White Paper

Adding Hardware Accelerators to Reduce Power in Embedded Systems

Not all functions are equally suited to trading circuits for frequency. Functions that operate in parallel run much faster when hardware is available to execute several steps simultaneously, which translates into greater performance for a given clock speed, but also into lower clock speed for a given performance level. Thus, the addition of hardware to a chip design can lower power demands while maintaining performance.

Introduction

The rule of thumb in embedded system design has been that adding hardware increases power demands. The careful use of hardware accelerators, however, inverts the rule: adding hardware can reduce power. By analyzing algorithms and implementing appropriate accelerators in programmable logic, developers can increase a design's performance while reducing power consumption in an embedded computing system. Test results show that accelerators extend trade-off options from as much as 200-fold performance improvement for the same power to the same performance with a 90% power reduction.

Programmable logic has, somewhat undeservedly, maintained a reputation from its early history as being a power-hungry approach to logic design. The rules of thumb have been that power consumption in an integrated circuit is roughly proportional to the chip's area for a given process technology, and a design implemented in programmable logic tends to be larger than if implemented in hard-wired logic. But these two factors, although suggestive, are misleading.

Far more significant than area-related power dependency is the frequency-related power dependency of an integrated circuit. Because CMOS circuits draw most of their current when transistors switch states, the frequency at which a circuit operates has a much greater impact on power consumption than simple chip size. The higher the frequency, the greater the power demand. This opens the possibility that designers can reduce chip power consumption by adding circuitry, if the result of adding hardware is a significant reduction in clock speed.

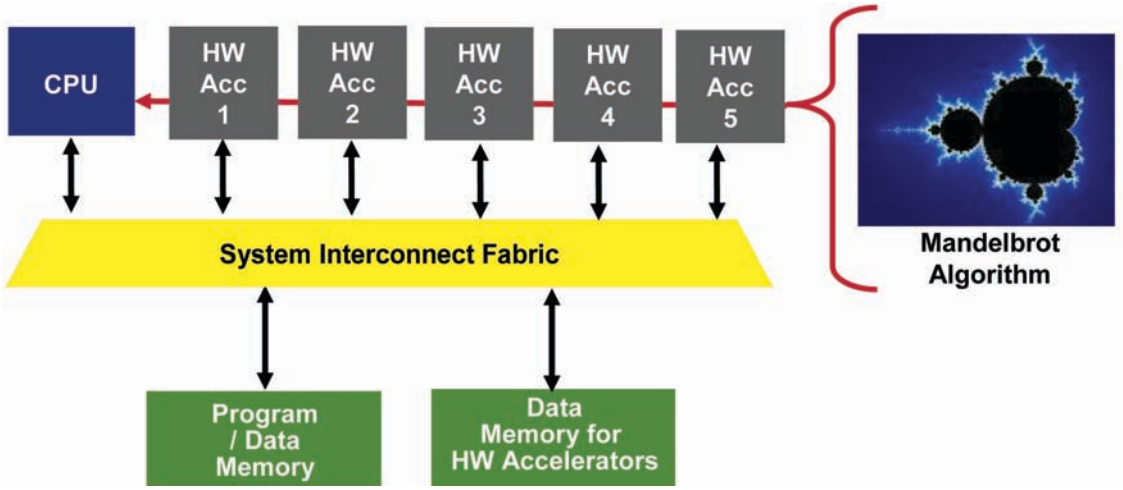
For years, embedded processors have relied on custom hardware functions to accelerate common algorithms such as graphics or signal processing to accomplish more work per clock cycle. While this approach increases system performance, it does not reduce the system clock or dynamic power consumption. If hardware can be applied to accelerate software algorithms AND reduce the clock frequency, power can be saved while meeting system performance.

Not all functions are equally well suited to trading circuits for frequency, however. Sequential processes, where one step must be completed before the next begins, typically see little benefit from added circuitry. Functions that can operate in parallel, on the other hand, can run much faster when hardware is available to execute several steps simultaneously. This translates into greater performance for a given clock speed, but also into a lower clock speed for a given performance level. Thus, the addition of hardware to a chip design can lower power demands while maintaining performance.

Mandelbrot Example

To demonstrate the types of power savings that designers can achieve, a lost cost FPGA-based design example has been developed using a 50-MHz Altera EP3C25F324 with 25K logic elements (LEs), 66 M9K memory blocks (0.6 Mbits), 16 18x18 multiplier blocks, and four PLLs. The design executed the Mandelbrot algorithm for calculating fractals, using as its baseline the Altera® Nios® II embedded processor. Despite the relatively small size of the FPGA used, the microprocessor only occupied a portion of the FPGA's resources. This left room for implementing additional hardware to accelerate the algorithm's execution (as shown in Figure 1).

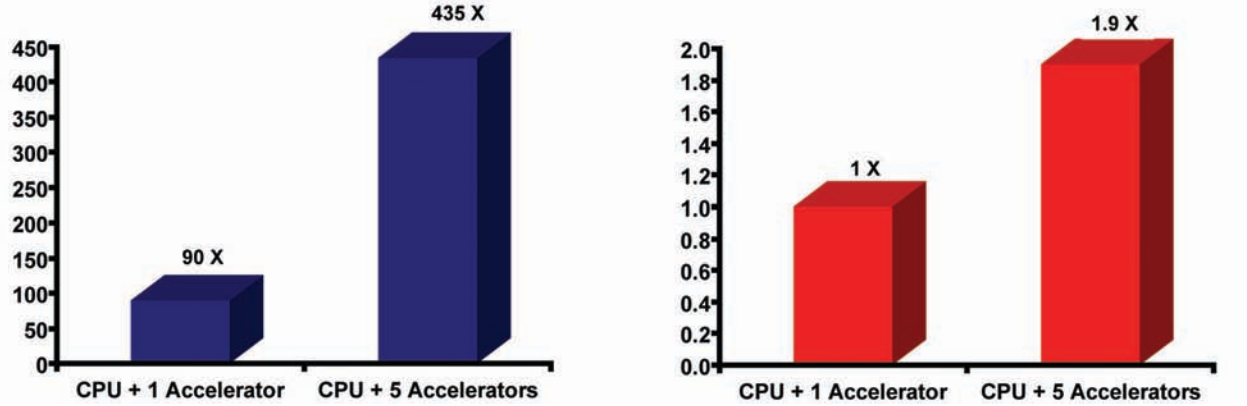
Figure 1. Typical Block Diagram of a Processor System



Testing evaluated the processor alone and the processor with as many as five hardware accelerators. Larger members of Altera's Cyclone® III and Stratix® III product families, which have many times the test device's capacity, would provide even more extensive trade-off opportunities.

Baseline tests showed that the Nios II processor operating alone required 435 million clock cycles to complete the calculations for one Mandelbrot frame. Adding a single hardware accelerator brought the execution requirement down to 4.9 million clock cycles—nearly a 90-fold improvement in performance (Figure 2, left)—without a measurable increase in power demand. Adding four more hardware accelerators yielded incremental improvements as much as 435 times the performance of the processor alone. The power consumed by the additional accelerators was only 90% greater than the CPU alone (Figure 2, right).

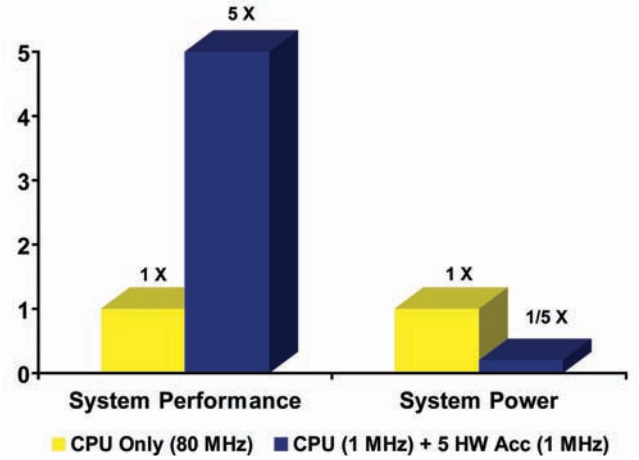
Figure 2. Effect of Adding Hardware Accelerators on System Performance (left) and Power Consumption (right)



Reducing System Clock Frequency

A 435X performance increase creates abundant computational headroom that can now be traded for lower power. One way to approach this reduction would be to slow the clock for the entire design. As Figure 3 shows, even with a single accelerator, the entire design can be run at 1 MHz and still achieve greater performance than the CPU alone running at 80 MHz.

Figure 3. Effects of Reducing System Clock Frequency



Meanwhile, the power savings are significant. The design with CPU and one accelerator running at 1 MHz used only 12 mW compared to 132 mW for the CPU alone running at 80 MHz while still achieving nearly twice the performance. If the 5-accelerator design is considered, power is reduced to less than one-fifth the CPU alone while the performance is increased by over five times.

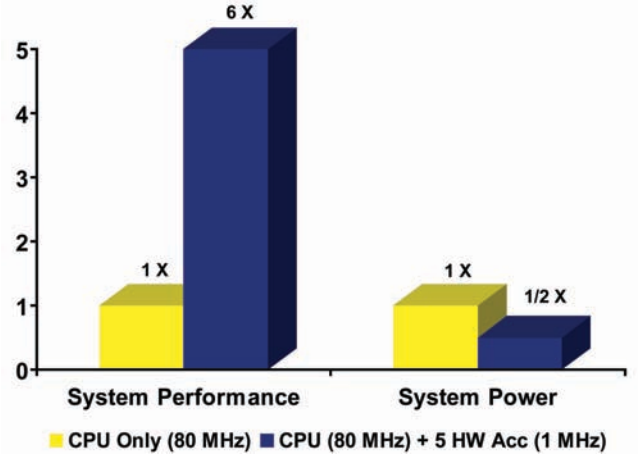
Reducing Accelerator Clock Frequency

In many applications, however, acceleration hardware is only effective for speeding part of the algorithm. In such cases, slowing the clock everywhere in the design might adversely impact performance in other functions. A more likely scenario is that the application software requires the processor to run at the higher clock frequency. In this case, power reduction may still be achieved by reducing the accelerator clock frequency.

Developers can evaluate the effect of different clocking speeds for various hardware blocks on performance and power. The FPGAs used in this design example allow multiple clocking domains, so the CPU and its acceleration hardware can each operate at their optimum speed. By adjusting domain clock speeds independently, developers can readily determine the minimum power required while still achieving the desired performance.

Consider the case where an embedded designer wants the processor to execute code at 80 MHz while off-loading the heavy computational algorithms to hardware running at a lower clock frequency. In the test case the embedded processor running application code at 80 MHz with five hardware accelerators running at 1 MHz increased system performance by six times while still reducing system power by 55% (Figure 4).

Figure 4. Effects of Reducing Accelerator Clock Frequency

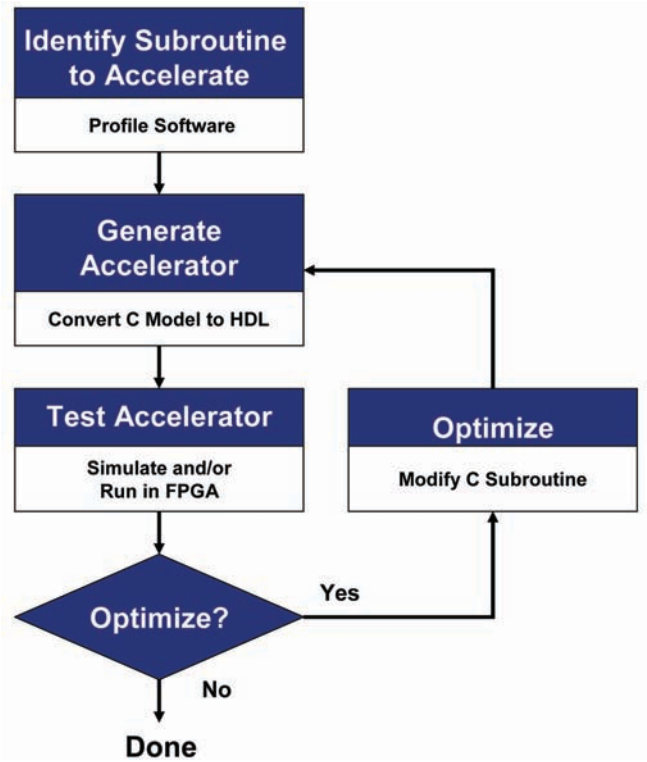


Developing Hardware Accelerators

The first step to take is examining subroutines to look for computational or state-machine algorithms. These are the types most likely to benefit from acceleration. Once developers have identified candidate subroutines, standard software profiling tools will provide the details needed to identify acceleration opportunities. Loops that take a long time to execute are good acceleration opportunities.

Creating hardware accelerators, while a familiar task for hardware developers, may pose a daunting task for software developers modeling algorithms in C. The ability to translate ANSI C algorithms into the corresponding logic is a key element in creating effective acceleration hardware (Figure 5).

Figure 5. Hardware Accelerator Design Flow



The application example was enabled using an advanced tool suite, including the C to the Hardware Acceleration Compiler (C2H) for the embedded processor and the SOPC Builder system development tool, both found in Altera's Quartus® II design software. The tools automated the conversion of the embedded processor C source code to a hardware accelerator implemented in HDL.

The accelerator with the process system was also integrated. Once the HDL is generated for the system, evaluation can take place using HDL simulation tools, or it can be run directly in FPGA hardware. Either way, the results will likely be evaluated; the code modified and iterated until the desired performance and power for the system is achieved.

Taking Advantage of Parallelism

In addition to creating accelerator hardware designs for key algorithm steps, developers should consider using multiple copies of such accelerators. Multiple copies are relatively easy to implement in a design once the HDL code is available, and the time required to make the evaluation is often worthwhile. When the algorithm is highly parallel, as in the Mandelbrot example, the parallel accelerators can amplify the opportunities for power savings, as the example demonstrated.

Conclusion

By using hardware accelerators, designers can put to rest the rule of thumb that more circuitry means more power. They also open the possibilities to a wide range of design exploration in terms of power and performance trade-offs and free designers to apply supposedly power-hungry FPGAs to a host of new applications where small form factor and battery-powered operation are essential.

Further Information

- *AN 351: Reducing Power with Hardware Accelerators*: www.altera.com/literature/an/an531.pdf
- Nios II Low Power Design Example: www.altera.com/literature/an/power.zip

Acknowledgements

- Rodney Frazer, Embedded Specialist FAE, Sales, Altera Corporation

101 Innovation Drive
San Jose, CA 95134
www.altera.com