



Optimizing for Energy Using the Xenergy Energy Estimator Tool

Application Note

Tensilica, Inc.
3255-6 Scott Blvd.
Santa Clara, CA 95054
(408) 986-8000
Fax (408) 986-8919
www.tensilica.com



© 2007 Tensilica, Inc.

Printed in the United States of America

All Rights Reserved

This publication is provided "AS IS." Tensilica, Inc. (hereafter "Tensilica") does not make any warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose.

Information in this document is provided solely to enable system and software developers to use Tensilica processors. Unless specifically set forth herein, there are no express or implied patent, copyright or any other intellectual property rights or licenses granted hereunder to design or fabricate Tensilica integrated circuits or integrated circuits based on the information in this document. Tensilica does not warrant that the contents of this publication, whether individually or as one or more groups, meets your requirements or that the publication is error-free. This publication could include technical inaccuracies or typographical errors. Changes may be made to the information herein, and these changes may be incorporated in new editions of this publication.

Tensilica is a registered trademark of Tensilica, Inc. The following terms are trademarks of Tensilica, Inc.: FLIX, OSKit, Sea of Processors, TurboXim, Vectra, Xenergy, Xplorer, and XPRES. All other trademarks and registered trademarks are the property of their respective companies.

Notice

Tensilica, Inc. reserves the right to make changes to its products or discontinue any of its products or offerings without notice.

Tensilica warrants the performance of its products to the specifications applicable at the time of sale in accordance with Tensilica's standard warranty.

Document Change History:

Published August 2007

Contents

1	Introduction	1
2	Xenergy Overview.....	1
3	How Xenergy Works	2
4	Xenergy Use Models.....	3
	Optimizing Application Software.....	3
	Relocating Critical Code to Local Memories.....	4
	Feedback-directed Compilation and Optimization Switch -o3	5
	Optimize Hardware for Energy and Performance.....	5
	Selecting Xtensa Configuration Options.....	6
	Designing Instructions to Optimize Total Energy.....	7
	Effect of Technology Libraries on Total Energy.....	7
	Optimizing Cache Sizes Using Xenergy.....	7
	Processor Frequency and its Impact on Power and Energy	8
	Creating and Modifying TIE to Reduce Energy Usage.....	9
5	Summary.....	11



Figures

Figure 1: Using Xenergy to estimate energy for an application running on a configurable Tensilica Xtensa Processor	2
Figure 2: Software Optimization Using Xenergy	4
Figure 3: Xenergy Flow with Base Configuration	6
Figure 4: Optimizing Cache Sizes.....	8
Figure 5: Xenergy flow with TIE Extensions.....	10

Tables

Table 1: Moving Critical Code to Local Memories.....	5
Table 2: Feedback directed Compilation and Compiler Optimization	5
Table 3: Effects of Configurable Instructions	7
Table 4: Effect of Technology Libraries.....	7
Table 5: Cache Optimization.....	8
Table 6: Effect of Frequency.....	9
Table 7: Benchmark Optimizations Using TIE	11
Table 8: Color Conversion Example Optimization Using TIE.....	11

Abstract

The Xenergy energy estimation tool enables system architects, software developers and TIE instruction designers to evaluate the power and energy dissipation characteristics of design choices and application code early in the development cycle. Xenergy estimates the energy dissipated by application code running on an Xtensa or Diamond processor subsystem consisting of the Xtensa or Diamond processor and the processor's local tightly-coupled memories and caches.

This application note examines the use models for Xenergy. It also demonstrates how to use Xenergy as an effective tool to optimize your software applications, select the right Diamond processor or Xtensa configuration options, and add TIE instruction extensions, all with the goal of optimizing the total processor sub-system energy.



1 Introduction

Energy has become a first order concern while designing SOCs for portable devices. The rapid growth in portable hand-held devices such as MP3 players and cellular phones has fueled a demand for longer battery life and thus, to lower the energy consumed by these devices. These requirements have created a need for a tool that can quickly assist system architects and designers in making the right choices while designing embedded systems. Making energy efficient choices early in the design cycle is more effective than trying to optimize for energy later in the development phase.

The Xenergy energy estimator is a tool that addresses this need by giving system architects and designers an early estimate of energy so that they can quickly determine the tradeoffs between area, performance, and energy of Xtensa and Diamond processors when running an application. This estimate can enable designers to do high-level architectural exploration for energy without running detailed RTL- based energy analysis.

Energy is measured as power times execution time. So, reducing execution time of an application reduces power. One way of doing this reduction is to create new instructions that accelerate an application. However, this has to be done in such a way that any increase in power due to the new instruction does not exceed the benefit in energy that it provides.

Xtensa processors can often use custom instructions (extensions) to significantly reduce the number of instructions executed by the processor there by reducing the overall energy consumption and frequency requirements of the processor. However this comes at the cost of additional hardware being used to add these custom instructions.

Xenergy provides an early, high-level estimate of the energy and power consumed by the processor when running an application. It is recommended to do detailed RTL-based power analysis once the final configuration and extension have been chosen by the designer to get more accurate results.

Xenergy is applicable to Xtensa and Diamond Standard processors. Whereas for the Diamond Standard processors, it can be used to analyze energy dissipated by executing application code, for Xtensa processors it can also be used to analyze the impact of configuration changes and TIE extensions on energy.

2 Xenergy Overview

Xenergy estimates the energy dissipated by application code running on an Xtensa or Diamond processor subsystem consisting of the Xtensa or Diamond processor and the processor's local tightly-coupled memories and caches. Xtensa processors can optionally contain designer-defined TIE instruction extensions. The processor's local memories include instruction cache (including tag and data arrays), instruction RAM, instruction ROM, data cache (including tag and data arrays), data RAM, and data ROM. Xenergy takes as input the process technology, operating condition, and clock period to predict the energy and power dissipated by the Xtensa processor and local memories for running a target application (by default, Xenergy assumes the conditions specified in the processor configuration). Please refer to the *Xenergy User's Guide* for more information on the options provided by Xenergy.

There are four primary use models for the Xenergy tool:

- ◆ Modifying the Xtensa processor configuration options (not applicable for Diamond processors). This includes exploring the impact on energy of different hardware functional units such as multipliers, floating-point unit.
- ◆ Exploring difference cache and local memory configurations. The Xenergy tool can sweep a range of memory configuration options – varying cache size, number of ways, and line size – to determine the energy optimal cache configuration (for both instruction and data caches).

- ◆ Adding or modifying the TIE instruction extensions attached to the Xtensa processor (again, not applicable for Diamond processors).
- ◆ Modifying the application code to tune it for energy (applicable to both Xtensa and Diamond processors).

Xenergy estimates power and energy for a processor sub-system when running a particular application code. Most RTL-based power analysis uses simple diagnostic code or a sample of the complete application because power analysis at the RTL-level can be time consuming. However, Xenergy provides these early energy estimates at the expense of accuracy. The goal of Xenergy, therefore, is to give designers a relative energy analysis between different processor configuration choices or application code tuning choices, so that they can make energy-directed choices.

3 How Xenergy Works

Input to the Xenergy tool includes a software binary, information about which processor the binary is targeting, and information about the process technology and operating conditions. Xenergy then executes the binary on the Tensilica instruction set simulator (ISS) and generates a power and energy report for the processor core and memories. This energy report includes power and energy consumed, broken down into leakage and dynamic components for the processor core and memories connected to the local memory interfaces. This flow is depicted in Figure 1.

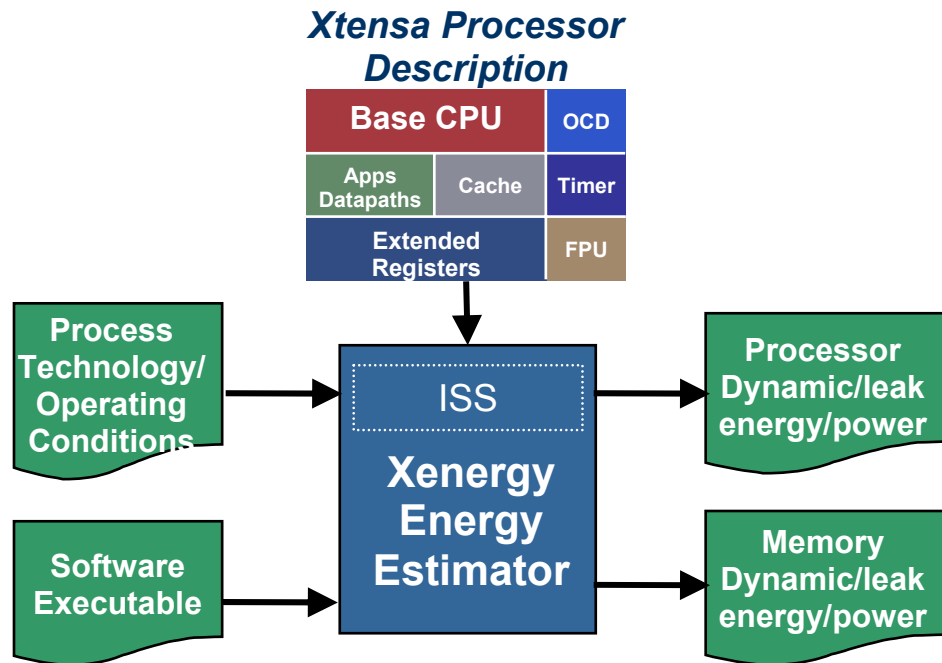


FIGURE 1: USING XENERGY TO ESTIMATE ENERGY FOR AN APPLICATION RUNNING ON A CONFIGURABLE TENSILICA XTENSA PROCESSOR

A focus on total energy consumption is key. Focusing on static milliwatts per megahertz (mW/MHz) power, but ignoring the total energy consumption per unit of workload can often lead to misleading results. For example, a designer may add a set of application-specific instructions to a processor that increases the total size of the processor core and, thereby, increases the average power per clock cycle (increases the mW/MHz). But, if that new instruction set addition dramatically lowers the total clock cycles (milliseconds) required to perform a given functional workload (a target application), then the total energy consumed (power-per-cycle multiplied by

total cycle time) can be reduced. For example, an increase in power-per-clock of 20% might be offset by a 3X speed up in application execution. The mW/MHz number increases 20%, but total energy consumption is actually reduced by 60%.

4 Xenergy Use Models

The Xenergy tool is designed to be used iteratively. You can select a Xtensa configuration, add new instructions, and then optimize your application or start with an optimized application and then determine the need for improving the configuration by modifying the configuration or by adding custom instructions. This application note guides the you by assuming that the you may first want to optimize your application given a base configuration and then add instruction extensions if the performance goals are not achieved by such optimizations.

Optimizing Application Software

The Xenergy energy estimator is useful for optimizing software, even on completed SOCs where the processor cannot be changed. The flow chart in Figure 2 demonstrates the method in which Xenergy can be used to optimize software application. Traditionally, software developers tune their code for performance or code size using Tensilica's standard profiling tools. Now you have the added advantage of the Xenergy tool to fine tune your C code to reduce energy dissipation by the processor and its memories. For example, a developer might use the feedback provided by the Xenergy tool to restructure the allocation of data structures in local and main memories. This can reduce memory and bus accesses, which will lower overall energy expenditures.

Writing efficient code and using the right compiler options can have a positive effect on the processor's overall performance and energy consumption. You can improve the performance of an application several ways, including:

1. XCC Optimization flags
 - a. Optimize for speed (`-O3`)
 - b. Optimize for size (`-Os`)
 - c. Feedback-directed compilation (`-fb_create`, `-fb_opt`)
 - d. Inter-procedural optimization (`-ipa`)
2. Code optimizations
 - a. Reduce function call depths (Avoid exceptions while using a windowed register file)
 - b. Avoid pointer aliasing
 - c. Efficient use cache locking and Write-through caches
 - d. Relocate critical code to local memories
 - e. Use arrays instead of pointers

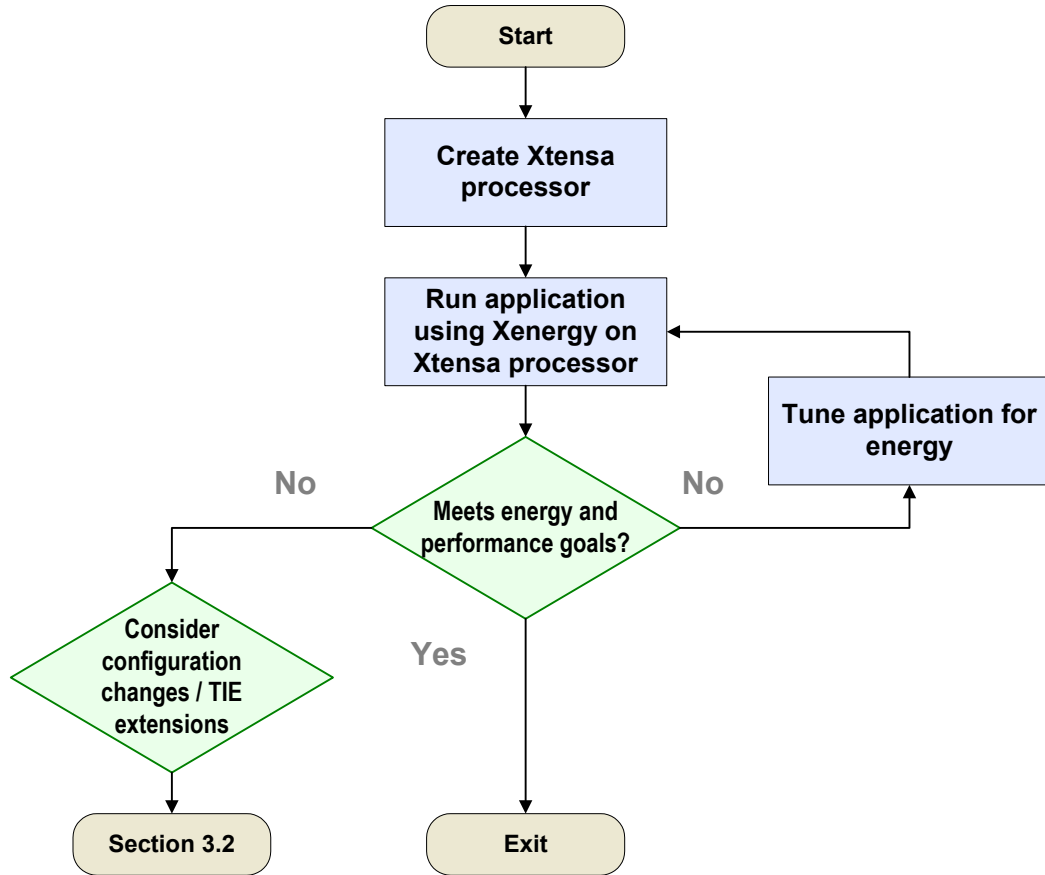


FIGURE 2: SOFTWARE OPTIMIZATION USING XENERGY

Relocating Critical Code to Local Memories

When you profile an application and determine the functions which consume the largest number of cycles (hotspots), you may want to also determine where most of the cycles are being consumed within the function. Xplorer provides a static pipeline view of the instructions being executed within any function. If the cycles are being consumed due to cache misses or due to slow system memory performances, you may want to consider relocating the code to local memories where the overhead associated with loading instructions/data is reduced.

You can annotate functions, types and variables with `__attribute__((section("<section_name>")))` directives, which instruct the linker LSP to place the section into a specific memory. Most memories have pre-defined section names that default LSP's will map into, such as `“.iram0.text”`, `“.dram0.text”`. Please refer to the *Linker Support Packages (LSPs) Reference Manual* for additional details.

The Xplorer workspace provided with this document contains a `ColorConversion` project and a `color_conv` software configuration which does not have caches. When you profile the application, you will notice that the function `ConvertYCbCrToRGB` takes up about 59% of the total cycles. We can speed up the overall performance by ensuring the overhead associated with this function can be eliminated as much as possible.

Table 1 summarizes the overall performance and energy improvement when the `ConvertYCbCrToRGB` is relocated to the Instruction memory using the `__attribute__` directive. Please refer to the *Xtensa C and C++ Compiler User's Guide* for details on using this feature.

TABLE 1: MOVING CRITICAL CODE TO LOCAL MEMORIES

Configuration		Color Conversion
Xtensa processor w/ entire code in system RAM	Cycles	7572886
	Energy (uJ)	1001.1871
Xtensa processor w/ code in system RAM and critical code in Instruction RAM.	Cycles	6653520
	Energy (uJ)	918.1618
Energy Improvement		8.29%
Performance Improvement		12.1%

This increase in performance and energy were achieved with no impact to the area of the processor. Hence, this option is more advantageous than adding caches to the configuration. Caches require SRAM's for Data and Tag and tend to be more expensive than local memories.

Feedback-directed Compilation and Optimization Switch -O3

Various compiler optimizations may benefit from information about profiling and branch frequencies. For example, when the Xtensa processor executes a conditional branch instruction, it incurs additional overhead when the branch is taken compared to falling through to the next instruction. Therefore, it is desirable to have as many branches as possible be fall-through, which XCC can achieve by reordering the code and changing the branch directions.

The overall performance and energy consumption of the processor can be improved by using feedback directed compilation. Using compiler optimization switch -O3 further improves code optimization—thereby improving the overall performance and energy consumed by the processor. Table 2 summarizes the improvement by using these features.

TABLE 2: FEEDBACK DIRECTED COMPILATION AND COMPILER OPTIMIZATION

Configuration		Color Conversion
Xtensa processor w/ entire code in system RAM	Cycles	7572886
	Energy (uJ)	1001.188
Xtensa processor w/ code in system RAM and critical code in Instruction RAM, with feedback-directed compilation and compiler optimization -O3	Cycles	4181725
	Energy (uJ)	594.63
Energy Improvement		40%
Performance Improvement		44%

Optimize Hardware for Energy and Performance

Designers can tune their hardware to achieve performance and energy goals for their target application as well. In this case, the user will create and modify an Xtensa configurable

processor and its associated memories. The user can optimize the system by selecting different configuration options, adding instruction extensions, register files, new execution units, and changing the number and size of local memories and caches.

Optimizing hardware can further be classified into the following two categories

- ◆ Selecting Xtensa configuration options
- ◆ Partitioning hardware and software using TIE

Selecting Xtensa Configuration Options

The flow chart in Figure 3 demonstrates the method in which a Xenergy can be used to optimize a software application by modifying the configuration options of the Xtensa processor. The Xtensa Configuration editor allows users to easily modify their processor configuration. You can configure memories, caches, PIF interface widths, instructions, etc., all of which can have an impact on the overall performance and energy consumed by the processor.

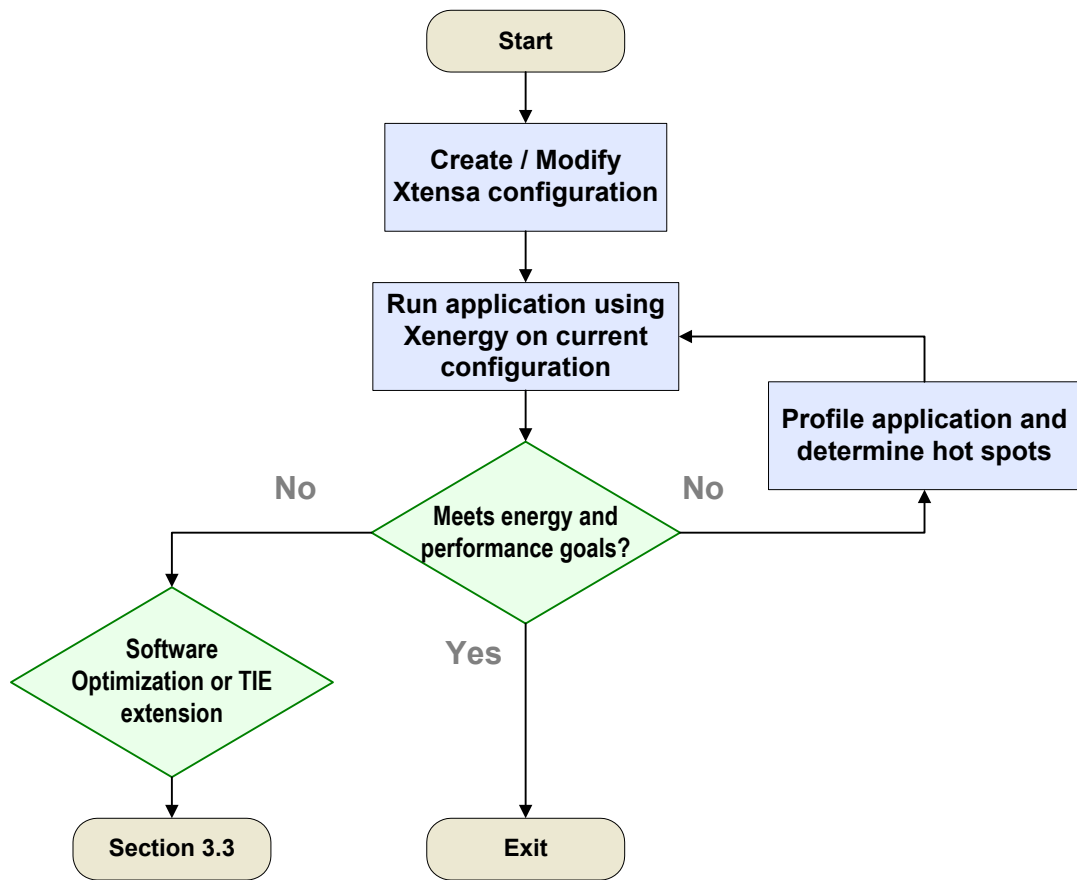


FIGURE 3: XENERGY FLOW WITH BASE CONFIGURATION.

The choice of processor configuration can greatly affect performance and energy consumption. TIE can increase performance and lower energy consumption by many factors, but configuration parameters and choice of memory subsystem can also make a substantial difference on the overall energy consumption by the processor subsystem.

There are two basic techniques that can be used to improve your choice of core processor configuration. Using the first technique, profile your application and take a closer look at the

hotspots. If profiling shows, for example, that a significant amount of time is spent in integer multiplication emulation, adding a multiplier to your configuration will probably help. Using the second technique, build multiple configurations and see how the performance and energy of your application varies across configurations.

Designing Instructions to Optimize Total Energy

When you profile an application to determine the hot spots you need to think about how some of the optional instructions which are available within the Configuration Editor can have an effect on the overall performance and energy consumption of the processor.

Table 3 compares the performance (cycle count) and energy improvement of using a floating point co-processor on an application which uses floating point operations.

TABLE 3: EFFECTS OF CONFIGURABLE INSTRUCTIONS

Configuration		
Base Xtensa configuration	Cycles	2053817
	Energy (uJ)	215.9
Base Xtensa configuration + FPU	Cycles	1583795
	Energy (uJ)	182.9
Energy Improvement		15%
Performance Improvement		22%

Effect of Technology Libraries on Total Energy

The technology library is also one of the configuration options used to create an Xtensa processor. The `-tech` option in Xenergy allows you to specify the process technology to use for processor core and memory energy scaling.

Xenergy enables you to determine the impact of various technology libraries on the total energy consumption for any application running on the processor. Table 4 illustrates data obtained for the `ColorConversion` application using different technology libraries passed as an argument to Xenergy (e.g., `-tech=90g`). Values were obtained using typical operating conditions assuming a clock period of 10ns.

TABLE 4: EFFECT OF TECHNOLOGY LIBRARIES

	180g	130g	130lv	130lvk	130lvkod	90g
Total Power (mW)	50.98	18.41	14.39	14.02	19.64	11.69
Total Energy (uJ)	2101.52	758.77	593.27	577.83	809.54	481.85

Optimizing Cache Sizes Using Xenergy

The Xenergy tool also provides a cache optimization option. If either or both of the instructions and / or data caches are configured, the tool can sweep over an internal database of third-party memory data, varying cache size, number of ways, and line size to determine the optimal cache configuration for the targeted application. There are several optimization targets to choose from, including cycle count, area, energy (dynamic and leakage), or power (dynamic and leakage).

When using this tool with the Dhrystone example (packaged within Xplorer) and trying to optimize for dynamic energy (`-opt dyn_energy`), Xenergy ran the example over 4536 different cache combinations and displayed the best cache configuration for the example. Results are available within a fraction of the time required to run it using the traditional RTL power analysis methods. The Xenergy tool creates a `CacheConfig` summary file sorted in the ascending order of dynamic energy consumption.

```
Cache settings:
ICSize16384_ICNumWays1_ICLineSize32_DCSize4096_DCNumWays1_DCLineSize64

TDK dir:

Software:
C:\workspaces\XX201CE_xenergy\dhrystone\bin\base_nofloat\Debug\dhrystone 10
Tech: 130lv OpCond: Worst Clock(ns): 10.00 SysMemLatency: 1
Cost (dyn_energy): 14954622.48
Cycles: 148466 Instructions: 93755

Energy (uJ)      Dynamic      Leakage      Total
Processor:      8.8944      1.0871      9.9815 (59.77%)
Memory:         6.0602      0.6573      6.7176 (40.23%)
Total:          14.9546 ( 89.55%)  1.7445 ( 10.45%)  16.6991

Power (mW)      Dynamic      Leakage      Total
Processor:      5.9908      0.7322      6.7231 (59.77%)
Memory:         4.0819      0.4428      4.5247 (40.23%)
Total:          10.0728 ( 89.55%)  1.1750 ( 10.45%)  11.2478
```

FIGURE 4: OPTIMIZING CACHE SIZES

From the Xenergy results, the best possible combination to optimize dynamic energy for this application is given in Table 5.

TABLE 5: CACHE OPTIMIZATION

Application	Dhrystone
Instruction Cache Size	16K
Instruction Cache Ways	1 way/ direct mapped
Instruction Cache Line Size	32 Bytes
Data Cache Size	4K
Data Cache Ways	1 way/ Direct mapped
Data Cache line Size	64 Bytes

Processor Frequency and its Impact on Power and Energy

Xenergy enables you to easily measure the impact of frequency changes on energy and power consumption for any application. The `-clock` option within Xenergy allows you to modify the frequency of the processor while running the application. The table below examines the impact of frequency changes on power for the `Dhrystone` and `ColorConversion` applications running on the `color_conv` configuration. The dynamic energy of the processor remains

constant as the frequency changes. However, there is a small variation in the leakage energy with frequency.

TABLE 6: EFFECT OF FREQUENCY

	Total Power @ 100MHz	Total Power @ 200MHz	Total Power @ 300MHz	Total Power @ 400MHz
Dhystone	13.48 mW	25.09 mW	37.04 mW	48.30 mW
Color Conversion	14.291 mW	26.56 mW	38.93 mW	51.24 mW

The total energy consumed by the application remains nearly constant as the frequency increases. However, the rate at which energy is consumed (power) goes up as the frequency increases. To better optimize a processor for lower power, you may want to consider adding application specific TIE instructions which enable you to obtain equal or even better performance than can be obtained by increasing the frequency of your processor. This is described in the section “Creating and Modifying TIE to Reduce Energy Usage”.

There are several other factors which are not discussed in this document that can have an impact on the overall performance and energy consumption by the processor. You can consider these factors while configuring a processor

- ◆ Running code from local memory vs. system memory
- ◆ Use Latches for register file implementation
- ◆ Enable Clock Gating options (can reduce dynamic power of the processor, but can increase the number of gates)
- ◆ Width of the processor interface (PIF) (can increase area and hence increase dynamic power)

Creating and Modifying TIE to Reduce Energy Usage

The flow chart in Figure 5 demonstrates the method in which Xenergy can be used to optimize software application by extending the Xtensa processor with application-specific (TIE) instructions. The *Tensilica Instruction Extension (TIE) Language User’s Guide* provides more information on adding instructions to your Xtensa processor.

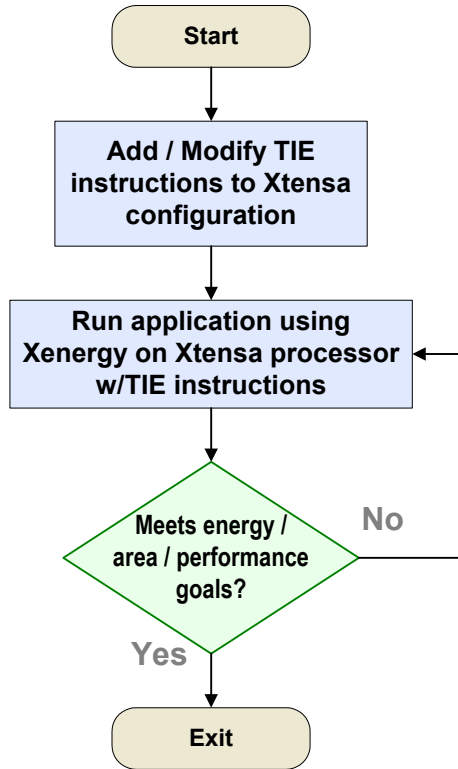


FIGURE 5: XENERGY FLOW WITH TIE EXTENSIONS

Tensilica Instruction Extensions (TIE) enables you to add your own application-specific instructions which allow you to partition tasks between hardware and software. When you profile your application to determine hot spots within the application, you may want to consider moving some of the tasks from software to hardware. This technique works very well especially when algorithms are involved. AES, DES, Viterbi and FFT are good examples where algorithms have been implemented in TIE to boost both performance of a processor and achieve energy efficiency within the processor. This enables users to design a high performance processor without the need to have high frequencies which can result in higher power consumption as explained earlier.

Using TIE to implement algorithms or create new instructions within the processor will mean additional costs (in terms of gates) associated with hardware. You have to carefully examine the tradeoff of area vs. performance vs. energy efficiency for your processor when designing with TIE. However, time-to-market with TIE for the processor can be significantly faster compared to using traditional RTL methods for building an application specific processor.

Table 7 lists the overall performance and energy improvements obtained by extending a processor with TIE for some algorithms.

TABLE 7: BENCHMARK OPTIMIZATIONS USING TIE

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

The Xplorer workspace available with this application note contains the `ColorConversion` example which is described in the *Tensilica Instruction Extension (TIE) Language User's Guide*. The base processor was extended to take advantage of FLIX/Fusion/SIMD feature of Xtensa processors. Some new instructions were added to the processors as well. The results are shown below.

TABLE 8: COLOR CONVERSION EXAMPLE OPTIMIZATION USING TIE

Configuration		Color Conversion
Baseline Xtensa Processor	K Cycles	4181
	Energy (uJ)	594.63
Optimized Xtensa Processor using TIE	K Cycles	553
	Energy (uJ)	88.89
Energy Improvement		86%
Performance Improvement		78%

Overall, there is an 80% improvement in both performance and energy consumption by using TIE instructions. However, this comes with a cost of using an additional 107K gates for implementing the TIE instructions.

5 Summary

The Xenergy energy estimator is a tool for predicting energy for the Xtensa subsystem (processor, caches, local memories). This tool gives early, high-level estimates of energy that enable designers to do architectural and application code exploration to meet their energy goals, without having to run time consuming RTL-based power analysis tools. Designers can get these coarse high-level energy estimates using Xenergy in minutes as opposed to hours for the finer, more accurate RTL-based power analysis.

The Xenergy tool is available for both Xtensa and Diamond processors. Whereas this tool can be used for tuning application software for energy for both Xtensa and Diamond processors, it can be additionally used for configuration exploration, cache exploration, and TIE instruction extension exploration for the Xtensa configurable processors.