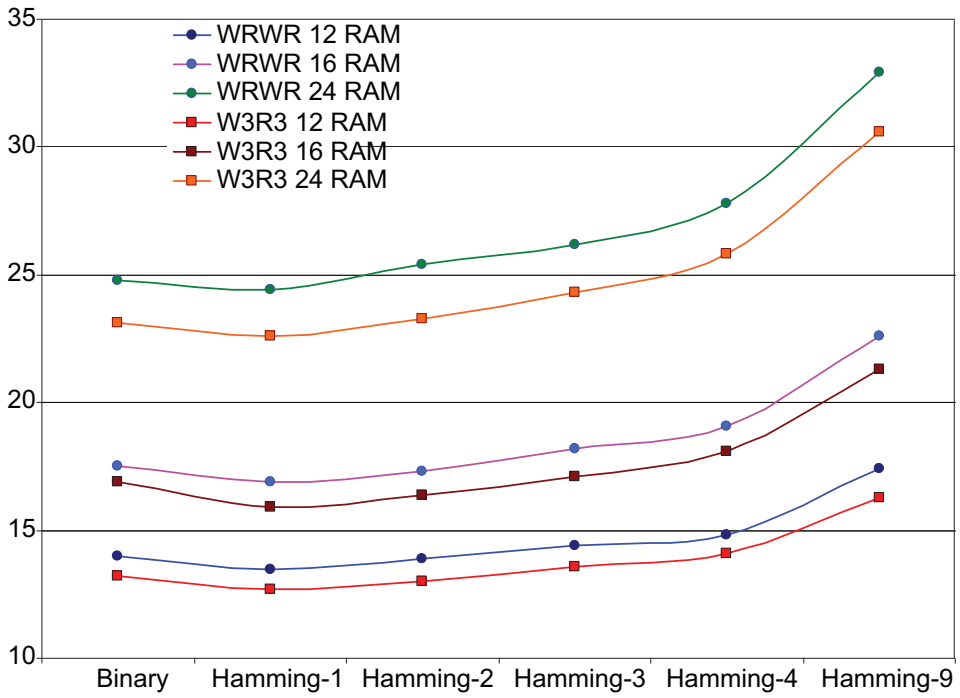

***Figure 20* • Impact of Write and Read Sequencing on RAM Power**

Low-Power RAM Cascading Scheme

Flash FPGAs offer several embedded RAM blocks with unique sizes but variable aspect ratio. Variable aspect ratio opens the door for different cascading schemes for RAM sizes exceeding block capacity. Figure 21 on page 19 is an illustration of two alternative implementations of a 4Kx4 RAM block. Each implementation has different timing and power attributes. In one case, all the RAM blocks toggle at each clock cycle as their outputs are concatenated to build the 4-bit output. In the second case, only one RAM block is active at a time. However, the overhead logic of the output multiplexer could consume extra power and might also affect timing. You should determine whether sequential operations share address locality, which is true when address-generation logic addresses one RAM a large number of times before moving on to the next one. If the address locality is guaranteed, then cascading schemes where only one RAM is active at a time are viable.

Note: The SmartGen tool allows both implementations. Use the implementation that is best for a given design.

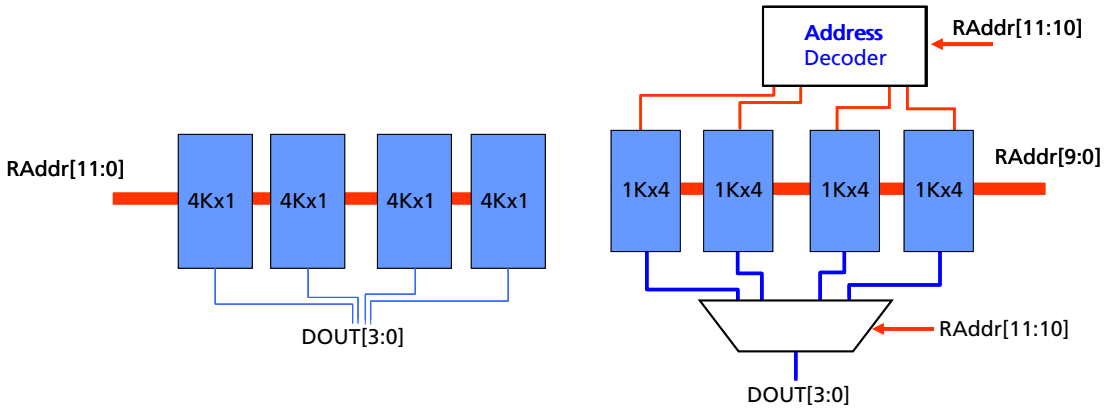

***Figure 21* • Potential Embedded RAM Cascading Schemes for 4Kx4 RAM**

Clocking Read and Write on Opposite Edge to Mitigate Peak RAM Power

When the same clock edge is used for the read and write, peak power consumption can be high due to simultaneous accesses on dual-port or two-port RAMs. Microsemi recommends using clocks with opposite edges for these ports. This method guarantees accesses are staggered in time and thus spreads the power dissipated by each access over time. Make sure that the timing constraints at the input and the output of one of the RAM access operations are met when inverting the clock.

Reducing I/O Power

The I/O power consumption is similar to logic power because it depends on switching, load capacitance, frequency, and voltage. By reducing any one of these components, you can reduce the I/O power. Here are some guidelines for reducing I/O power:

- Choose low V_{CCI} for I/O. Changing V_{CCI} from 3.3 V to 1.5 V can save up to 80% of your I/O power. Note that any reduction in voltage has a quadratic effect on power. This also helps in reducing I/O bank static power by reducing capacitance.
- Reduce capacitive load to reduce the I/O power consumption.
- Use differential I/O standards (LVDS, LVPECL) and resistively-terminated I/O standards (HSTL, SSTLs) for highest toggling frequencies. Use single-ended I/O standards such as LVCMOS for low frequencies. The differential I/Os have higher static power, but have the lowest dynamic power because of the limited voltage swing. Assuming you have a choice as to which I/O standard to use, evaluate the options based on the anticipated activity of the I/Os. For example, I/Os that spend most of the time active will benefit from differential I/Os, but I/Os that are static most of the time may suffer from the higher static component of differential I/Os.
- Reduce the number of I/Os by eliminating I/Os that can be time multiplexed.
- Use bus encoding that helps reduce the number of toggling bits and correlates successive values on the bus.
- Use low output drive strength. For example: you may not need the default 12 mA drivers. If you switch to 2 mA, you could save up to 20% of your I/O power.
- Divide the active outputs into two groups, some at the positive edge of the clock and others at the opposite edge.

General Tips During Synthesis and Place-and-Route

Most of the FPGA synthesis tools do not have any special mapping for reducing the power consumption. They typically allow optimization only for timing or area. In general, less area means less power. But this is not always true. In addition, many FPGA designers do not pay too much attention to the synthesis process and use stringent global timing constraints. This usually leads to larger utilization of the FPGA logic elements and the associated routing resources. Logic resources could be conserved by studying the so-called "slack distribution" for each clock domain; namely, the number of paths violating the required timing specification and the overall severity of these violations. For the blocks that have relaxed timing or sufficient margin in all internal paths, careful area-oriented synthesis will lead to a sizeable reduction of the needed logic resources and thus lower the power dissipation for these blocks, giving you the best opportunity to create the lowest power FPGA design. Refer to Figure 22 for more information.

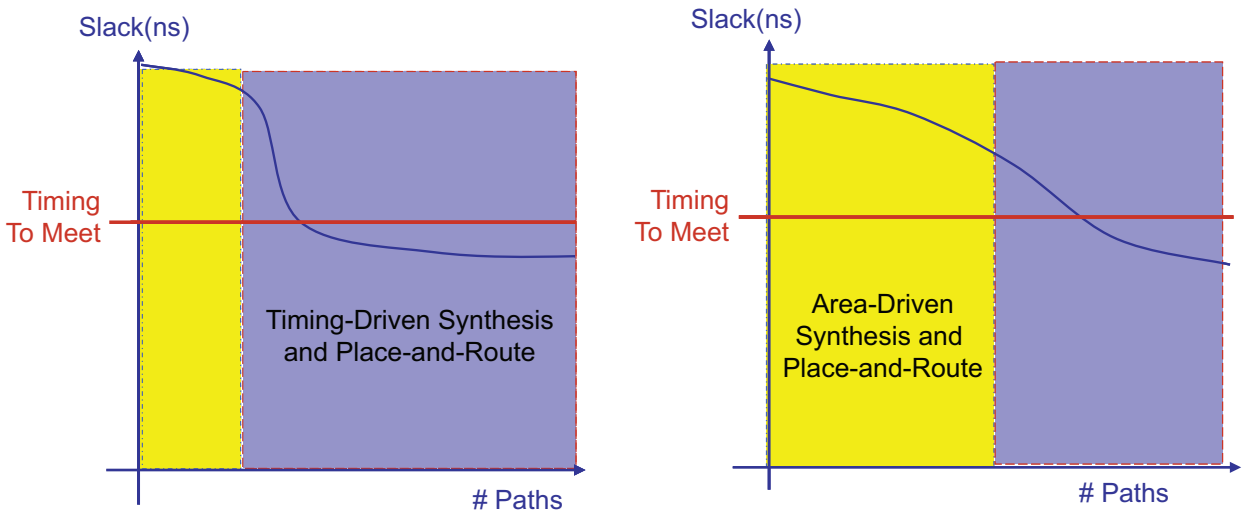

***Figure 22* • Potential Embedded RAM Cascading Schemes for 4Kx4 RAM**

The following are some guidelines for area optimization in Synplicity® synthesis tools:

- Focus on synthesis options settings and constraints and apply them appropriately to timing-critical blocks or sub-blocks. Analysis of slack distribution and avoidance of global synthesis settings are keys to improve logic dynamic power dissipation.
- Avoid over-constraining your design.
- Check the Resource Sharing option when you set implementation options. With this option selected, the software shares hardware resources such as adders, multipliers, and counters wherever possible, and minimizes area.
- For designs with a large state machine, follow the guidelines from the "Finite State Machine (FSM) and Counter Encoding" section on page 11 and write the RTL directly into the intended encoding.

The Microsemi Designer place-and-route tool provides the option to run power-driven layout in addition to timing-driven layout. The power-driven layout option drives placement of the design and reduces the power consumption in the following ways:

- Clock tree reduction by using fewer spines and rows
- Net capacitance reduction based on estimated activities
- Power-driven placement also works with activities estimation from timing constraints and average toggle rates.

To get the most out of power-driven layout, follow these recommendations:

- Use the VCD files from post-layout simulation.
- Enter maximum delay, minimum delay, setup, and hold constraints in SmartTime's constraint editor or in SDC.
- Set false paths on any paths that have a constraint but do not need one (this will help layout meet the constraints that are needed).

General Design Practice

For the design team, the approach is summarized below:

1. Starting from the dynamic power profile, identify the largest power contributor (logic, clock trees, RAMs, I/Os, etc.).
2. Depending on the identified power-bottlenecks, use Table 1 through Table 4 on page 23 and the following steps in order of priority:
   a. Use RTL changes as far as timing allows.
   b. Perform functional and timing validation after the changes. Run static timing analysis.
   c. Apply the synthesis hints.
   d. Use power-driven place-and-route (PDPR).
   e. Apply the floorplan and place-and-route hints.
   f. Validate your design.

Table 1: **Summary of Main Clock Tree Power-Aware Techniques**

Design Option	Premises and Alternatives	Estimated Saving
Clock Gating	Static timing analysis and functional validation required.	Could be substantial, but design-dependent
RTL Changes to Clock Groups of Registers with Opposite Clock Edge	Timing must allow for the change.	Peak power reduction depends on the size and routing in the logics driven by registers (could be substantial).
	Easy: No feedback loops	Up to 50%
RTL Changes for Pipelined Logic	Challenging: If feedback loops exist between stages, will require detailed static timing analysis.	10%+
RTL Elimination of Unnecessary Pipeline stages	Requires functional and timing validation.	Depends on the size of the eliminated registers
Timing-Driven "set_max_fanout" Setting	Apply if large number of high fanout nets is important and relaxed timing.	Depends on the number of eliminated register replications
Area-Oriented Synthesis	Enough positive slack margin and sizeable blocks	Depends on the size of the timing relaxed blocks
Power-Driven Place-and- Route	Need to combine it with timing constraints and timing-driven place-and-route.	Up to 10%
Clock Tree Floorplan	Analysis of implicit placement constraints and potential artificial congestion	Depends on die size and span of clock tree

Table 2: **Summary of Main I/O Power-Aware Techniques**

Design Option	Premises and Alternatives	Estimated Saving
Reduce the Number of I/Os	Partitioning and/or time multiplexing	The number of I/Os contribution per I/O (I/O standard, voltage swing, and toggle rate dependent)
Use Tristate Out Buffer Instead of Plain Output Buffers	Drive the enable signals wisely for tristate buffers.	Proportional to enable toggle rate
I/O Toggle Rate	Investigate bus encoding.	Depends on bus encoding quality
I/Os Toggle Timing	Stagger I/O toggling in time if static timing analysis allows.	No gain but reduction of peak power
I/O Standards Selection and I/O Bank Assignment	For high-frequency signals, use differentials or voltage-referenced standards. For lower frequency signals, use single-ended with least voltage, provided timing and waveform requirements allow.	The higher the frequency and lower the voltage swing, the higher the savings

Table 3: **Summary of Main Logic Power-Aware Techniques**

Design Option	Premises and Alternatives	Estimated Saving
Area-Oriented Synthesis	Static timing analysis shows enough positive slack	Could be substantial but design-dependent
Power-Friendly Arithmetic Blocks	Timing and area attributes of blocks required	5% to 15%
Power-Friendly Counters	If only final count is used, binary is best. If counter used to drive bus, Gray is best. If several counter values need decoding, ring counters are best.	Depends on number and size of counters as well as use model
RTL Changes for Glitch Reduction	Insertion of registers in multi-cycle and false paths	Depends on logic size in false and multi-cycle paths
	Insertion of AND gate driven by opposite clock edge; requires minor positive slack in "glitchy" paths.	Depends on size of downstream logic and glitching activity
	Insertion of latches driven by opposite clock edge requires static timing analysis.	Depends on size of downstream logic and glitching activity
Power-Driven Place-and-Route	Need to combine it with timing constraints and timing-driven place-and-route	5% to 10%

Table 4: **Summary of Main RAM Power-Aware Techniques**

Design Option	Premises and Alternatives	Estimated Saving (depends on RAM Size)
Cascading Scheme	If overhead logic (MUXing and decode) is timing critical, and if addresses are not local, adopt cascading with all blocks active; elsewhere use the proposed alternative (refer to the "Reducing RAM Power" section on page 15 for more information).	"-2%" – 5%
Root and Leaf Gating of the Enable and Clock	If not possible, adopt root gating of the clock.	5% – 10%
Write/Read Access Sequencing	As many write operations before as many reads as possible. If read/write are simultaneous, spread them over 2 edges of the clock (if timing allows).	2% – 5%
Successive Address Changes	Use least Hamming distance between the addresses.	1% – 4%

Conclusion

This application note discussed various techniques for reducing dynamic power design in FPGAs and presented a methodology to help efficiently minimize the dynamic power consumption, starting from the analysis of power profiles and then minimizing the power using RTL coding and tools flow. Using Microsemi low-power flash-based FPGAs in conjunction with the low-power design techniques discussed in this application note will assist you in creating an FPGA design with the lowest possible power consumption.

References

White Paper

Power-Aware FPGA Design

www.microsemi.com/soc/documents/Power_Aware_WP.pdf

Application Notes

Design for Low Power in Microsemi Antifuse FPGAs

www.microsemi.com/soc/documents/Low_Power_AN.pdf

Power Conscious Design with ProASIC

www.microsemi.com/soc/documents/A500K_PowerConscious_AN.pdf

List of Changes

The following table lists critical changes that were made in each revision of the document.

Revision*	Changes	Page
Revision 1 (June 2011)	Modified Figure 11 · Glitch Reduction by Rearranging Logic.	12

*Note: *The revision number is located in the part number after the hyphen. The part number is displayed at the bottom of the last page of the document. The digits following the slash indicate the month and year of publication.*

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo CA 92656 USA
Within the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

Microsemi Corporation (NASDAQ: MSCC) offers a comprehensive portfolio of semiconductor solutions for: aerospace, defense and security; enterprise and communications; and industrial and alternative energy markets. Products include high-performance, high-reliability analog and RF devices, mixed signal and RF integrated circuits, customizable SoCs, FPGAs, and complete subsystems. Microsemi is headquartered in Aliso Viejo, Calif. Learn more at **www.microsemi.com**.

51900184-1/10.12