Compiler-Based Optimizations Impact on Embedded Software Power Consumption

Mostafa E. A. Ibrahim[1+2], Markus Rupp[1], and S. E.-D. Habib[2]
[1]Institute of Communications and RF Engineering-Vienna University of Technology, Austria
[2] Electronics and Communication Department Faculty of Engineering-Cairo University, Egypt
Email: {mhalas,mrupp}@nt.tuwien.ac.at, seraged@ieee.org

***Abstract*— Compilers traditionally are not exposed to the energy details of the processor. In this paper, we present a quantitative study wherein we examine the influence of the global performance optimizations -o0 to -o3, of the code composer studio C/C++ compiler, on the energy and power consumption. The results show that the most aggressive performance optimization option -o3 reduce the execution time, on average, by 95%, while it increases the power consumption by 25%. Moreover, we inspect the optimizations effect on some other execution characteristics, such as the memory references and the data cache miss rate. The results show that the memory references decreases by 94%, while the IPC increases by 250% and consequently lead to the consumed power increase.**

I. Introduction

In recent years, reducing power dissipation and energy consumption of a program have become optimization goals in their own right, no longer considered a side-effect of traditional performance optimizations which mainly try to reduce program execution times. Power and energy optimizations can be implemented in hardware through circuit design, and by the compiler through compile-time analysis, code reshaping, and hints to the operating system.

Compilers traditionally are not exposed to the energy details of the processor. Current compiler optimizations are tuned primarily for performance and code size. Hence, it is important to evaluate how these optimization options influence power and energy consumption within the processor while running a software kernel.

In this paper, we present a quantitative study wherein we examine the effect of the global optimizations levels -o0 to -o3 of the compiler on the energy and power consumption of the targeted processor. The targeted processor in our experiments is the fixed-point VLIW TMS320C6416T (for the rest of the paper it is referred to as C6416T for brevity) DSP from Texas Instruments. In our experiments we use the Code Composer Studio (CCS), the Texas Instruments C/C++ compiler, to produce the code binaries. This compiler contains many optimizations, including those that specifically target the capabilities and features of the C6416T.

First, we evaluate the effect of the global compiler optimization options on the energy and power consumption of the targeted processor. Second, we analyze the effect on other performance measures such as memory references, the cache miss rate, the instructions per cycle (IPC) and the CPU stall cycles.

This work has been funded by the Christian Doppler Laboratory for Design Methodology of Signal Processing Algorithms, as well as the comet funded the K-project: Embedded Computer Vision.

The paper describes prior research related to this work in Section II and presents a general overview of the experimental platform in Section III. The results of invoking various global compiler optimizations and their effect on the power and energy are shown in Section IV. Finally, the conclusions are drawn in Section V.

II. Previous work

In recent years some attempts to understand the scope of compiler optimizations, from the perspective of power dissipation and energy consumption, of programmable processors have been introduced. Tiwari et al. [1] presented an instruction level power model for a Fujistu 3.3v, 40MHz DSP. Moreover, the effect of two architectural features (dual-memory accesses, and packing of instructions into pairs) on the energy consumption has been illustrated.

With the help of a cycle-accurate energy simulator (SimplePower), a source-to-source code translator, and a number of benchmark codes, Kandemir et al. [2] studied the influence of five high-level compiler optimizations ,such as loop unrolling and loop fusion, on energy consumption.

Valluri et al. [3] provided an evaluation of some general and specific optimizations in terms of the power/energy consumption of the Alpha processor while running some SpecInt95 and SpecFp95 benchmarks. The processor in his work was simulated by means of Wattch (A frame work for analyzing processor power consumption at architectural-level) [4].

Chakrapani et al. [5] also presented a study of the effect of compiler optimization on the energy usage of an embedded processor. Their work targets an ARM embedded core and they use an RTL level model along with Synopsys Power Compiler to estimate power. Seng et al. [6] revised the effect of the Intel compiler general and specific optimizations, for energy and power consumption, for a Pentium 4 processor running some benchmarks extracted from Spec2000.

Zafar et al. [7] examined the effect of loop unrolling factor, grafting depth and blocking factor on the energy and performance for the Philips Nexperia media processor PNX1302. But, they interchangeably use the term energy and power for the same meaning. Hence the improvement in energy is directly related to the performance enhancement.

Finally, Casas et al. [8] studied the effect of various compiler optimizations on the energy and power usage of the low power C55 DSP from Texas Instruments. The work does not consider the effect of the compiler optimizations on many performance measures that significantly affect the power and energy usage, such as the memory access and the instructions per cycle.

III. Measurement Setup

All power measurements are carried out on the DSP Starter Kit (DSK) of the C6416T manufactured by Spectrum Digital Inc. Although the targeted architecture operating frequency ranges from 600 MHz to 1200 MHz, in our setup, the operating frequency is adjusted to 1000MHz and the DSP core voltage is 1.2V. The Agilent 34410A $6\frac{1}{2}$ digit Digital Multi-Meter (DMM) is used for measuring the current drawn by the DSP core. As shown in Fig. 1 the current is captured in term of differential voltage drop across a 0.025Ω sense resistor inserted, by the DSK manufacturer, in the current path of the DSP core.

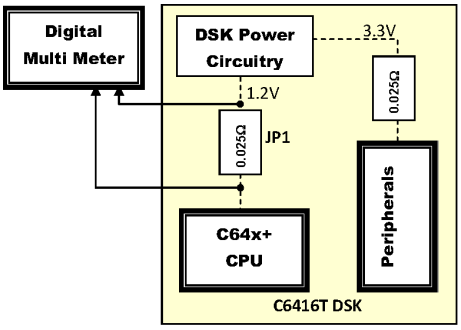

Fig. 1. CPU current measurement setup.

Some common signal and image processing benchmarks from Texas Instruments libraries are used for demonstration purpose. The input data for all used benchmarks are located in the internal data memory. All the benchmarks are executed in an infinite loop to get a stable reading on the DMM.

The C/C++ compiler embedded in the Code Composer Studio (CCS3.1) from Texas Instruments is used for getting the binaries to be loaded to the DSP. This compiler features many levels of optimization mainly tuned for speed and code size. This can be achieved by invoking the {-o0, -o1, -o2, -o3} options for global speed optimization.

IV. Results

The CCS allows the programmer to only choose the optimization level, but it does not allow controlling the individual transformations constituting these optimization levels. Comparing to other processor's compilers, we find that, for example, the compiler of the Philips Trimedia processor allows the programmer to control some of the individual transformations.

We divide our analysis into two parts, first we examine the effect of the global optimization on the power and energy. Second, we analyze the effect of these optimizations on various execution characteristics and try to relate that to the first part.

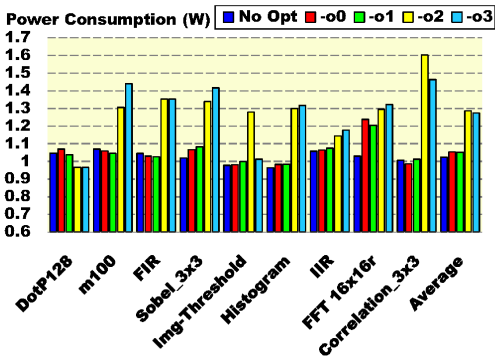

Fig. 2. Power consumption of the C6416 while running different benchmarks vs. different optimization options.

A. Effect on Energy and Power Consumption

Figure 2 presents the measured power consumption at the four optimization levels. It is obvious from Fig. 2 that, on average, these optimization options increase the power consumption. This emphasizes the fact that most aggressive optimizations (although they may lead to minimum execution times) do not necessarily result in the best code from the perspective of power consumption. The highest optimization level -o3 increases the power consumption on average by 25% compared to the no optimization option. For some individual benchmarks, e.g. correlation_3 $\times$ 3, this percentage power increase reach 45%. By invoking -o2 or -o3 much more power is consumed, on average, than when -o0 or -o1 is invoked. The software loop pipelining feature is enabled with -o2 or -o3 allowing better instruction parallelization, and as we will explain later this have a significant impact on the power consumption.

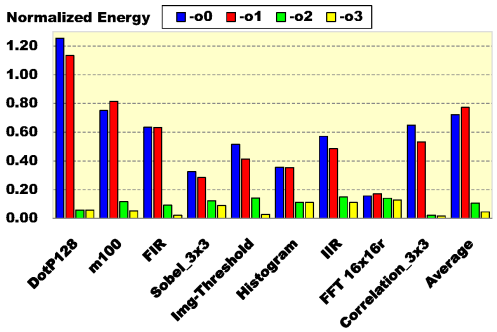

Fig. 3. Normalized Energy versus various optimization options.

Although this data demonstrates that these optimizations increase, on average, the power consumed by the processor, the energy is significantly decreased. Figure 3 shows the normalized energy used by each of the benchmarks and for the average of the benchmarks. The results for each of the benchmarks is normalized to the energy used when the benchmark is run with all optimizations disabled. This significant decrease in the energy is related to the perfect correlation between execution time and energy.

The optimization levels can be classified into two main groups namely -o0 & -o1, enhanced mainly by register mapping, and -o2 & -o3 distinguished by the use of software

pipelined loops (hardware loops). The no optimization, -o0 and -o1 imply the use of pointer registers. however, in no optimization case, data is pre-fetched from memory prior to the execution of the instruction that needs this data. In case of -o0 and -o1 data is fetched simultaneously with the instruction execution, saving CPU cycles and consequently energy. Therefore, if the programm uses many variables, the power consumption increases in companion with an energy drop such as the case in all the benchmarks except the DotP128 which have few number of variables. If the programm uses few variables it presents similar energy drop with unchanged or decreased power consumption, since the registers are loaded less frequently and reused more often.

Invoking -o2 and -o3 provides substantial energy saving than when invoking -o0 or -o1. This can be explicated by the fact that, at -o2 and -o3 the software loop pipelining is enabled which in turn avoids most of the stall cycles resulting better execution time. Given that stall cycles consume lower power than the normal instruction cycles, enhancing the execution time in that way leads to considerable increase in the power consumption.

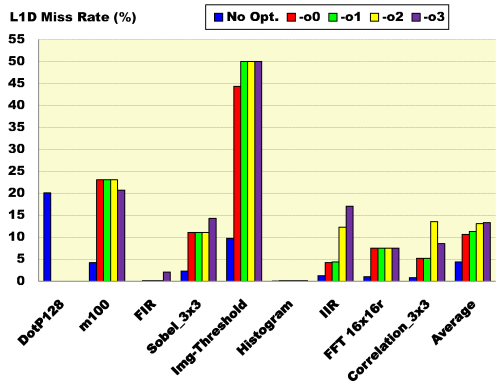

Fig. 4. Impact of optimizations on the L1D miss rate.

B. Effect on Other Execution Characteristics

In order to analyze the previous results we find that it is worth to study the effect of the compiler optimizations on four important execution characteristics: cache misses, memory references, instructions per cycles, and CPU stall cycles.

Figure 4 exposes the effect of different optimization levels to the level 1 data (L1D) cache miss rate. The L1D cache miss rate increases, on average, by almost 200% when invoking -o3. The L1D cache misses consequently require L2D cache/SRAM access which in turn provide additional power consumption. Knowing that one data cache miss causes six stall cycles, we may expect the CPU stall cycles to increase as the miss rate increases. But, as shown in Fig. 5 the CPU stall cycles decreased by 78% when also -o3 is invoked.

The first reason that causes the previous result is that, the L1D cache of this DSP pipelines read misses. Hence, a single L1D read miss takes six cycles when serviced from L2 SRAM, and eight cycles when serviced from L2 cache. Miss pipelining can hide much of the miss overhead (CPU stall cycles) by overlapping the processing of several cache misses. The miss

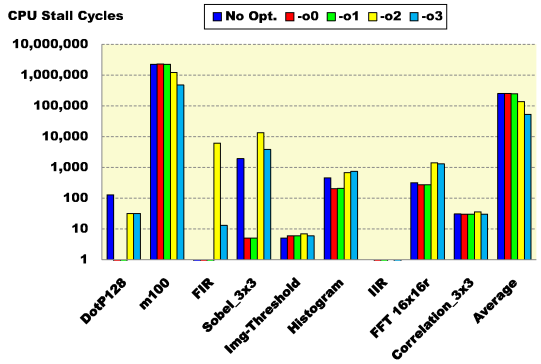

Fig. 5. CPU stall cycles versus different optimization options.

overhead can be expressed as $(4+(2\times M))$ when the serviced from L2D SRAM or as $(6+(2\times M))$ when the serviced from L2D where M is the number of misses. Therefore, the cache miss pipelining provides a reduction in the execution time but it still has no effect on the power consumption.

The second reason is that the write cache miss does not stall the CPU directly because of the use of a L1D Write buffer [9]. This also, affects the execution time but has no effect on the power consumption especially when the write buffer is full.

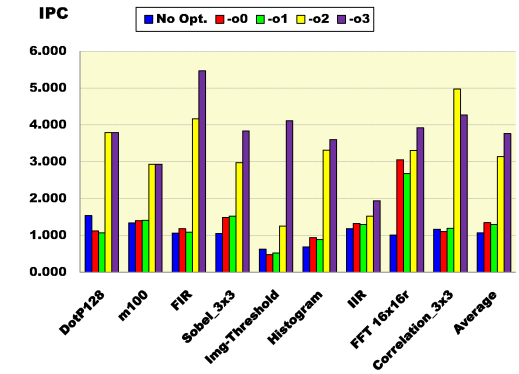

Fig. 6. Effect of various optimization options on the instructions per cycle.

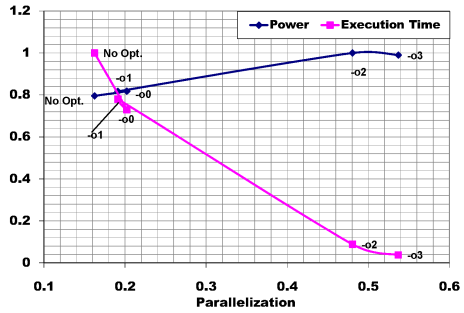

Fig. 7. Parallelization impact on the execution time and the power consumption.

Figure 6 illustrates that the instructions per cycles (IPC) increased by about 250% compared to the case when all optimization options are disabled. This obviously decreases the execution time and hence the energy, because there will be overlapping in the instruction execution per cycle. But, this results in much more parallelization that in turn increases the power consumption.

Figure 7 shows the impact of the parallelization on the consumed power as well as the execution time. While the execution time is directly proportion to the parallelization, the power consumption is inversely proportion.

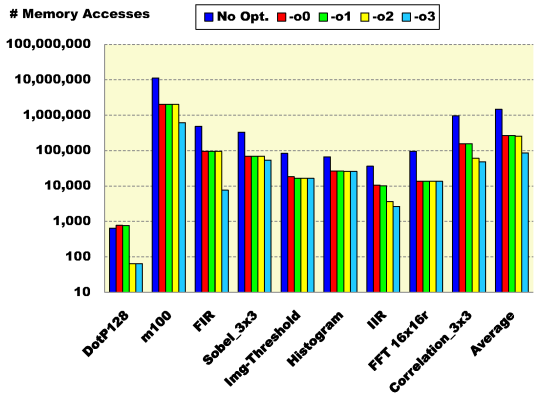

Fig. 8. Effect of different optimization options on the memory references.

Although, Fig. 8 points out that the memory references decreased by 94%, that is expected to save the consumed power, the power is actually increased. This emphasizes our results in [10], [11] that the Instruction Management Unit (IMU), the unit which is responsible for fetching and dispatching instructions, contribution to the total power consumption dominates the memory references contribution.

Finally the relation between the memory access rate of change, the L1D cache miss rate, and the CPU stalls rate of change is summarized in Fig. 9.

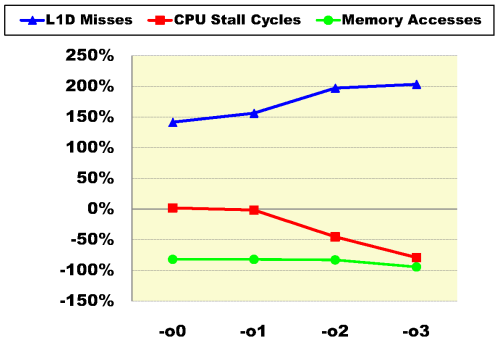

Fig. 9. L1D miss rate versus CPU stall cycles and memory references.

V. Conclusion

In this work, we measure the effect of various compiler optimizations on the energy and power consumption of the fixed-point C6416T DSP from Texas Instruments. The compiler in this study is the C/C++ compiler V.6.0.1 embedded in the CCS3.1. Although some may expect that invoking aggressive performance optimization options may lead to power consumption increase, assuring this assumption and precisely determining the percent of increase is still important. We find that enabling various global speed compiler optimizations lead to considerable increase in the power consumption of the DSP, on average, by 25% when -o3 is invoked. Although these optimizations increase the consumed power by the DSP, we find that the energy usage while running an algorithm is significantly decreased, on average, by 95.6% when -o3 is invoked. This is related to the perfect correlation between execution time and energy usage.

To investigate the cause of this power consumption increase we inspecte the optimizations effect on some other execution characteristics , such as the memory references, the data cache miss rate, the CPU stall cycles and the IPC. Despite the decrease of the memory references by 94%, the IPC increases by 250% and consequently increases the consumed power by 25% which emphasizes our results in [10] that the IMU contribution to the total power consumption dominates the memory references contribution.

The programmer targeting low power design is not encouraged to use this processor. But, if the target is to reach a trade-off between performance and power consumption, the programmer is recommended to compile the programm with -o3 while disabling the software pipelined loop using -mu in conjunction with -o3.

Due to the very limited programmer's control over the individual code transformations invoked by any of the compiler optimization levels, further research is required to evaluate the impact of some source code transformations on the power consumption. Hence, we can introduce power aware optimization levels to the compiler designers.

References

[1] Mike Tien-Chien Lee, Masahiro Fujita, Vivek Tiwari, and Sharad Malik. Power analysis and minimization techniques for embedded dsp software. *IEEE Trans. Very Large Scale Integr. Syst.*, 5(1):123–135, 1997.

[2] Mahmut Kandemir, N. Vijaykrishnan, and Mary Jane Irwin. Compiler optimizations for low power systems. *Chapter 10, Robert Graybill and Rami Melhem: Power aware computing*, pages 191–210, 2002.

[3] M. Valluri and L. John. Is compiling for performance == compiling for power? In *Proceedings of the 5th Annual Workshop on Interaction between Compilers and Computer Architectures*, Monterrey, Mexico, Januarry 2001.

[4] David Brooks, Vivek Tiwari, and Margaret Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. *SIGARCH Comput. Archit. News*, 28(2):83–94, 2000.

[5] L. N. Chakrapani. The emerging power crisis in embedded processors: What can a poor compiler do. In *Proceedings of the International Conference on Compilers, Architecture, and Synthesis for Emdedded Systems(CASES)*, November 2001.

[6] John S. Seng and Dean M. Tullsen. The effect of compiler optimizations on pentium 4 power consumption. In *Proceedings of the Seventh Workshop on Interaction between Compilers and Computer Architectures INTERACT'03*, pages 51–56, Washington, DC, USA, February 2003. IEEE Computer Society.

[7] N. Zafar Azeemi and M. Rupp. Energy-aware source-to-source transformations for a VLIW DSP processor. In *proceedings of the 17th ICM'05*, pages 133–138, Islamabad, Pakistan, December 2005.

[8] C.J. Bleakley J.R. Gonzalez Miguel Casas-Sanchez, Jose Rizo-Morente. Effect of compiler optimizations on dsp processor power and energy consumption. In *Conference on Design of Circuits and Integrated Systems (DCIS)*, Seville, Spain, November 2007.

[9] Texas Instruments. *TMS320C6416T,DSP Two-Level Internal Memory Reference Guide*, February 2006. SPRU610C.

[10] Mostafa E. A. Ibrahim, Markus Rupp, and S. E.-D. Habib. Power consumption model at functional level for VLIW digital signal processors. In *Proceedings of the conference on Design and Architectures for Signal and Image Processing DASIP'08*, pages 147–152, Bruxelles, Belgium, November 2008.

[11] M. E. A. Ibrahim, M. Rupp, and H. A. H. Fahmy. Power Estimation Methodology for VLIW Digital Signal Processor. In *proceedings of the Conference on signals, systems and computers Asilomar08*, pages 1840–1844, Asilomar, CA, US, October 2008. IEEE.