AN 531: Reducing Power with Hardware Accelerators

May 2008, ver. 1.0 Application Note 531

Introduction

Reducing power consumption in embedded products that use FPGAs is increasingly important, particularly for battery-powered applications or to reduce heat or cost. You can use parallel algorithms to exploit the parallel architecture of FPGA devices to accomplish more work per clock cycle, allowing you to lower the clock frequency. High-level development tools such as SOPC Builder and the Nios® II C-to-Hardware Acceleration Compiler (C2H) can help you use the power-saving potential of the FPGA hardware by easily adding hardware accelerators and lowering clock frequencies.

Basics of Power

There are two essential components to the power consumption of a CMOS device: static power and dynamic power.

Static Power Versus Dynamic Power

Static power, sometimes also called leakage current or standby current, is the power consumed as a result of current that leaks through a device's transistors while they are static. Because static power is a relatively constant characteristic of a particular device, there is little you can do in an FPGA design to reduce the static power consumption, except to use a smaller FPGA device.

Dynamic power is the power consumed as a result of the current that flows through transistors as they switch states. In digital CMOS circuits, this transition time is when most current flows through the transistors; therefore, this is when the transistors consume the most power. In a synchronous digital CMOS circuit, most power is consumed during clock transitions, so higher clock frequencies usually translate to higher power consumption. One key to reducing power consumption is clocking the logic as infrequently as possible.

Because dynamic power is highly dependent on the clock frequency, you can potentially reduce the amount of dynamic power that an FPGA design consumes by parallelizing some of the application's tasks so that more work is done per clock cycle, thereby lowering the clock rate. The total power is only reduced if the power saved by reducing the clock rate is greater than the power consumed by the additional logic required to parallelize the task.

For more information about power dissipation refer to Altera's Power Management Resource Center web page.

FPGA Core Power Versus I/O Power

There is also a distinction between the power consumed by the FPGA core and the power consumed by the I/O pins. The power consumed by I/O pins is not only dependent on clock frequency, but also the load that those pins are driving. I/O power consumption is beyond the scope of this document. This document only addresses FPGA core power consumption.

For information about I/O power consumption refer to the *I/O Power Guidelines* section in the *Power Optimization* chapter in volume 2 of the *Quartus II Handbook*.

Using Hardware Accelerators to Reduce Power

A standard perception in embedded system design is that the more hardware added to a system, the more power the system consumes. While the volume of hardware certainly has a relationship to power consumption, it is certainly not the strongest relationship to power. System clock frequency has a much stronger relationship to power consumption than the amount of logic. What the standard perception fails to consider is that by adding logic, you can potentially run parts of the system at a slower clock frequency, which reduces power.

Trading Performance for Power Savings

In programmable logic, you can use hardware accelerators to increase the performance of computational tasks by using concurrency to complete more work per clock cycle. By splitting a task into independent parts executed in parallel, you reduce the total number of clock cycles required to complete the task. If you maintain a constant clock frequency, less time is required to finish the task. This technique allows more tasks to be completed in a given period of time. Historically, hardware accelerators are used to increase high-end performance at a given clock frequency.

However, performance is not the only area where hardware accelerators can help. You can also use hardware accelerators to reduce power. If adding hardware accelerators to your system increases performance at the same clock frequency, adding hardware accelerators also maintains the original performance at a slower clock frequency. Considering the strong relationship between clock frequency and power consumption, you can trade extra performance for power savings.

The reason this technique works so well is that, in an FPGA, clock frequency typically has a far greater effect on power consumption than the amount of logic it implements. Any unused logic resources in your FPGA can potentially be used to help reduce power.

Creating Hardware Accelerators

You can create hardware accelerators in a variety of ways. Some designers create hardware accelerators directly in a hardware description language (HDL) such as VHDL or Verilog HDL. This method results in finer-grained control over the accelerator logic and related timing. However, building an efficient accelerator in HDL requires detailed knowledge of logic design techniques and a solid understanding of the system's bus architecture.

Another faster and more efficient method, that requires far less logic design knowledge, is using an accelerator generation tool such as the C2H Compiler. The C2H Compiler takes individual C functions that you specify and compiles them into hardware accelerator modules that are implemented in the FPGA. The C2H Compiler can quickly and efficiently generate hardware accelerators for your system, which allows you to experiment with accelerating different areas of your design to observe the effects on performance and power. For details and documentation regarding the C2H Compiler, visit the C2H webpage.

Choosing Functions to Accelerate

The amount of power savings you can achieve using hardware accelerators depends on a number of factors, including the number of accelerators you add to the system and the efficiency of those accelerators. Therefore, it is important to carefully consider the architecture of your system and its accelerators.

Some functions and algorithms are less suitable for hardware acceleration. Attempting to implement accelerators for these such functions may not yield much improvement in power consumption, or may even increase power. Algorithms that are sequential in nature, so that each operation depends on the result of a prior operation, do not translate well to an efficient hardware accelerator. For example, functions that traverse linked lists in memory are generally inefficient when implemented as hardware accelerators because each memory read depends on the result of the memory read preceding it, which limits the amount of parallelism the algorithm can achieve.

Conversely, algorithms that can process data in parallel are ideal candidates for hardware acceleration. Video processing algorithms, such as the Mandelbrot example presented in this document, can often be

implemented in a hardware accelerator with great efficiency because a large number of independent pixels or blocks of video data can be processed in parallel.

For more information on choosing efficient algorithms to accelerate with C2H, refer to the C-to-Hardware Acceleration section of the Nios II Processor web page.

Implementing Multiple Hardware Accelerators

Adding just a single hardware accelerator can often produce significant power savings over a processor alone, but adding multiple accelerators can sometimes help reduce power even further.

When implementing multiple accelerators, it is even more important to consider the architecture of the whole system because adding large amounts of hardware inefficiently can reduce the system's power efficiency rather than increasing it. For example, if multiple accelerators are constantly arbitrating for the same system resource, the efficiency benefit of multiple accelerators can be negated by the over-utilized shared resource.

In the five-accelerator Mandelbrot example presented in this document, five accelerators share the work of processing each video frame. Each accelerator is identical to the other four and the task of each individual accelerator is to process a single line of video from left to right, calculating the result of the Mandelbrot algorithm for each pixel in that line.

To ensure that the five accelerators work efficiently in unison, the algorithm uses a simple line-arbitration scheme. To process an entire frame of video, the Nios II processor acquires an available empty frame buffer in memory, then individually starts each accelerator. The accelerators each query a hardware mutex component to acquire a single video line to process. The hardware mutex ensures that no two accelerators ever begin processing the same line. When an accelerator finishes processing a line, it gets the next available line from the hardware mutex, and begins processing again. The accelerators continue this sequence of acquiring and processing individual lines of output until an entire frame has been processed. Once every line in the frame has been processed, the accelerators signal to the Nios II processor that the frame is complete.

Understanding Other Design Requirements

The primary goal of using hardware accelerators to reduce power is to lower the clock frequency of the FPGA logic while maintaining acceptable performance levels. However, some applications require rapid response time to asynchronous events such as interrupts in addition to a particular level of data throughput.

Unfortunately, by lowering the clock frequency of the entire system, you also lower the clock frequency of the processor, effectively slowing its response time to such events. Therefore, if your application requires a fast CPU response time to asynchronous events, you may not have the option of lowering the clock frequency of the processor.

However, even when your design requires rapid response time, you can still attain significant power savings by adding hardware accelerators. You can use two separate clock domains: a slower domain for the hardware accelerators and a faster domain for the processor. By adding hardware accelerators running at a very low clock frequency, you can relieve the processor of heavy processing work that consumes more power, reducing overall system power consumption without having to reduce the processor clock frequency. You can implement the separate clock domains with clock-crossing bridges in SOPC Builder.

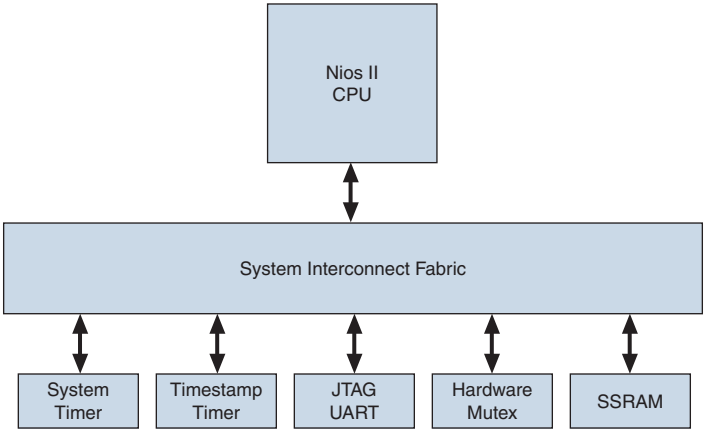
For more information regarding multiple clock domains in SOPC Builder, refer to *Avalon Memory-Mapped Design Optimizations* in the *Embedded Design Handbook* and the *Avalon Memory-Mapped Bridges* chapter in volume 4 of the *Quartus II Handbook*.

Example Designs

This section demonstrates the power savings that hardware accelerators can achieve with three example designs that run on the Cyclone® III Starter Kit. Each example design performs the same task, a Mandelbrot fractal calculation. The difference between the designs is the number of hardware accelerators that they contain.

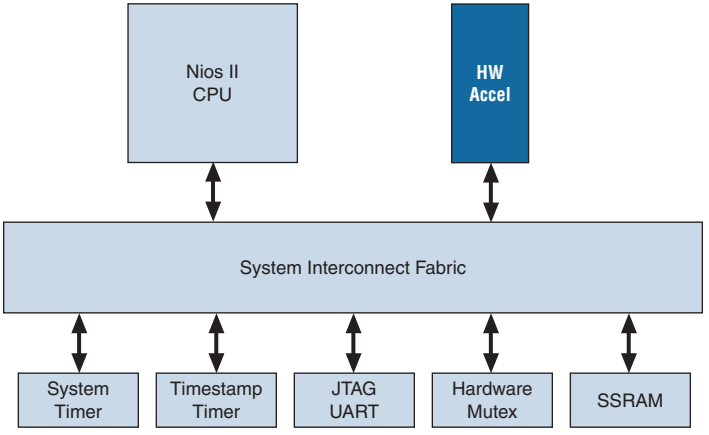
The first example contains no hardware accelerators. It includes a single Nios II CPU that performs the entire Mandelbrot calculation itself. Figure 1 shows a block diagram for the example with no hardware accelerators.

Figure 1. Simplified Block Diagram of a System with No Accelerators



The second example contains one hardware accelerator that performs the Mandelbrot calculation. The accelerator increases the performance of the design at a given clock frequency, allowing the clock frequency to be lowered, which reduces power consumption. Figure 2 shows a block diagram of the one-accelerator system.

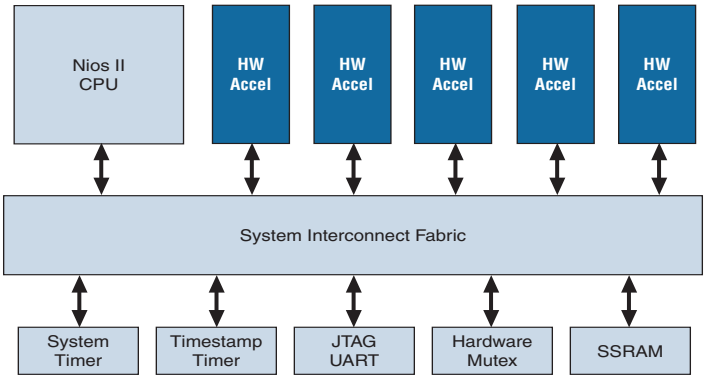
Figure 2. Simplified Block Diagram of a One-Accelerator System



The third example contains five hardware accelerators which perform the Mandelbrot calculation. The five accelerators work in parallel to further increase the performance of the design at a given clock frequency,

allowing the clock frequency to be lowered even more than in the one-accelerator system, which further reduces power consumption. Figure 3 shows a block diagram of the five-accelerator system.

Figure 3. Simplified Block Diagram of a Five-Accelerator System



Running the Examples

You can download the files to run the Nios II Low Power Design Example from the Altera website. A hyperlink to the design files appears next to this application note on the Application Notes web page. For information on cabling and powering up the Cyclone III FPGA Starter Kit used in these examples, refer to the *Cyclone III FPGA Starter Kit User Guide*.

1. Copy the **power.zip** file to a working directory on your computer and extract the files in the **.zip** file. This document refers to the newly-created directory as the **/<*projects*>/power** directory

2. Choose **Programs > Altera > Nios II EDS** <*version*> **Command Shell** (Windows Start menu) to run a Nios II command shell.

3. Change to the directory: **/<*projects*>/power/c3_power_c2h_0_accel/software_examples/app/accel_0_test**

4. To create and build the software project, type the following command:

```
./create-this-app ↵
```

5. To configure the Cyclone III FPGA on your board, type the following command:

```
nios2-configure-sof ../../../c3_power_proj.sof ↵
```

6. To download and run the software on the Nios II processor, type the following command:

```
make download-elf ↵
```

7. To open a terminal session to capture messages from the Nios II processor, type the following command:

```
nios2-terminal ↵
```

   The Nios II processor prints its clock frequency information and Mandelbrot performance measurements to the terminal session.

8. Type `Ctrl+C` to stop processing.

   Processing at 1 MHz with no hardware acceleration may take longer than 7 minutes.

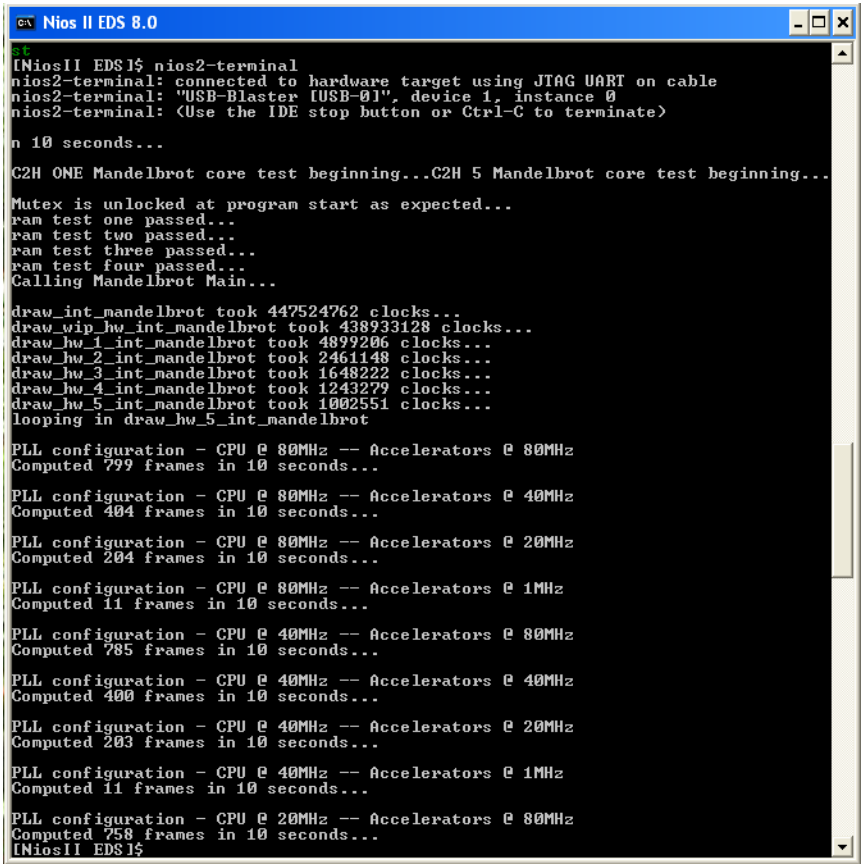
9. To run the one accelerator and five accelerator designs, repeat steps 3–6, with the following changes.

   a. For the example with one accelerator, replace the **c3_power_c2h_0_accel/software_examples/app/accel_0_test** subdirectory in step 3 with: **c3_power_c2h_1_accel/software_examples/app/accel_1_test**.

   b. For the example with five accelerators, replace the **c3_power_c2h_0_accel/software_examples/app/accel_0_test** subdirectory in step 3 with: **c3_power_c2h_5_accel/software_examples/app/accel_5_test**.

Figure 4 shows part of the output from the session with five accelerators. This output shows that reducing the clock frequency from 80 MHz to 20 MHz while maintaining a 80 MHz frequency for the accelerators has very little effect on performance.

Figure 4. Results for Example with Five Accelerators



Measuring Power

You can measure the total power consumption of the FPGA core running each of the examples by measuring the voltage across a current sensing resistor on the board.

Refer to the *Cyclone III FPGA Starter Kit User Guide* for full instructions on accurately measuring and calculating FPGA core power consumption.

Reviewing Results

These example designs measure and compare power versus performance for the following four configurations:

- Nios II alone running at 80 MHz
- Nios II with 1 hardware accelerator running at 40 MHz
- Nios II with 5 hardware accelerators running at 20 MHz
- Nios II running at 80 MHz with 5 hardware accelerators running at 1 MHz

The results show that for the Mandelbrot algorithm example, adding 5 hardware accelerators and reducing the system clock frequency by a factor of 4, reduced the power consumption from 132 mW to 84 mW, a reduction of 63.6%. At the same time, system performance increased from 0.184 frames per second to 20.1 frames per second, an increase of over 100X. Combining the power savings and the performance increase, the total power efficiency of the system increased from 717 mWs/frame (miliwatt-seconds per frame) to 4.1 mWs/frame, an efficiency improvement of 175X.

Even in the case where the CPU frequency is kept constant at 80 MHz, the addition of 5 hardware accelerators operating at 1 MHz increases the power efficiency of the system from 717 mWs/frame to 54mWs/frame, a factor of 13.2, without having to reduce the clock frequency of the Nios II processor.

Table 1 shows the complete measurement results of the four sample configurations.

Table 1. Measured Dynamic Power (1)

System Configuration	Nios II	Nios II + 1 Accelerator	Nios II + 5 Accelerators	Nios II + 5 Accelerators
CPU Clock Frequency	80 MHz	40 MHz	20 MHz	80 MHz
HW Accelerator Clock Frequency	80 MHz	40 MHz	20 MHz	1 MHz
Total Dynamic Power*	132 mW	72 mW	84 mW	60 mW
Mandelbrot Calculation Performance	0.184 Frames/s	8.2 Frames/s	20.1 Frames/s	1.1 Frames/s
System Efficiency	717 mWs/Frame	8.7 mWs/Frame	4.1 mWs/Frame	54 mWs/Frame

Note to Table 1:
(1) Dynamic Power is defined as the total power less the static power (power measured with no clock applied).

Other Documents

The following documents provide additional information about designing for low power using Cyclone III, Nios II, and SOPC Builder.

- *Achieving Low Power in 65-nm Cyclone III FPGAs*
- Arrow Low Power Reference Platform featuring Cyclone III
- *Cyclone III Device Handbook*
- Nios II C2H Compiler User Guide

Referenced Documents

This application note references the following documents:

- *Avalon Memory-Mapped Bridges* chapter in volume 4 of the *Quartus II Handbook*.
- *Avalon Memory-Mapped Design Optimizations* in the *Embedded Design Handbook*
- *Cyclone III FPGA Starter Kit User Guide*.

Document Revision History

Table 2 shows the revision history for this application note.

Table 2. Document Revision History

Date	Changes Made	Summary of Changes
May 2008 v1.0	Initial release.	—

101 Innovation Drive
San Jose, CA 95134
www.altera.com
Technical Support:
www.altera.com/support/