

# How to Minimize Energy Consumption While Maximizing ASIC and SOC Performance

*Power has become a first-order concern for ASIC and SOC designers right next to performance and area, whether the design is for portable mobile devices, for networking boxes, or for any other application. Optimizing a design for energy at an application and system level has the potential to cut processor and local-memory energy requirements by as much as half in many cases through intelligent design trade-offs. The amount of power savings made at the early architectural level far outweighs any potential power savings that might be made later at the RTL or physical design levels.*

There are several EDA design methodologies for reducing ASIC and SOC operating power and energy consumption including clock gating, voltage and frequency reduction, gate sizing and logic optimization, leakage reduction techniques, and low-power libraries and technology processes. Unfortunately, these low-power design methodologies can take months to implement and they still may not reduce power and energy consumption as much as system-level architectural decisions. Architectural decisions are made before any RTL code has been written and it is at that point in the design cycle when the design team has the greatest ability to reduce power and energy consumption.

A lot of emphasis has been placed on guiding ASIC and SOC designers towards *performance-* and *area-optimized* architectures with respect to architectural choices such as memory sub-system design (banked memories versus a single large memory), interconnect (single bus versus a hierarchy of buses versus point-to-point interconnects), caches, etc. However, little has been done to guide designers towards the development of *energy-efficient* system architectures for ASICs and SOCs.

The Xenergy design tool is the industry's first EDA tool to provide a realistic and practical way for hardware designers to quickly estimate the overall energy impact of different processor configurations and extensions. This tool can also help software developers optimize energy-driven application code tuning by running the code on a simulation of the overall processor-plus-memory subsystem. While most software-development tool chains have focused on guiding application-code development to

improve performance and code size, Tensilica's Xenergy energy-estimation tool guides designers towards more energy-efficient processor-and-memory sub-system configurations.

Designers can use the Xenergy energy estimator to execute a software application binary on one of Tensilica's Xtensa processor core configuration or a Diamond Standard processor core to get a quick and early estimate of the power and energy consumed by the processor, caches, and local (tightly coupled) memories. The designer can then modify the processor configuration, add instruction extensions, register files, application-specific execution units, or simply tune the application code, with the explicit goal of reducing overall processor and memory energy requirements.

### **Focus on total energy consumption**

A focus on total energy consumption is key to energy-efficient ASIC and SOC design. Too often, system designers fixate on a static milliwatts-per-megahertz (mw/MHz) power number, ignoring the actual total energy consumption per unit of workload. This mistake occurs because there's an implicit assumption that roughly equal amounts of work are done each clock period by all processors. This assumption may hold true for general-purpose, 32-bit RISC processors, but it's not a good assumption for task-specific processors such as Tensilica's dataplane processing unit (DPU), which can be customized for specific on-chip tasks.

For example, an ASIC or SOC designer can add custom processor instructions that increase the amount of silicon consumed by the processor core. On average, a larger processor consumes more power per clock cycle. Making the processor larger generally will increase its mW/MHz number. However, if those custom instructions dramatically lower the total number of clock cycles required to execute a specific functional workload, then the overall energy consumed for that work (power-per-cycle multiplied by total cycle time) can be reduced. For example, consider a processor configuration that adds task-specific instructions resulting in a 20% increase in the power consumed by the processor per clock cycle. If that mW/MHz increase is offset by a 3X speed up in application execution, the total energy consumed to perform the task is actually reduced by 60%.

### **Estimating energy consumption cycle-by-cycle**

The Xenergy energy estimator computes power consumption per cycle for each instruction of an Xtensa or Diamond Standard processor. For each custom instruction created using Tensilica's powerful TIE (Tensilica Instruction Extension) language, the Xenergy estimator uses empirical, statistical models to calculate the energy consumed by a memory access (read and write) and the energy consumed by the execution of each instruction.

The Xenergy estimator then simulates the application on a cycle-accurate instruction set simulator (ISS), which produces detailed profiling information about each instruction executed and each memory access. Based on this profiling information, the processor configuration, the TIE instructions, and the process technology information, Xenergy uses its statistical models to estimate the dynamic power, leakage power, and total energy dissipated by the processor, the associated caches, and the local (tightly coupled) instruction and data memories attached to the processor.

The Xenergy estimation tool builds on Tensilica’s existing energy-estimation tools. Xtensa processor configuration tools already provide ASIC and SOC designers with dynamic estimates of the area, maximum clock rate, and power consumption of a processor core in real time as the processor is configured. The Xenergy estimator takes these estimates one step further by giving designers an estimate of the energy dissipated by the processor and its local memory subsystem while running a particular application.

The Xenergy tool is best used iteratively, first as the designers are selecting configuration options and adding new task-specific instruction customizations, and then by the software application developer during application tuning. In reality, the hardware/software co-design process itself is iterative. Before the Xenergy estimator became available, hardware and software developers had only performance and area analysis tools to guide them through the hardware/software tuning process. The Xenergy estimator now provides designers with early energy guidance as well.

**Impact on Processor Design**

The total energy required to complete a task (power dissipated over the time taken for the task to complete) can be dramatically reduced by customizing an Xtensa processor, as shown in Table 1 below.

Configuration		Dot Product	AES	Viterbi	FFT
Baseline Xtensa Processor	K Cycles	12	283	280	326
	Energy (μJ)	3.3	61.1	65.7	56.6
Customized Xtensa Processor	K Cycles	5.9	2.8	7.6	13.8
	Energy (μJ)	1.6	0.7	2.0	2.5
Energy Improvement		2x	82x	33x	22x

**Table 1: Improvement in energy consumption using customized instructions**

The numbers in Table 1 assume that there are no changes in the algorithm software (except for the use of C intrinsics to invoke custom processor instructions) and also assumes identical memory-cache sizes in the baseline and optimized processors.

Because it runs quickly, designers can use the Xenergy estimator to measure the energy effects of customization while developing custom instructions for an Xtensa processor. Designers can immediately see the changes in total energy consumption

when they select configuration options (multipliers, DSP engines, a floating point unit, and many additional configuration choices) and when they add custom instructions. In addition, they can see the effect of different interface and memory-subsystem options.

The Xenergy estimator's ability to model custom instruction extensions is critical to designers using the Xtensa processor as an alternative to developing hand-coded RTL accelerator blocks for the data plane of their ASIC or SOC. Some designers create a significant amount of new, custom instructions, which produce hardware structures within the processor that resemble the hardware data-path blocks in custom RTL. Being able to get an early estimate of the energy impact of these custom instructions is therefore just as important to most designers as the more familiar area-estimation and performance-profiling tools.

The inclusion of memory power consumption is another important Xenergy feature. Consider a scenario where processor customizations are used to create additional state registers and register files within an Xtensa processor core. These custom extensions are not added to appreciably improve execution performance but are instead aimed at significantly decreasing the number of accesses to local memory, which will decrease overall energy consumption. The Xenergy estimator will clearly show this energy decrease, making it easy for designers to weigh area, performance and power tradeoffs early in the system's design.

### **Impact on Software Design**

Similarly, a software programmer who is developing application code traditionally would tune the application for either performance or code size. The Xenergy estimator now provides a tool to help developers tune the application to reduce processor and memory energy consumption. For example, restructuring the application's data structures to reduce memory accesses by exploiting temporal locality of the data will reduce energy consumption. Intuitively, this change should not only reduce energy, it should result in better application code performance. Tensilica's standard software-profiling tools will show that the application performance improves and the Xenergy estimator will demonstrate that this code tuning does indeed reduce energy consumption.

### **Start with a Low Power Architecture**

The base Xtensa instruction set architecture (ISA), common to both the Xtensa customizable processor cores and the Diamond Standard processor cores, provides the industry's lowest power and highest performance when compared to older, fixed-ISA processor architectures. For example, Table 2 shows that a high-performance version of the Xtensa LX2 processor core uses less than half the die area and consumes half of the power-per-MHz of the equivalent ARM 1136J-S. Note: This comparison is not between the base Xtensa LX2 processor and the ARM 1136J-S. Rather, it's a 3-way static superscalar version of the Xtensa LX2 processor core configured as a high-performance, general-purpose CPU, equivalent to the ARM

1136J-S. Performance analysis on EEMBC benchmarks for this Xtensa configuration has shown it to be an average of 2.5x higher performance and more than 5x better performance/mW than the ARM11 processor core.

Processor	Equivalent Frequency (0.13 $\mu$ G worst case)	Power – mW per MHz (0.13 $\mu$ G)	Dhrystone MIPS/mW
ARM 1136J-S	333 MHz (single issue)	0.60	1.98
Xtensa LX2 3-way FLIX performance configuration	700 equivalent MHz (three issue)	0.170	10.4

**Table 2: Power and performance for Xtensa and ARM11 processor cores**

### Built-in low-power processor features

Tensilica has built several energy-saving features into the Xtensa processor architecture to serve as an excellent base for low-power ASIC and SOC designs. These features include power-down modes that lower overall system power, including power-down modes for local-memory accesses and power-down of the trace port control and on-chip debug modules. The architecture also implements automatic fine-grain clock gating for every functional element within the processor including any function units added through custom instructions.

The power consumption for a minimum Xtensa configuration is:

- 15 $\mu$ W/MHz in 45nm GS process, speed-optimized netlist, typical operating conditions, mac clock speed 1.1GHz.

### Conclusion

Tensilica's Xenergy tool estimates energy for processor/memory subsystems based on Xtensa or Diamond Standard processor cores and uses the actual application code that will run on that subsystem to generate the estimate. Energy estimates take minutes to generate versus hours or days for RTL power analysis, which can greatly speed the design cycle.