WP285 (v1.0) February 14, 2008

Virtex-5 FPGA System Power Design Considerations

By: Peggy Abusaidi, Matt Klein, and Brian Philofsky

With the ever increasing power consumption trend in today's high-performance systems, managing the power within the system design budget is becoming as important as meeting the performance specification of the system design. Managing system power within the budget also has significant impact in maintaining overall system reliability.

This white paper offers design tips on changes that can be made to the FPGA environment, features, and tool options to optimize the system design power consumption, thus reducing thermal and power component cost as well as increasing overall system reliability.

Introduction

The semiconductor industry's rapid migration to the *next* advanced process node produces benefits of performance and cost, but these benefits have an adverse effect on the power budget. As transistor sizes decrease, leakage current, and hence static power, increases. Dynamic power also increases with the increase in system speed and design density—but in a more linear fashion. Today, many designs have 50-50 static and dynamic power dissipation. According to International Technology Roadmap for Semiconductors (ITRS) projections, static power is increasing exponentially at every process node, and innovative process technologies are imperative.

With the adoption of FPGAs in more system designs, FPGA power consumption within the entire system is becoming a critical part of the overall system budget. Consequently, leading FPGA vendors are adopting new techniques to mitigate the increasing static and dynamic power consumption. Xilinx uses triple oxide technology in its 65 nm process to reduce static power in Virtex™-5 FPGAs, while also providing embedded blocks and software tools to help system designers further optimize power in high-performance system designs. The latter part of this white paper gives more detail on the various power design tips including:

- Modifying the FPGA environment by varying the temperature and voltage
- Leveraging FPGA features and block-level control to reduce the dynamic and static powers
- Changing the software tool to use the power optimize options, reduce area, and using the pre- and post-implementation power analysis tools

Using some of the techniques described in this white paper, a memory interface design consisting of seven Virtex-5 FPGAs saw a power reduction of over 20 watts, from 110W to under 90W. The 18% power savings is the result of applying the I/O design tips provided in this white paper. The savings offers significant benefits in reducing thermal component and power supply cost and increasing overall system reliability.

Power Considerations

In the past, ASSPs and ASICs often consumed the majority of power in system power budgets. With the increase in FPGA performance, functions, and density, FPGA power consumption is now a key design consideration and must meet certain industry standards for maximum power allowed. For example, the communication industry has a standard for the maximum power allowed per rack. GR-63-CORE, a standard from Network Equipment Building Systems (NEBS), defines a power limit at shelf and frame levels (e.g., 4 KW/rack), operational and non-operational temperature limits, humidity range, and other environmental and safety levels for the equipment. Because of these power limits, a product's functional density, such as ports per rack (in networking equipment) can be limited by power consumption at the chassis, board, or FPGA level. In addition, power consumption is also closely linked to thermal consideration, which needs to be understood to keep the system working within its temperature specifications of the various components. The reliability decreases as parts are operated at higher temperatures, so keeping the temperature lower is also important.

Power Consumption Areas in FPGAs

There are two primary types of power consumption in FPGAs: static and dynamic power. Static power is consumed due to transistor leakage. Dynamic power is consumed by toggling nodes as a function of voltage, frequency, and capacitance. It is important to understand both power types and how each varies under different operating conditions so that they can be properly optimized to meet a design's power budget.

Static Power and Variation with Process, Voltage, and Temperature

Static power is power consumed by transistors due to leakage. The leakage current starts to be fairly significant at 90 nm for both ASICs and FPGAs, and becomes even more challenging at 65 nm. To obtain higher performance from the transistor, the threshold voltage (V_T) of the transistor, which also increases leakage, needs to be lowered. The variation in leakage and static power is about 2-to-1 between worst case and typical process. Static power and leakage are also strongly influenced by core voltage (V_{CCINT}), with variations that are approximately the square and cube of V_{CCINT}. Static power shows an approximate 15% increase with only a 5% increase in V_{CCINT}. Leakage is strongly influenced by junction (or die) temperature (T_J). Figure 1 shows the relative variation in transistor leakage, and hence static power, in 65 nm FPGAs, due to process, voltage, and temperature.

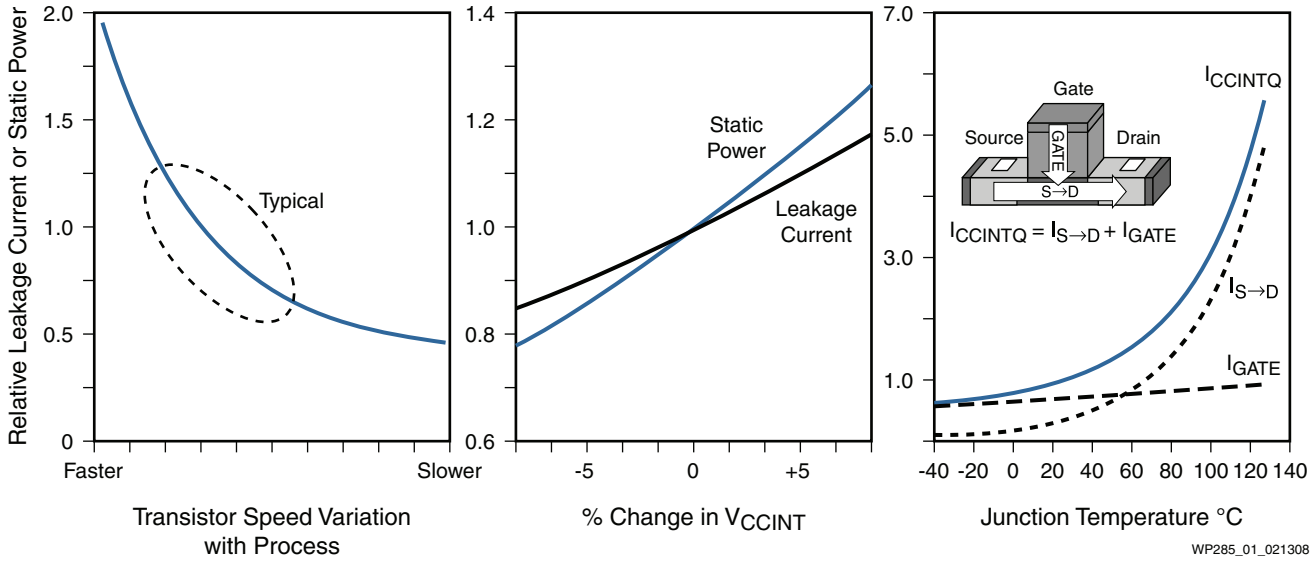


Figure 1: **Leakage and Static Power Variations with Process, Voltage, and Temperature**

Traditional FPGAs and ASICs only used two oxide thicknesses (dual oxide): a thin oxide for core transistors and a thick oxide for I/O transistors. Seeing increasing transistor leakage when moving toward high-performance 90 nm FPGAs, Xilinx integrated circuit (IC) designers started to adopt the use of a third-gate oxide thickness (*triple oxide*) in the transistors of the 90 nm Virtex™-4 FPGAs. The third medium thickness oxide (*midox*) and higher V_T in a portion of the transistors of the Virtex-4 FPGA allow a dramatic reduction in overall leakage, and hence static power, compared to other competitive FPGAs. Virtex-5 FPGAs continue to deploy the triple oxide technology in the 65 nm process node to enable a significant lower leakage current, about 38% lower than what the industry expects for a 65 nm device.

Dynamic Power: Variation with Process, Voltage, and Temperature

Dynamic power is the power consumed during switching events in the core or I/O of an FPGA. To calculate dynamic power, the number of toggling transistors and traces, capacitance, and toggling frequency must be known. Transistors are used for logic and programmable interconnects between metal traces in the FPGA. The capacitance consists of transistor parasitic capacitance and metal interconnect capacitance. The formula for dynamic power is given by:

$$P_{DYNAMIC} = nCV^2f \qquad \text{Equation 1}$$

Where:

n = number of toggling nodes

C = capacitance

V = voltage swing

f = toggle frequency

All nodes in the FPGA consume power through a combination of charging transistor parasitic capacitance and metal interconnect capacitance. The latter depends on the length of routes in the FPGA, while node capacitance is determined by the number of transistors that are switching. Tighter logic packing reduces the number of switching transistors and minimizes routing lengths, which reduces dynamic power. The 65 nm transistors in the Virtex-5 FPGA have lower gate capacitance and shorter interconnect traces. This combination contributes to lowering the node capacitance by about 15%, which in turn lowers the dynamic power.

Voltage also has significant effect on dynamic power. Moving from the 90 nm to 65 nm process node enables Virtex-5 FPGA designs to reduce dynamic power by approximately 17% simply by decreasing V_{CCINT} from 1.2V to 1.0V. This decrease in power combined with the 15% reduction in node capacitance provides a total of about 40% dynamic power savings as shown in Table 1. Process and temperature cause little variation in dynamic power, providing less than 5-10%.

Table 1: **Relative Dynamic Power Savings**

	Virtex-4 FPGA 90 nm	Virtex-5 FPGA 65 nm	% of Change	Power Ratio
V_{CCINT}	1.2	1.0	-16.6%	0.69
C_{TOTAL}	1.0	0.85	-15%	0.85
Power	1.44	0.85	-40%	0.59

Thermal Considerations and Reliability

Consideration must be given to thermal management at both the component and system levels to ensure that all devices are operating within their specified temperature range and to maximize overall system reliability. The device's operating temperature (junction temperature) is a function of the device power, its ability to transfer the resultant heat to the surrounding environment via the component packaging, and the ambient temperature of the system.

Reducing the device power consumption, therefore, has two significant benefits. First, it lowers system cost by enabling the use of less expensive thermal solutions to keep the device in its intended operating range. Second, reduced power means lower operating temperatures, which directly translates into improved component and system reliability.

External to the package, the additional components of thermal management become more varied. Due to the inherent FPGA flexibility, the user application can vary significantly in power consumption based on the particular combination of FPGA features utilized in the design and the performance level. Designs consuming less than 6W can simply require moderate airflow across the bare package, whereas designs with mid-range power consumption (5-10W) are likely to require external, finned passive heat sinks, which transfer heat away from the FPGA. Designs on the high end of power consumption (greater than 10W) should consider some form of active heat sink, such as a heat sink incorporating a miniature fan or thermoelectric cooler.

The location of the FPGA component within the system also determines the relevant thermal solution. Other important considerations include:

- Ambient temperature in the vicinity
- Proximity of other heat producing components
- Airflow around the device

With comparatively low Virtex-5 FPGA power dissipation, system designers have more options and more cost-effective solutions for managing both component and system-level thermal issues.

Additionally, the low power operation of the Virtex-5 FPGA also leads to a lower junction temperature. This in turn improves component and system reliability. This is best illustrated by examining the lifetime acceleration factor with respect to temperature. For example, compare two devices with only a 10°C difference in junction temperature:

- Device 1: T_J = 76°C (349°K)
- Device 2: T_J = 86°C (359°K)

Using a 0.75ev activation energy, the ratio of lifetime acceleration factors (AF) between Device 1 and Device 2 is then given by:

Equation 2

$$AF = \frac{t_{f1}}{t_{f2}} = e^{\left[\frac{E}{k}\left(\frac{1}{T1} - \frac{1}{T2}\right)\right]} = e^{\left[\frac{0.75}{8.6 \times 10^{-5}}\left(\frac{1}{349} - \frac{1}{359}\right)\right]} = 2.0$$

where

k = Boltzman's constant

E = Activation energy

T_1, T_2 = Maximum device junction temperature

Thus, a 10°C difference in junction temperature can result in a 2X acceleration in time to failure. This acceleration illustrates the reliability advantage of lower temperature operation for any system component. Combined, the lower power device operation and the thermally efficient packaging technologies in Virtex-5 devices facilitate cost-effective, highly reliable designs.

Tools for Power Design Consideration

Xilinx has two types of power analysis tools. These tools are separated into pre-implementation and post-implementation categories. The pre-implementation tool is the XPOWER™ Estimator (XPE) spreadsheet tool. It provides an estimate for initial power evaluation, which helps the designer with power supply selection and thermal considerations (see Figure 2).

XPE users enter the estimated resource utilization, clock frequency, toggle rate, and operating environment inputs to calculate the estimated power as shown in the power summary (Figure 2). The XPE tool offers the flexibility to do different iterations of a what-if analysis to determine the most optimal way to reach the power design target. The accuracy of the result is heavily dependent on the integrity of the user input. Xilinx provides detailed information in UG440, the Xilinx Power Estimator User Guide.

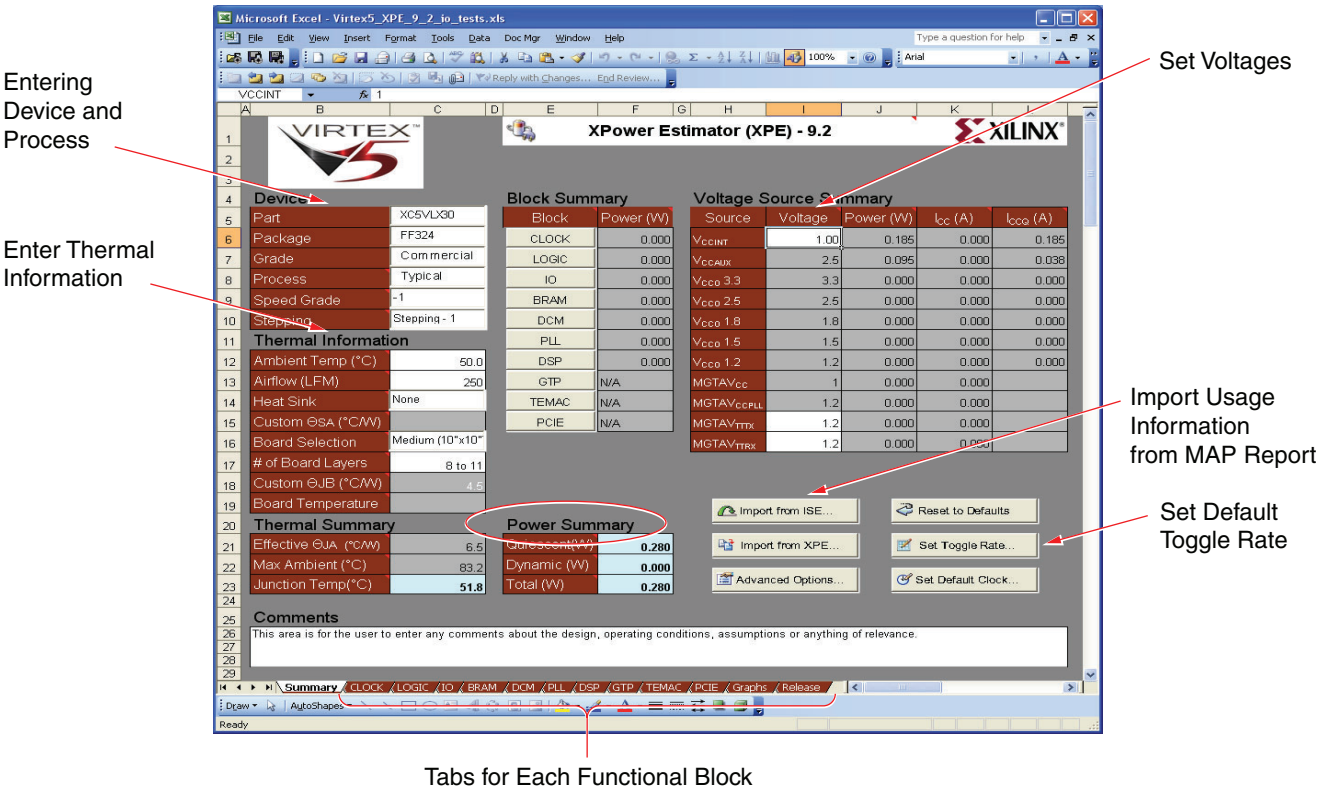


Figure 2: **Xilinx Power Estimator Tool for Initial Power Estimation**

The post implementation tool is the XPower Analyzer software. It gives designers a more accurate view of the power breakdown based on the exact resource utilization information extracted from the FPGA design implementation. It provides valuable data in the product development phase to understand where power is dissipated and focuses the effort on the correct area for power optimization. More information is available in the online help included in the XPower GUI tool.

Optimizing Designs for Power Consumption through Changes to the FPGA Environment

To optimize the power consumption in any design, certain things can be done independently of the design contained within the FPGA. Knowing one's environment, e.g., operating temperature and core voltage, is therefore important.

Temperature Control

Controlling temperature not only helps with reliability, as described in the "Thermal Considerations and Reliability" section, but it also reduces static power. For example, a reduction in junction temperature from 100°C to 85°C reduces static power by ~ 20%, as shown previously in Figure 1 and with greater detail in Figure 3.

The static power of Virtex-4 and Virtex-5 FPGAs is already reasonable. However, reducing it by another 20% is valuable because in some designs, the static power of the FPGA represents a sizeable portion (30-40%) of the total power budget.

A reduction in junction temperature can be achieved by increased airflow and larger heat sinks. The reduction in junction temperature also has the added benefit of increasing reliability as shown in the "Thermal Considerations and Reliability" section.

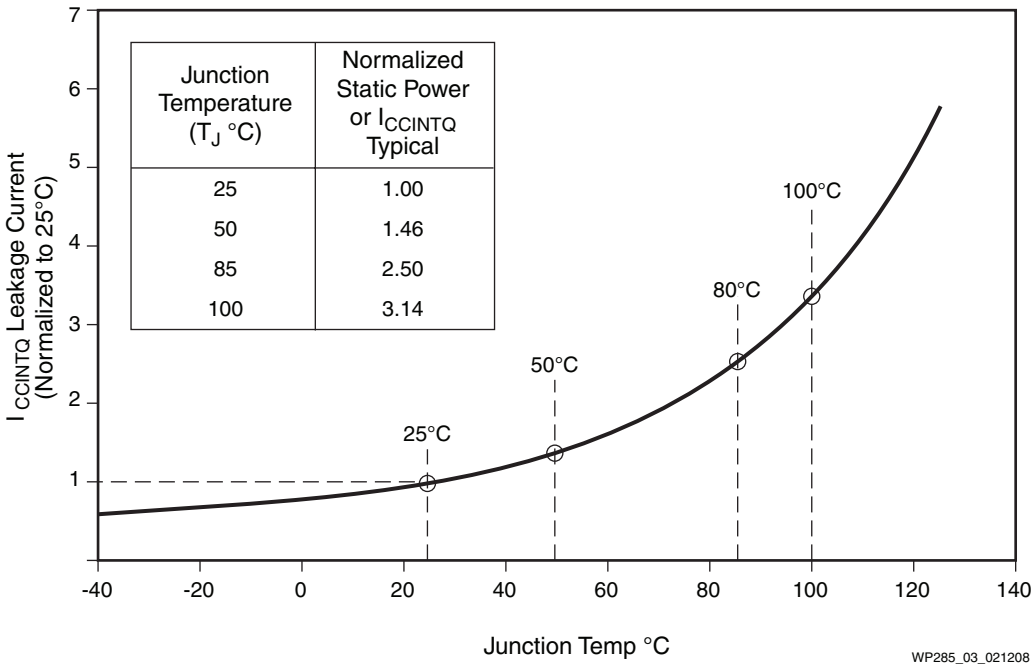


Figure 3: **I_{CCINTQ} vs. Junction Temperature with Increase Relative to 25°C**

Static power is a function of die temperature (T_J), and T_J is a function of how much power the device is consuming, the thermal properties of that device, and its package. Consequently, the FPGA's ability to transfer the resultant heat to the surrounding environment, via the component packaging, is very important.

Heat flows out of the die from the top of the FPGA and into the package balls and PCB, so it is important to understand the system model (PCB, FPGAs, heat sinks, airflow, and other components in a system). See Figure 4.

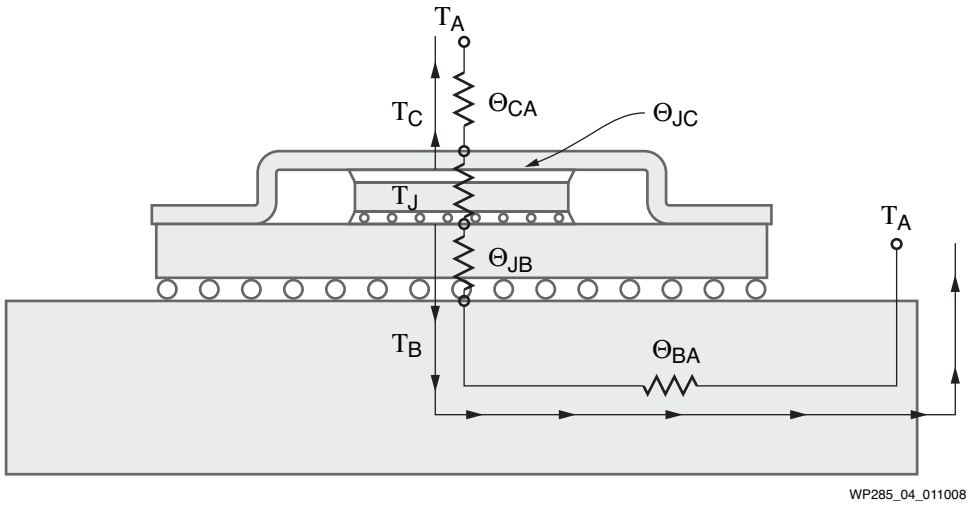


Figure 4: **Paths of Heat Flow and Thermal Resistance**

Because of shrinking thermal budgets, accurate temperature prediction is critical. Xilinx provides Compact Thermal Models (CTMs), which predict package temperature at the selected nodes (e.g., junction, case, and package balls) and work with industry standard thermal modeling tools. One of these tools is FLOWTHERM by Flomerics. Xilinx provides free downloads in FLOWTHERM format on xilinx.com for usage with customers' thermal simulation tools.

With the CTMs and the thermal modeling tools, the location of the junction temperature on the FPGA can be determined, leading the designer to the correct areas in which to reduce it, e.g., PCB copper, airflow, power supply, and FPGA design.

UG112: *Device Package User Guide* is a good reference on the thermal characteristics of Virtex-5 devices.

Voltage Control

Static and dynamic power consumed at the core voltage, V_{CCINT}, is often the largest power consumer in the FPGA. FPGAs are usually specified to run and meet performance with power supply voltages within ±5% of nominal.

Table 2 shows that ±5% variation in V_{CCINT} causes approximately ±15% and ±10% variation in static and dynamic power, respectively, in a Virtex-5 device.

Table 2: **Virtex-5 FPGA Variation of Static and Dynamic Power Due to V_{CCINT} Core Voltage Adjustment**

V_{CCINT} Adjustments			Power Consumption from V_{CCINT}	
Voltage at FPGA	Δ%	mV	% Static	% Dynamic
0.95	-5	-50	-14.3	-9.8
0.96	-4	-40	-11.5	-7.8
0.97	-3	-30	-8.7	-5.9
0.98	-2	-20	-5.9	-4.0
0.99	-1	-10	-3.0	-2.0
1.00	0	0	0.0	0.0
1.01	1	10	3.0	2.0
1.02	2	20	6.1	4.0
1.03	3	30	9.3	6.1
1.04	4	40	12.5	8.2
1.05	5	50	15.8	10.3

Notes:

1. All variations are shown relative to the highlighted Virtex-5 FPGA nominal V_{CCINT} of 1V.

The dynamic power variation with voltage is based on the standard dynamic power formula given in Equation 1.

The static power variation with voltage is based on semiconductor physics principles, measurement, and curve fit approximations and is approximated by the formula:

$$P_{STATIC} \propto V^3 \qquad \textit{Equation 3}$$

where V is the core voltage V_{CCINT}.

Voltage regulators sometimes have an accuracy as large as ±5%. In the specific case of V_{CCINT}, the variations in power shown in Table 2 can be mitigated by using a voltage regulator with a tighter tolerance, e.g., ±1%. Additionally, using voltage sense feedback from the FPGA allows the V_{CCINT} voltage to be kept at the nominal 1V level. Since the Virtex-5 FPGA can operate within its specifications even with a voltage of 0.95V, some power benefit can be achieved by adjusting V_{CCINT} to slightly below the nominal voltage, such as 0.98V.

Tighten Up the V_{CCINT} Power Supply Tolerance and Sense It at the FPGA

Even a small change in V_{CCINT} causes a reasonable change to static and dynamic power. It is therefore important to sense the V_{CCINT} voltage at the FPGA so that small IR drops can be accounted for. The IR drops are fed back to the power supply, which helps to determine the exact setting for the V_{CCINT} voltage.

If cost and board space allow, one regulator can be used for each FPGA. Dedicated regulators provide tight control of V_{CCINT} at each FPGA and reduce the static and dynamic power by optimally setting V_{CCINT} (see Table 2). The best setup for sensing the voltage for a Virtex-5 FPGA design is shown in Figure 5.

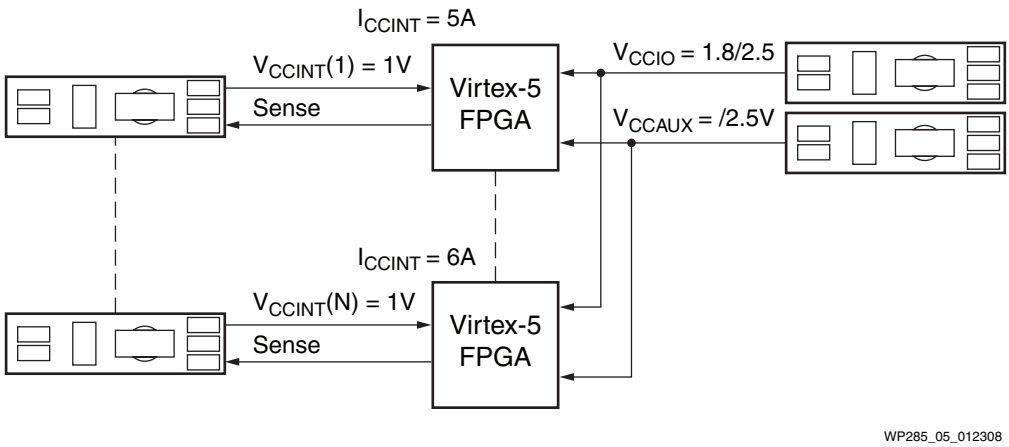


Figure 5: **One V_{CCINT} Regulator per FPGA for *N* Number of Virtex-5 FPGAs**

If one V_{CCINT} regulator must be used for multiple FPGAs, then it is best to sense the V_{CCINT} voltage at the FPGA and feedback to regulator at a point very near the FPGA, which is expected to be the highest current user of V_{CCINT}. This case is shown in Figure 6.

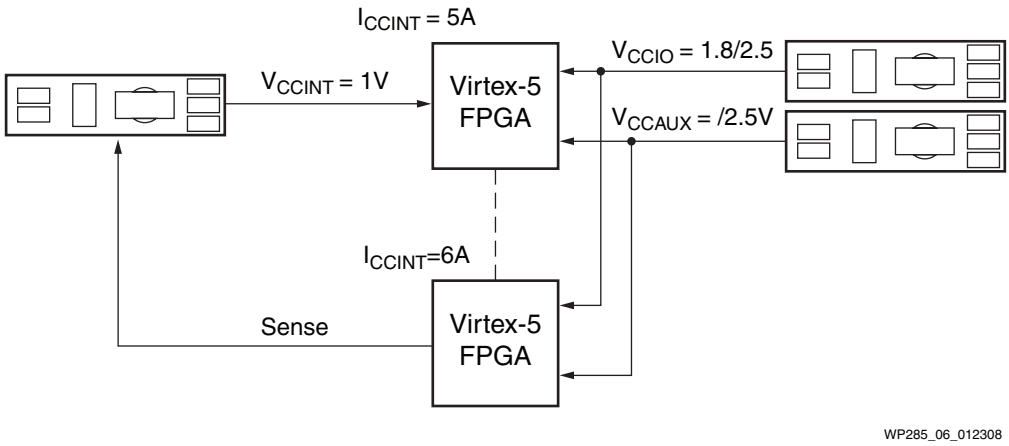


Figure 6: **Use of One V_{CCINT} Regulator for Multiple Virtex-5 FPGAs**

Both of the sensing techniques allow closer control over the exact V_{CCINT} voltage that is used by the FPGA and eliminate small PCB losses.

For the best power results, a Virtex-5 FPGA should be operated at a target voltage that is less than the nominal V_{CCINT} voltage of 1V. Setting V_{CCINT} to 0.98V reduces power while still allowing for some V_{CCINT} noise, without going below the minimum voltage of 0.95V for V_{CCINT}. This is safe because Xilinx FPGAs are guaranteed to meet timing and performance for a given speed grade and temperature grade over the full voltage range for V_{CCINT}.

Using FPGA Features More Optimally for Power Concerns

Block RAM (Internal FPGA Memories)

In many designs, block RAM can be a major contributor to the overall power dissipation within the FPGA. Thus careful selection, use, and composition of this resource are extremely important to realizing a low-power FPGA design. When designing the RAM portion of the design, the following should be considered:

- Choose the right RAM primitives for the job
- Ensure the block RAM is only enabled when data is needed
- Ensure the WRITE_MODE is set properly
- Used dedicated FIFO and ECC logic if needed

When choosing a RAM organization within the target architecture, the width, depth, and functionality must be considered. Choosing the right memory facilitates the selection of the most power-efficient resource for the end design. For wide (over 18 bits), shallow (64 bits or less) RAMs, it is generally advantageous to use SelectRAM™ memory (LUT-based RAM) when possible—unless the RAM is being used as a FIFO, in which case the crossover point becomes a depth of 32-bits or less. When building interfaces less than 18 bits wide, the LUT-based SelectRAM memory can be a better choice for depths up to 128 bits. Beyond 128 bits, dedicated block RAMs are a better choice for power. The XPE tool simplifies power selection. With the correct input on width, depth, and functionality, the XPE tool provides suggestions for the best memory selection.

Here is an example of an XPE tool exercise targeting Virtex-5 FPGA architecture. A designer wants to implement 16, 300 MHz, 32-bit-wide by 64-bit-deep Simple Dual-Port RAMs. This equates to a SelectRAM memory size of 512 (16 x 32). Registered outputs are associated with 512 registers. Using the current version of the XPE tool (v9.2i), the RAM configuration results are shown in Table 3:

Table 3: **XPE Results for 16, 300 MHz, 32-Bit-Wide by 64-Bit-Deep Simple Dual-Port RAMs**

Name	Clock (MHz)	LUTs	Shift Registers	Select RAM Memory	Flip-Flops	Toggle Rate	Average Fanout	Power (W)
	300.0	0	0	512	512	25.0%	3	0.050

If targeting the same implementation to block RAM, each 32x64 RAM can be placed into a single block RAM, which results in twice as much dissipated power. See Table 4.

Table 4: **XPE Result for Placing Each 32x64 RAM into a Single Block RAM**

				Port A					Port B					
Name	Block RAMs	Mode	Toggle Rate	Clock (MHz)	Enable Rate	Bit Width	Write Mode	Write Rate	Clock (MHz)	Enable Rate	Bit Width	Write Mode	Write Rate	Power
	16	RAMB18DSP	25.0%	300.0					300.0	100.0%	36		50%	0.100

After the appropriate RAM resource is selected, the next task is to ensure that the RAM is connected and configured in the most power-efficient way. The first and most important thing to consider is the management of the RAM. Using the analogy of managing a residential electrical bill by turning off lights when leaving a room, the enable should be turned off when not using a RAM. Why? Because the power requirements of a block RAM is directly proportional to the amount of time it is enabled. In other words, RAM that is enabled for only half of the clock cycles uses only 50% of the dynamic power than if enabled for all clock cycles. This is an important

detail because a common design practice is to tie a RAM enable High for ease-of-design purposes. However, efficient enable management can pay dividends in power savings. Again, using the XPE tool for analysis, the results are shown in Table 5.

Table 5: **XPE Results for Placing Each 32x64 RAM into a Single Block RAM**

			Port A					Port B					
Block RAM	Mode	Toggle Rate	Clock (MHz)	Enable Rate	Bit Width	Write Mode	Write Rate	Clock (MHz)	Enable Rate	Bit Width	Write Mode	Write Rate	Power
250	RAMB18	50.0%	250.0	100.0%	18	WRITE_FIRST	50.0%	250.0	100.0%	18	WRITE_FIRST	50.0%	2.383
250	RAMB18	50.0%	250.0	25.0%	18	WRITE_FIRST	50.0%	250.0	25.0%	18	WRITE_FIRST	50.0%	0.601

Table 4 shows the results from two sets of 250 identical RAMB18 configurations, one that is constantly enabled and one where the enable is set to 25% of the time (i.e., data is written/read every four clock cycles). The second configuration uses 25% of the power because the RAMs are enabled only 25% of the time, which in this case saves 1.78W (75% of the total power).

Enables can also be utilized for low power when more than one block RAM is needed to create a much larger memory structure. For example, a memory array that is 2k locations by 36 bits is often constructed as four sets of 2k x 9 block RAMs, as shown in Setup A in Figure 7. This is the easiest way to construct the desired 2k x 36 structure while still reaping the benefits of minimum area and maximum speed; however, this structure requires that all four 2k x 9 blocks are enabled at the same time.

Another alternative to reducing power is to have all four block RAMs configured as 512 x 36 and selectively enable one block RAM at any given time based on the address selection. In this case, four sets of block RAM are stacked, organized as 512 x 36 each. This arrangement requires surrounding logic for address decoding to enable only one of the four block RAM, decode the output multiplexer signals, and make the output multiplexer. Setup B in Figure 7 shows the new block diagram of the 2k x 36 array, formed as four blocks of 512 x 36 bits plus the extra logic. The trade-off in this method is power versus speed and area. In the block diagram, approximately 75% of the power is saved because only one of the four block RAMs is enabled at a time.

This power savings is at the expense of an output multiplexer, which slows down the access, and some extra logic. However, the overall power savings can be significant. Plus, the speed degradation can be mitigated by pipelining in some cases if the design permits.

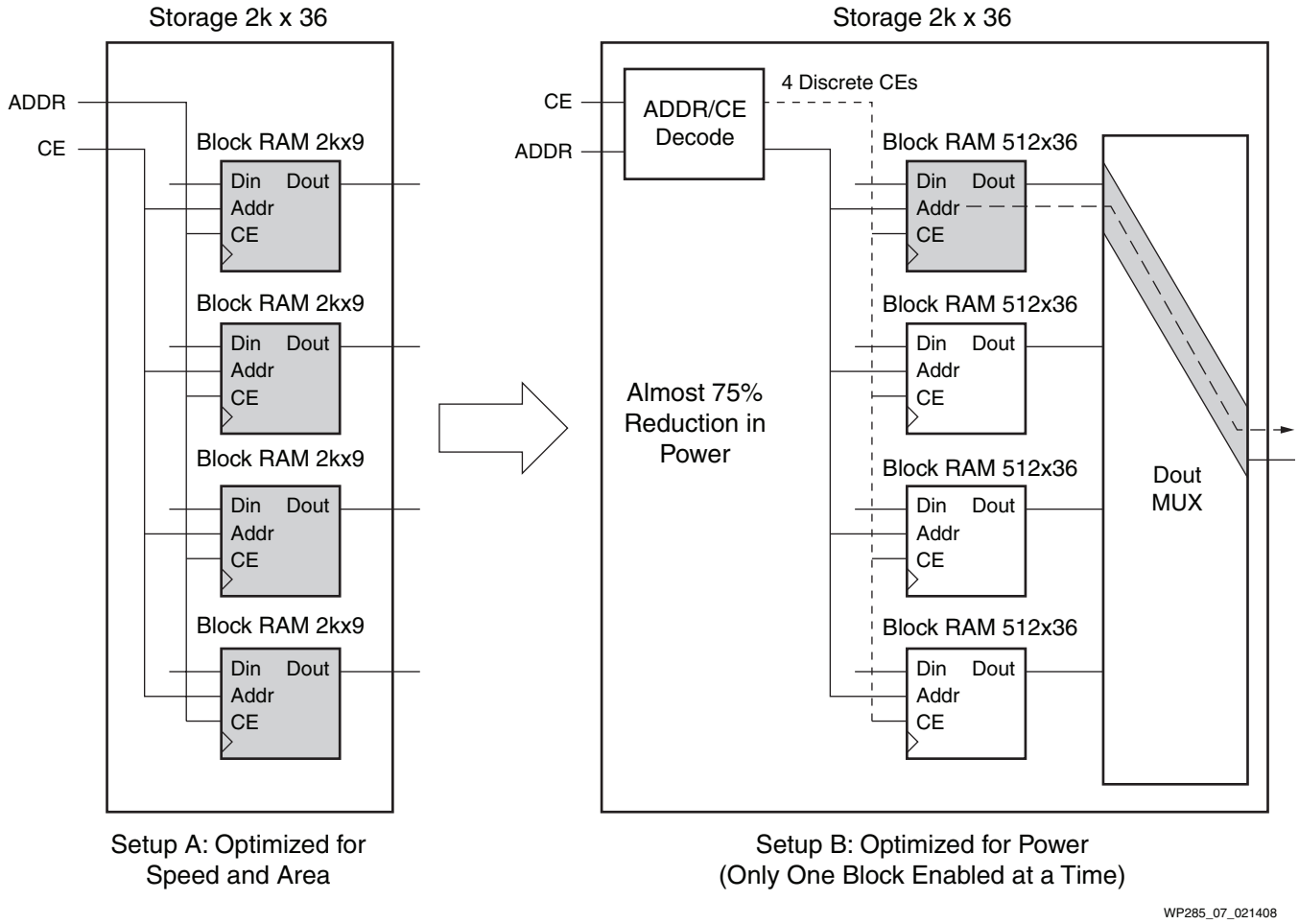
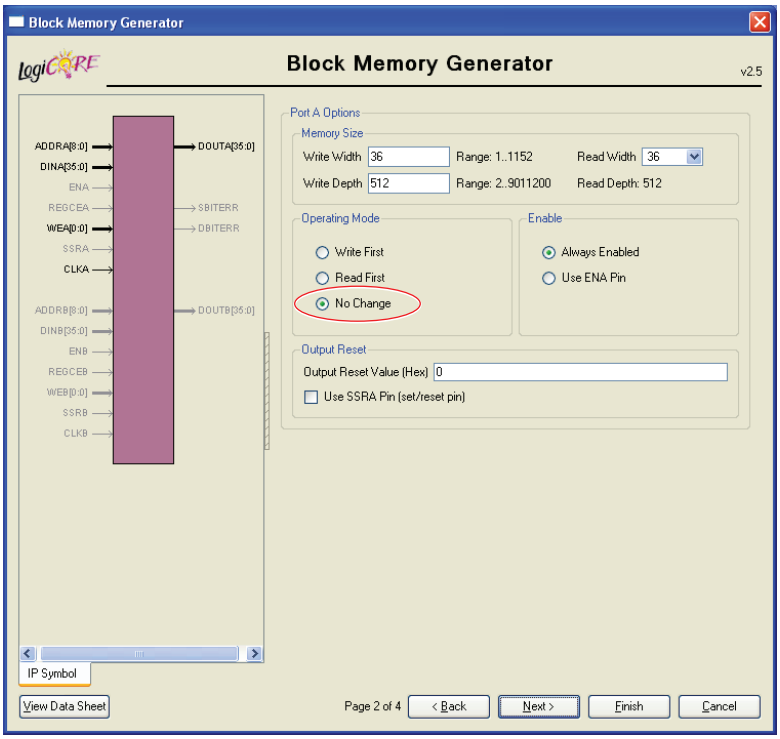


Figure 7: **Memory Organization Optimized for Speed Versus Power**

Another power reduction consideration in block RAM usage is the write mode. If the block RAM contents are never read during a write, the RAM power can be reduced by approximately 10% with the selection of the NO_CHANGE mode rather than the default WRITE_FIRST mode. The block RAM outputs in NO_CHANGE mode do not change during writes, but they do change while in the other modes. Eliminating the read access and associated node toggle of the RAM results in this power savings. The control method for this mode differs depending on how the RAM is specified. If coding the RAM, a coding template for the NO_CHANGE RAM from the Language Templates can be used. If using CORE Generator™ software, NO_CHANGE can be selected from the Operating Mode selection. If instantiating the RAM primitive, the WRITE_MODE attribute must be set to NO_CHANGE for each desired port. See Figure 8. Finally, if the block RAM is being used to implement a FIFO function or require error correction circuitry (ECC), the use of these dedicated functions is more power efficient than building the equivalent functionality from soft logic (e.g., LUTs and flip-flops).



WP285_08_011008

Figure 8: **CORE Generator Tool Graphical User Interface**

Digital Signal Processing

The digital signal processing (DSP) element, available in Virtex-5 FPGAs is the DSP48E slice. For high-speed designs that use several DSP48E slices to realize high throughput filters and other digital signal processing algorithms, certain design styles around the DSP48E slice can result in substantial power savings. Some examples of this are:

- Pipeline, pipeline, and more pipeline
  - Especially the multiplier register (MREG)
- Proper construction
  - Use the cascade whenever applicable
  - Use the adders whenever applicable
- Tie off the unused clock enables

Pipelining of DSP algorithms is often required to get the best performance as well as the best power characteristics from the FPGA. When using the DSP48E slice as a multiplier, it is highly recommended to use all three pipelining stages whenever possible. However, when that is not possible due to latency or other design considerations, it is most advantageous to, at minimum, use the MREG. Enabling this register breaks up many of the intermediate calculations of the multiplier over two clock cycles, which in turn saves approximately 15% of the overall DSP48E slice power.

Another important aspect of DSP design is to construct algorithms that make the most efficient use of the DSP48E slice's connectivity and construction. The cascade functionality provides a high-performance and low-power implementation of DSP

filter functions. Many filters are composed of multipliers and adder/accumulators. A traditional way to construct such circuits is by successive multiplications summed up by an adder tree structure. The DSP48E slice, however, has an efficient cascade function that saves logic and routing resources—and significant power because the general routing in the FPGA logic is not used. The additional filter portion needs to be described as an adder chain rather than an adder tree. See Figure 9.

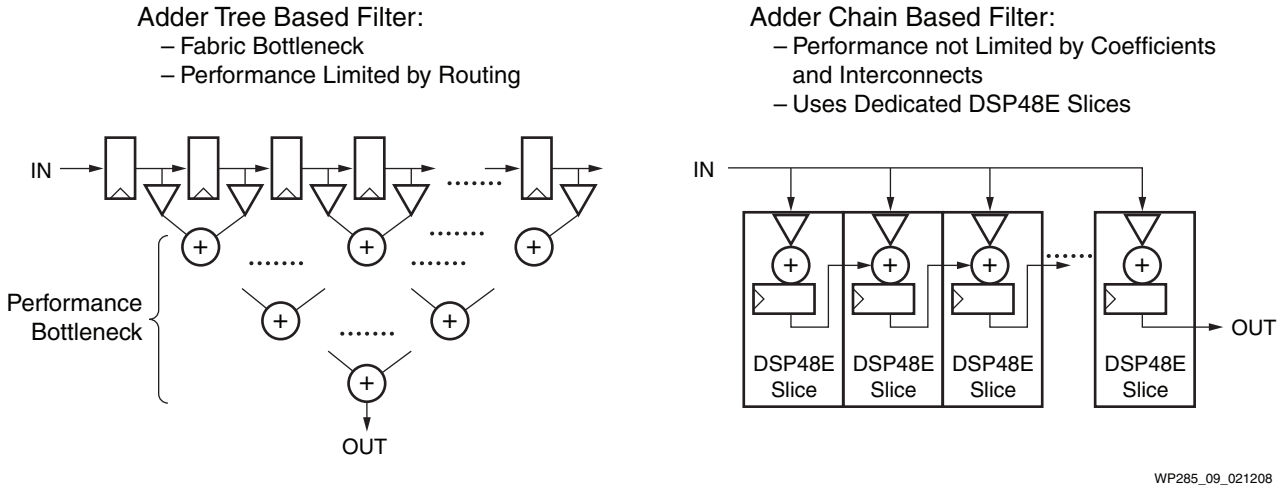


Figure 9: **DSP Implementation Comparison between Virtex-5 FPGAs and Previous Generation FPGAs**

Another simple way to save power within the DSP48E slice is to tie off unused clock enables to ground. Pipelining is still strongly recommended. But if full pipelining is not used, then grounding the unused associated clock enables of the pipeline registers disables those registers. For example, tying the AREG and BREG clock enable inputs to a logic zero when not in use results in additional power savings (with no negative side effects) of approximately a 2%. For the product register (PREG), there is a 3% power savings, and for the MREG, there is for a 4% power savings.

Use of Resets

It is a common practice to specify a global reset, many times an asynchronous global reset. Whether used or not in the design, the addition of a global reset can result in less efficient, higher power FPGA designs.

Design recommendations include:

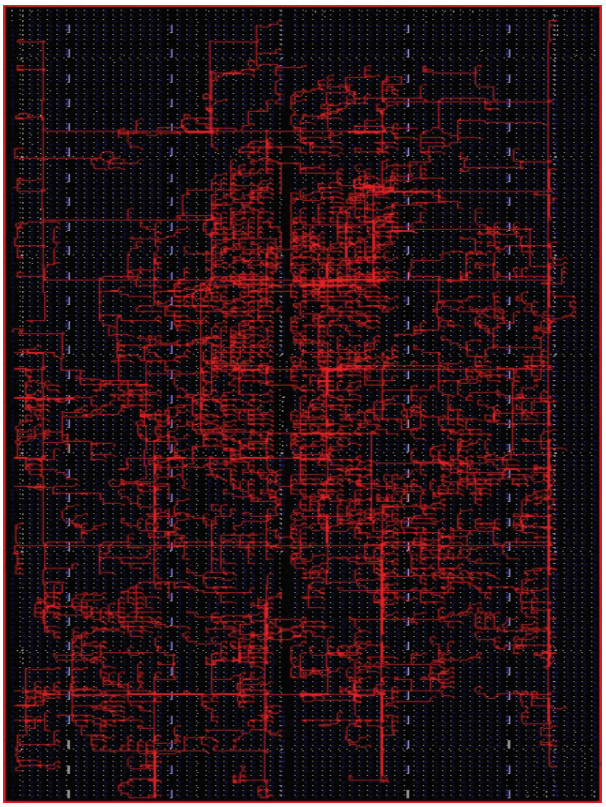
- Use resets only when necessary for the proper function of the design.
- When resets are necessary, use synchronous resets.

Many designers code a reset for any inferred register without much thought as to whether that reset is truly needed or not. Coding a reset for an FPGA is not necessary for initialization because there is already a dedicated reset for this purpose. So if global reset is coded solely for that purpose, the reset is redundant and consumes logic and routing resources that could be made available to other functions of the design. This resource usage also indirectly contributes to the overall power budget; the description of the reset, particularly an asynchronous reset, can result in the consumption of more LUTs and register resources as well as less optimal routing choices due to the reset logic utilization. Designs that incorporate a global reset versus those that do not are found to consume on average 4% more registers, 6% more LUTs, and up to 18% more routing resources. This increase is attributed in part to greater Shift Register LUT

(SRLs) inference, less need for logic and register duplication, additional use of the synchronous set and reset circuitry as a part of the datapath, and some other second order optimization effects. The additional resources consumed when a global reset is described consumes more power in the additional configuration and switching of the logic *and* in the added connectivity and node switching between them. The additional routing consumed by the global reset also consumes routing resources that otherwise could be used by the general datapaths of the design. This adds congestion, and less optimal choices are available for routing in terms of both power and performance. Therefore, when coding the design, it is highly recommended to consider whether a reset is really necessary for design operation. When a reset is used only for initialization purposes or for periodic resets, the designer should consider:

- placing resets on the input and output registers as well as in critical control logic like state machines or FIFO logic.
- refraining from using resets in general datapaths unless it is found necessary to allow proper function of the circuit as determined from simulation.

In Figure 10, the red shows the amount of routing consumed by a redundant reset that could have otherwise been available to help choose more optimal datapaths for performance and power.



WP285_10_011008

Figure 10: **Unnecessary Routing Consumption**

Clocking

Some key clocking design considerations for power include:

- Minimize use of DCMs and PLLs
- Choose the right clocking resources for the job

Each DCM or PLL used in the design adds to the overall power consumption in the FPGA. For this reason, the number of these clocking elements should be reduced when possible. Each DCM and PLL has multiple outputs to provide different frequencies, phases, and other characteristics of the clock available within the design. Using DCMs or PLLs at the module level can result in unnecessary duplication of clocking resources. Therefore, it is generally suggested that all clock management either be defined in the top-level hierarchy of the design node or in a dedicated clocking module to be supplied to the rest of the design. This prevents unnecessary duplication of the element and leads to improved power characteristics. See Figure 11.

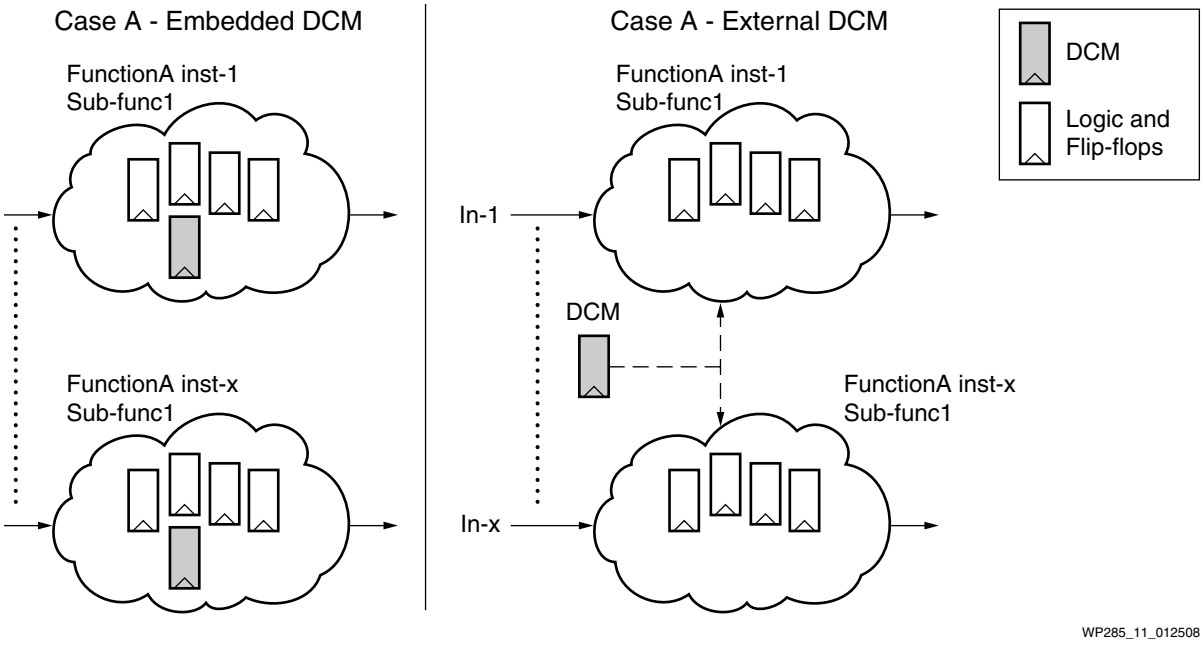


Figure 11: **Implementation Comparison between Embedded DCM and External DCM**

Another clocking-related consideration is the selection of clocking resources. The Virtex-5 FPGA has multiple types of clock buffers. The power characteristics of each clock buffer is different depending on the clock rates and what is being driven by the clock. In general, synthesis automatically infers a global buffer. However, if a regional buffer or an I/O buffer can be selected, they often show better power characteristics, especially for low fanout, low-speed clocking situations.

I/O Interfaces

The choice of I/O standards for various interfaces on the FPGA can have a large influence on power consumption. The FPGA interfaces fall into a number of categories, each having different trade-offs for power and performance.

I/O standards that consume static power should be avoided when possible. LVDS and single-ended termination standards (HSTL and SSTL) consume extra power in both the receiver and transmitter for LVDS and in the receiver for HSTL and SSTL. This is fixed DC power that is not always necessary except in the very highest performance cases. For terminated or constant-current I/O standards, increasing frequency and reducing bus width to achieve the same bandwidth at lower power should be considered (SERDES makes this easier). Also, the designer should use the lowest voltage, lowest current drive strength, and slowest slew rate that can do the job.

Interface options include:

- FPGA to ASICs: The I/O standards of these interfaces are defined by either vendor-specific needs or industry standards. For example, the SPI-4.2 interface is usually a 16-lane LVDS interface with a separate forwarded clock. To the extent that this is specified, the FPGA designer must go with that specification, so there is really not much choice with regard to optimizing the interface for power, speed, drive strength, etc.
- FPGA to FPGA: These interfaces are at the discretion of the designer. They are based on requirements that range from simultaneously switching outputs (SSO) noise, interface width, interface speed, impedance, and drive strength. While LVDS or HSTL can be used for better power characteristics, LVCMOS is the best choice. Often the Xilinx HSLVDCI I/O standard is an excellent choice for an FPGA-to-FPGA interface because it has a series Z_0 and gives a clean waveform without burning DC current. The FPGA receiver can use LVCMOS and not burn DC current like it does with LVDS, HSTL, or SSTL.
- FPGA to Memory: These interfaces are chosen to meet impedance, signal level and type, and performance based on the specifications of the memory vendors. DDR2 SDRAM memory uses SSTL_I and SSTL_II; RLDRAM II memory uses HSTL_I and HSTL_II. Depending on speed and trace impedances and distances involved, some trade-offs can be made to minimize power consumption. Other trade-offs, such as putting terminations to V_{TT} externally, reduce power compared to the HSTL_II_DCI or SSTL_II_DCI standards. However, these trade-offs can be at the expense of some signal integrity, which should be okay in all but the highest speed interfaces.

Here is a real-world example of how much power consumption can be reduced based on the I/O interface choices for the memory interfaces connected to the FPGA. This example includes seven Xilinx FPGAs, seven sets of memory (RLDRAM II memory at approximately 400 Mb/s), and some ASICs. The customer's FPGA pins, connected to the RLDRAM II memory interfaces, were using the HSTL Class II with Digitally Controlled Impedance (DCI) technology (by Xilinx) enabled for termination purposes on the DQ pins.

From an impedance and board-space point of view, the DCI high-speed, single-ended I/O standards are a good choice. But they consume more power than the non-DCI I/O standards. More detail regarding the DCI trade-off between power and performance can be found in XAPP 863, *Using Digitally Controlled Impedance: Signal Integrity vs. Power Dissipation Considerations*. Working with Xilinx, the customer determined that a much lower power termination scheme (the HSLVDCI solution by Xilinx) could be

used. Also, by setting the RLDRAMs into a specific termination mode, low-power and high-performance results could be met simultaneously. The net result was a significant 20W reduction (out of 110W) across the whole assembly, since it was implemented in each of the seven Xilinx FPGAs.

Optimizing Designs for Power Consumption through Changes in the FPGA Design Tools

Using Proper Constraints

Timing, placement, and other constraints help to guide and occasionally dictate implementation tool decisions in an attempt to get a desired result. In terms of power, constraints and pinouts can have a significant impact on the end power dissipation of a given design.

Timing constraints can have a very influential effect on the end implementation of the design. Synthesis tools attempt to build the function in a way that has the best chance of meeting the requested constraint. In the case of a tight timing constraint, the synthesis tool can attempt to widen logic functions, duplicate logic and/or registers, and use less resource sharing as well as other tactics to improve the overall timing of the circuit. Many times, these actions are at the cost of area and power. When given a lot of timing slack, the synthesis tools can attempt better optimizations for area and power, and thus for the same functionality, they yield a lower power circuit. When not given a timing constraint at all, the tools generally default to deriving the fastest circuit possible, which again can sacrifice area and power. Similarly, for the place and route algorithms, the synthesis tools are tuned to meet performance first, at the cost of power, so giving a tight timing constraint can yield worse results in terms of power. The solution is to give both the synthesis and the place and route tools complete and realistic timing constraints. The tools can then find a balance between meeting performance goals while attempting to reduce power consumption with any remaining timing slack. Balance in constraining the design is key because both over and under constraining a design can lead to less desirable results in terms of power.

Often the selection of design pinouts is finished early in the design process. And many times, the pinouts are completed by PCB engineers, not the same engineers who are building the FPGA designs. While this process can lead to more efficient board layout, it can also lead to less efficient FPGA placement, which can have an overall negative impact on FPGA performance and power. Therefore, it is essential to choose device pinouts that group together like data pins with their associated control signals so that logic grouping and minimization of routes can be realized in the end design. This generally means that the FPGA design and board layout engineer need to be in close communication to ensure that optimal pinout placement is used for both board and FPGA design purposes.

Using the Power Optimization Options

In general, the ISE™ design tools attempt to create a power-efficient design using default settings. However when trade-offs need to be made for opposing goals such as performance, area, or runtime, the tools need to know which goal is most important in order to make the proper trade-offs. For this reason, power switches exist in the tools that helps the optimization and placement and routing algorithms to select algorithms that give enhanced results for power—at the expense of longer runtimes and potentially larger implementations and less performance.

When using the Xilinx Synthesis Technology (XST) tool included with the ISE software, select the power optimization switch:

- During MAP, use the **-power** on switch
- During PAR, use the **-power** on switch

The XST synthesis tool has power optimization algorithms that can be activated either globally, per module, or per function in the design. Today, these algorithms center primarily around RAM and DSP inference where more power-efficient methods are used to construct the described functionality (sometimes at the expense of area and performance). An example if this can be seen in Figure 12:

64 x 8k x 4-bit	Area	Fmax MHz	Total Dynamic Power (mW)	Dynamic RAM Power (mW)
Speed Implementation	224 Slices 128 RAMB16	120	406.8	277.2
XST 9.2i-power	368 Slices 128 RAMB16	114	285.6	163.2
	-39.1 %	-5 %	29.8 %	41.1 %

1 x 32.5k x 36-bit	Area	Fmax MHz	Total Dynamic Power (mW)	Dynamic RAM Power (mW)
Speed Implementation	70 Slices 73 RAMB16	139	135.6	99.6
XST 9.2i-power	751 Slices 65 RAMB16	123	48	2.4
	-90.7 %	-11.5 %	64.6 %	97.6 %

WP285_12_012608

Figure 12: **Percentage of Area, Frequency, and Power Savings after Leveraging the Power Optimization Algorithms**

MAP has a similar optimization algorithm that can be enabled independently of the synthesis power attribute (although they generally are used in unison), where the design tools attempt an improved placement for reduced power. MAP works by first attempting to choose a design layout that best meets timing requirements and then does an estimated timing analysis to assess potential timing slack. If adequate timing slack is found, the placement is then modified to derive better placement for power, which in turn allows for better routing as well.

PAR has a power option that invokes the power aware router. The PAR algorithm's first goal is to meet timing. Its second goal is to reduce power on non-timing-critical nets. This stage also performs some logic optimizations that do not change timing or functionality but yield a better power profile. When both MAP and PAR are run with these options, an average power reduction of 5–10% is seen with a 15% runtime penalty and potentially a few percentages of performance degradation. Greater benefits are seen for less-timing-critical designs due to more timing slack. Designs with tight timing requirements generally see less benefit from MAP and PAR due to fewer options for design change that do not disrupt timing.

Optimizing for Area to Reduce Power

In some cases, area and power selections directly oppose one other because some power reduction methods automatically increase the amount of logic or other resources needed to implement. As a general rule, however, the fewer resources needed to implement a particular function, the less power it consumes.

This section describes some tool options that are generally effective in reducing area *and* power (in most cases) for Virtex-5 designs:

- Use synthesis options to reduce area
- Use map options to reduce area

As previously mentioned, synthesis area can be influenced by timing constraints, so it is highly suggested to create realistic timing constraints for the design. Beyond that, some synthesis options are generally advantageous for area, and many times have a shared benefit for power. One such option is resource sharing. Enabling resource sharing generally makes for more compact arithmetic functions that can lower the overall area, capacitance, and switching of that function. In XST, setting the Optimization Goal constraint to **Area** can prove to be beneficial for power. Setting the finite state machine (FSM) encoding to more compact encoding schemes like binary generally show more power efficiency for state machines than less area efficient styles like one-hot. Also, disabling safe implementations of FSM can have minor impacts as well. If the design contains asynchronous resets, selecting the switch to treat asynchronous resets as synchronous can prove to create a better design in terms of power.

The other program that can significantly affect device area and thus power is the MAP process. The **-lc switch** gives improved results in terms of power by reducing the overall number of LUTs necessary for a given design. In the current ISE release, the **-lc switch** can be used by including **-lc auto** on the command line:

```
map -lc auto [other_options] [-o <outfile[.ncd]>] <infile[.ngd]>
[<pcffile[.pcf]>]
```

Or if using Project Navigator, within the advanced properties for MAP, add **-lc auto** to the *Other Map Command Line Options*. See Figure 13.

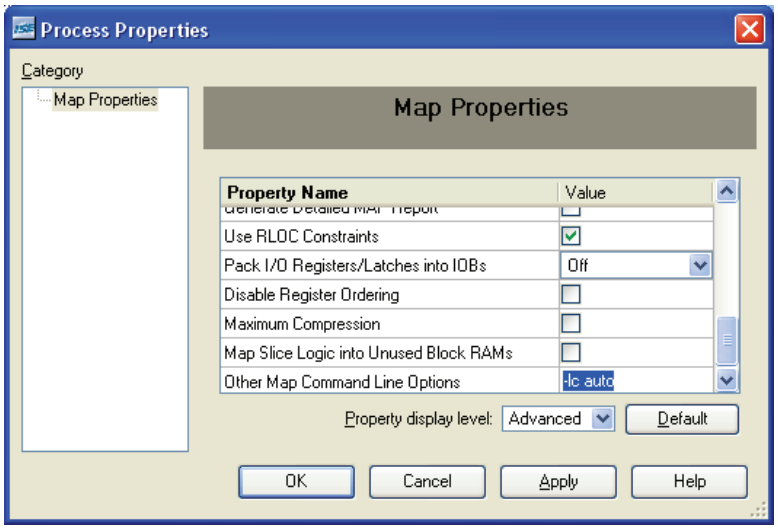


Figure 13: **Project Navigator View for Selecting Power Optimizing Features**

Using Power Analysis Tools for Power Optimization

Xilinx offers XPE and XPower Analyzer tools to understand and analyze the power dissipated in an FPGA. A proactive method to reduce power is to use these power analysis tools to understand the power trade-offs of different implementation and system-level strategies:

- Use XPE to understand the power trade-offs of different implementation styles
- Perform what-if analysis of different environmental effects on the design

The power analysis tools provide an invaluable view into the power dissipation of the FPGA. This view is useful in determining the trade-offs of different board and environmental designs on the power consumption of the FPGA. Early in the design process, the tools can:

- Determine the power differences between using LVCMOS vs. SSTL for an I/O standard
- Specify whether a state machine should be implemented in a block RAM, in the LUTs and flip-flops as a one-hot, or stay implemented with binary encoding
- Determine the better choice between LUTRAM or block RAM
- Calculate how much power can be saved by reducing the temperature by 10°C or regulate V_{CCINT} to a maximum of 1.0V

Many what-if scenarios can be reviewed with the tools to ensure proper choices in terms of power utilization. And when finished with their analysis, the tools provide information about potential power dissipation and potential power savings.

Summary

In today's high-performance system designs, power consideration is becoming increasingly important. It has a direct impact on power supply and thermal component considerations, and the device temperature is also directly tied to system reliability. It is important to know the system power budget and operating environment. Understanding where various forms of power consumption occur allows designers to adjust the FPGA environment and design characteristics to minimize power consumption and successfully meet a given power budget to maintain a reliable product. By implementing the power design tips provided in this white paper, the user can reduce FPGA power consumption to meet the design budget, thus decreasing overall system BOM cost and enhancing system reliability.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
02/14/08	1.0	Initial Xilinx release.

Notice of Disclaimer