SPIRAL: JOINT RUNTIME AND ENERGY OPTIMIZATION OF LINEAR TRANSFORMS

Marek Telgarsky, James C. Hoe, José M. F. Moura

Carnegie Mellon University
Department of Electrical and Computer Engineering
5000 Forbes Ave.
Pittsburgh, PA 15232

ABSTRACT

There is much interest into joint runtime and energy optimization of implementations of signal processing algorithms. Applications in domains such as embedded computing, sensor networks, and mobile communications often require processing of signals under simultaneous runtime, energy and/or power constraints. Hence, in addition to runtime, power and energy are first-order design considerations for both hardware and software developers in those domains. This paper studies the *automatic* generation of software implementations of digital signal processing (DSP) transforms that are optimized with respect to both runtime and energy. We explore the impact of algorithm selection (a software technique) and voltage-frequency scaling (a hardware technique) on the runtime and energy of computing fast linear transforms. We use SPIRAL, a code generation system, to enumerate automatically many alternative algorithms for the discrete Fourier transform. We measure the runtime and energy of these algorithms at different voltage-frequency settings of an Intel Pentium M microprocessor. We report experimental results supporting that algorithm selection and voltage-frequency scaling do achieve the following: (1) have large impact on the runtime and energy of computing the discrete Fourier transform on a microprocessor; and (2) enable the optimization of important joint runtime-energy objectives.

1. INTRODUCTION

The SPIRAL code generation system [1] generates platform-adapted linear transform implementations whose runtime is competitive with the best-available hand-coded implementations and/or vendor libraries on a given platform. SPIRAL achieves optimization by exploring and evaluating automatically the space of algorithm design for fastest runtime. This basic approach also has potential for impacting the optimization of energy and other joint runtime-energy optimization metrics.

This research is supported in part by DARPA through the Department of Interior grant NBCH1050009 and by NSF grant ACI-0325687.

Energy and power have become major concerns for not only the traditional low-power applications but also increasingly for high performance computing applications (due to power delivery and cooling concerns). Recent microprocessors, whether for lower-power embedded computing, mobile computing or high-performance servers, have begun to support *voltage-frequency scaling* where the performance of the processor can be reduced to conserve energy when performance is not critical.

In this paper, we study the automatic generation of software implementations of signal processing transforms that achieve joint optimization of runtime and energy. We consider two promising degrees of freedom, algorithm selection and voltage-frequency scaling, and evaluate their impact on the runtime and energy of linear transform computations on microprocessors. Specifically, we evaluate a broad range of SPIRAL-generated algorithms for the discrete Fourier transform at varying voltage-frequency settings available on the Intel Pentium M microprocessors. We demonstrate that algorithm selection and voltage-frequency scaling have large impact on the runtime and energy of computing the discrete Fourier transform on a microprocessor. There is no single universally applicable joint runtime and energy optimizing criterion. Rather, there is a Pareto optimal set of designs in the runtime-energy optimization plane. Different algorithms and different voltage-frequency scaling settings can create a large population of runtime and energy choices. A subset of these forms the Pareto optimal set.

Paper outline. We begin in Section 2 with an overview of the joint runtime-energy optimization problem. Section 3 next presents the two degrees of freedom we employ to affect runtime and energy, namely algorithm selection and voltage-frequency scaling. Section 4 describes our experiments and presents our results. Section 5 discusses our findings based on the experimental results. Section 6 provides our conclusions.

2. JOINT RUNTIME-ENERGY OPTIMIZATION

This section provides an overview of the joint runtime-energy optimization space. We review the metrics for joint runtime-energy optimization and Pareto optimality.

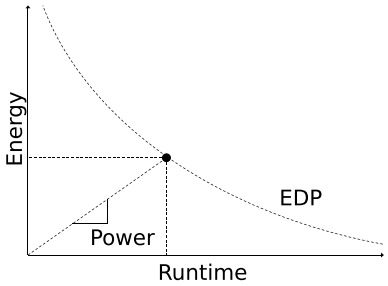

Fig. 1. The conceptual design space for optimizing runtime and energy.

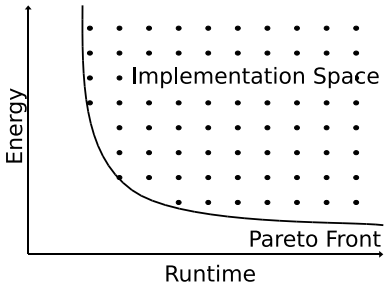

Fig. 2. Pareto optimal front for joint runtime-energy optimizations.

2.1. Runtime and energy

Computing a linear transform on a processor requires some time (T) and energy (E). For a given transform, factors, such as algorithm, compiler, and processor performance, affect the runtime and energy. Each possible instantiation of the transform's execution would correspond to a point on a two-dimensional plot with runtime and energy as the x- and y-axes respectively (Fig. 1). An execution with lower runtime would be further to the left; an execution with lower energy would be further in the down direction.

In a joint optimization, one not only reduces runtime and energy, i.e., moving towards the origin in Fig. 1, but also considers the tradeoff between them. Such joint-optimizations often are formulated as the optimization of a scalar composite objective function $f(E, T)$. For example, an often minimized composite metric is power (E/T). Another well-known composite metric is the energy-delay-product (EDP or $E \cdot T$) [2]. By balancing the gain and loss between runtime and energy, many different points in the runtime-energy plane can achieve the same EDP (as indicated by the EDP isogram in Fig. 1).

In addition to the scalar objective, a joint optimization can also be constrained by inequalities such as ceilings on runtime and/or energy. The exact formulation of the objective function and any associated constraint is decided by the specific requirements of the application context.

2.2. Pareto optimality

Considering all possible factors that could affect runtime and energy, all possible executions of a given transform would populate the runtime-energy plane with a large number of points as depicted in Fig. 2. Regardless of the choice of objectives and constraints, Pareto optimality theory requires that the optimal solution be a member of the Pareto optimal set [3]. A point is a member of the Pareto optimal set only if there does not exist another point with both lower runtime *and* energy. In Fig. 2, we show the Pareto optimal front, which is the line through the Pareto optimal set. A joint runtime-energy optimization corresponds to finding the Pareto optimal set and optimizing the objective function over the members that satisfy all given constraints.

3. FACTORS AFFECTING RUNTIME AND ENERGY

In this paper, we examine two factors that affect runtime and energy to achieve joint optimization of linear transform computation on a processor. The first factor is the algorithm design and selection. The second factor is the operating voltage and frequency of the processor, which can be adjusted on modern low-power processors including the Intel Pentium M and Intel X-Scale.

3.1. Algorithm Design in SPIRAL

SPIRAL, [1], produces platform adapted code for linear transforms (such as the discrete Fourier transform, discrete trigonometric transforms, wavelets, and filters). Inside the SPIRAL framework, knowledge about the algorithmic implementation of transforms is captured symbolically as parameterized factorization rules (e.g., the Cooley-Tukey rule for the discrete Fourier transform). By choosing different rules and factorization parameters in the recursive applications of the factorization rules, SPIRAL can generate *fast*[1] linear transform algorithms with different memory access patterns and different operation mixtures and orderings. In addition, SPIRAL applies a set of parameterizable high-level code optimizations.

To optimize for runtime, SPIRAL enumerates and evaluates different algorithmic implementations of a given transform with the help of a feedback-driven search engine. For runtime optimizations, SPIRAL can find transform implementations that are competitive with the best-available vendor libraries for a given platform (e.g., Intel Integrated Performance Primitives for the Intel P4 microprocessors). The question of interest is whether algorithm selection and code optimization can have the same impact on energy and other joint runtime-energy objectives.

[1]We stress that all these algorithms are $O(n \log n)$ where n is the dimension of the transform. Their different runtimes arise from their data flows and the impact of the memory hierarchy of the computing platform.

3.2. Voltage-frequency scaling

To minimize energy when performance is not critical, modern low-power processors allow software to adjust the operating voltage and frequency, a mechanism commonly referred to as dynamic voltage and frequency scaling [4]. The operating voltage and frequency is increased or decreased together in specific combinations. For example, the Intel Pentium M model 770 can operate at six operating points: 2133MHz at 1.34V, 1866MHz at 1.27V, 1600MHz at 1.2V, 1333MHz at 1.13V, 1067MHz at 1.06V, and 800MHz at 0.99V [5]. When the operating frequency is reduced, the first-order effect is an approximately linear reduction in the effective performance of the processor. On the other hand, reducing operating voltage leads to a much better than linear reduction in the power dissipation of the processor. Thus, voltage-frequency scaling results in a significant reduction in energy for a given computation—longer runtime but much lower power.

4. EXPERIMENTAL RESULTS

For a transform, algorithm selection and voltage-frequency scaling can be combined to generate different points in a joint runtime-energy optimization plane. In this section, we present an empirical study on the effects of algorithm selection and voltage-frequency scaling on the runtime and energy of the discrete Fourier transforms (DFT).

4.1. Methodology

The host system under test is an Intel Pentium M model 770 microprocessor on an AOpen i615GMm-HFS motherboard, running Linux 2.6.10. The runtime of a DFT algorithm on the microprocessor is determined using built-in cycle counters. The energy of executing a transform cannot be measured directly. Instead, instantaneous power is sampled by measuring the supply current to the microprocessor. Energy is computed by integrating power over time.

In our test system, the chassis power supply feeds the microprocessor through two dedicated 12V supply lines. We sample the supply current 1400 times per second using an Agilent 34134A AC/DC current probe and an Agilent 34401A digital multimeter. The functionality of measuring the runtime and energy of a transform implementation is fully automated in SPIRAL.

4.2. Results

We instruct SPIRAL to search over 100 randomly generated fast algorithms for each two-power sized DFT transforms up to $\mathrm{DFT}_{1\mathrm{M}}$. We measure runtime and energy at all six voltage-frequency settings. We observe two classes of behaviors distinct for large transforms (i.e., $\mathrm{DFT}_{64\mathrm{K}}$ and larger whose data working set exceeds the 2MByte on-chip cache) and small transforms (i.e., $\mathrm{DFT}_{32\mathrm{K}}$ and smaller whose data working set fits entirely in the cache).

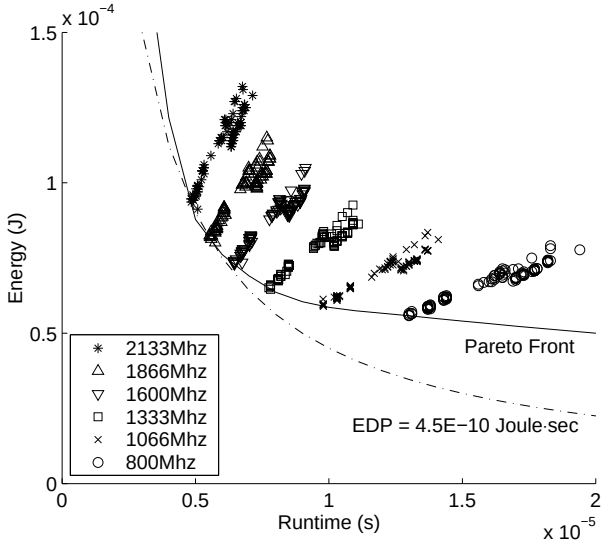

Fig. 3. Runtime and energy when varying DFT_{256} algorithms and voltage-frequency scaling.

Fig. 3 and Fig. 4 report for DFT_{256} (representative of in-cache DFT sizes) and $\mathrm{DFT}_{1\mathrm{M}}$ (representative of out-of-cache DFT sizes) the runtime and energy of different algorithms at different voltage-frequency settings. The data points corresponding to different algorithms executing at the same voltage-frequency setting form six discernible bands in each graph. The distribution of data points in Fig. 3 and Fig. 4 supports that both algorithm selection and voltage-frequency scaling have a large impact on the runtime and energy of computing DFT on a processor. Furthermore, the data points form a Pareto optimal front that permits interesting optimizations of joint runtime-energy objectives.

5. DISCUSSION

In Fig. 3 and Fig. 4, for a given voltage-frequency setting, the fastest algorithm is also the lowest energy algorithm. In other words, algorithm selection for runtime optimization also optimizes for energy. This is not surprising because more runtime-efficient algorithms tend to perform less computation and involve fewer data movements. Furthermore, due to the proportional relationship between runtime and energy for different algorithms at a given voltage-frequency setting, processor power dissipation is roughly constant independent of the algorithm. On the other hand, voltage-frequency scaling results in a wide range of power levels.

For the six points in the Pareto optimal set of DFT_{256}, the runtime varies linearly with the inverse of the operating frequency. This is expected for on-chip computations because the entire processor is slowed down, including the on-chip

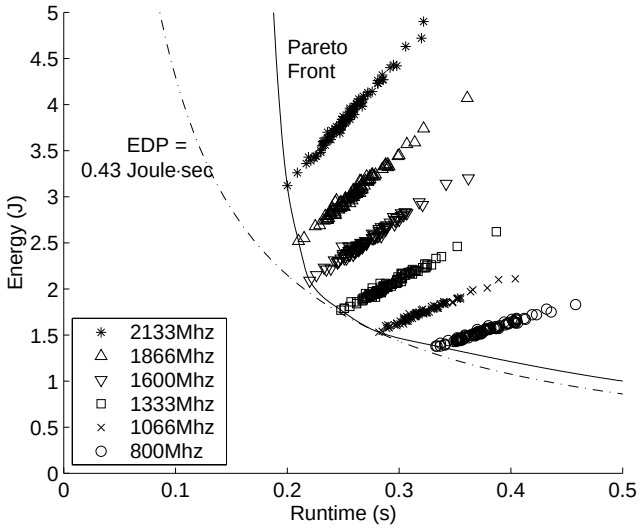

Fig. 4. Runtime and energy when varying $\mathrm{DFT}_{1\mathrm{M}}$ algorithms and voltage-frequency scaling.

caches. On the other hand, for the six points in the Pareto optimal set of $\mathrm{DFT}_{1\mathrm{M}}$, the runtime slows less than expected when operating frequency is reduced. This is because the runtime of large DFT computations is dominated by a large number of expensive off-chip DRAM accesses due to frequent level-2 cache misses. This dominating off-chip component in the runtime of large DFT computations is not increased when the frequency of the processor is reduced.

The minimization of the energy-delay-product (EDP) criterion also illustrates the difference between compute-bound and memory-bound computations. Fig. 3 and Fig. 4 include EDP isograms through their respective lowest-EDP points. Fig. 3 supports that for compute-bound small DFT computations, EDP is minimized at the highest performance voltage-frequency setting (i.e., maximum frequency and maximum voltage). For memory-bound large DFT computations, EDP is minimized at a lower-performance setting.

6. CONCLUSION

SPIRAL explores the space of runtime and energy efficient software implementations of linear transforms automatically. The degrees of freedom exploited in the paper are algorithm selection and voltage-frequency scaling. We study the impact of these degrees of freedom on uniprocessor DFT implementations executing on the Intel Pentium M. An expected conclusion of the study is that with respect to algorithm selection, the two goals of runtime and energy minimization are highly correlated—fast implementations running at a given voltage-frequency setting are energy efficient. Voltage-frequency scaling provides the opportunity for tradeoff between runtime and energy. This tradeoff, captured by the Pareto front, changes with problem size. As the Pareto front changes, so does the location of the optimal operating point for a given metric, like EDP. Finding the optimal operating point for a given problem size and optimization criterion is best done with a system like SPIRAL. SPIRAL can find these optimal operating points automatically.

We are currently extending SPIRAL to generate multi-threaded transform code to address the energy/power vs. runtime tradeoff in the increasingly-popular multi-core processors. Although traditionally a performance optimization technique, parallelization is, in fact, an efficient joint runtime-energy optimization. Under ideal parallelization by a factor of N, runtime is reduced by a factor of N, but the energy required by the computation is not changed, thus reducing energy-delay-product by a factor of N. Furthermore, parallelization can be combined with voltage-frequency scaling to achieve the same level of performance as a single-thread execution but using lower energy and power (e.g., [6]), because voltage-frequency scaling reduces energy and power more drastically than runtime performance. The SPIRAL design generation and exploration framework is particularly well-suited for automating this complex optimization problem.

7. REFERENCES

[1] Markus Püschel, José M. F. Moura, Jeremy Johnson, David Padua, Manuela Veloso, Bryan W. Singer, Jianxin Xiong, Franz Franchetti, Aca Gačić, Yevgen Voronenko, Kang Chen, Robert W. Johnson, and Nick Rizzolo, "SPIRAL: Code generation for DSP transforms," *Proceedings of the IEEE, special issue on "Program Generation, Optimization, and Adaptation"*, vol. 93, no. 2, pp. 232–275, 2005.

[2] Ricardo Gonzalez and Mark Horowitz, "Energy dissipation in general purpose microprocessors," *IEEE Journal of Solid-State Circuits*, vol. 31, no. 9, pp. 1277–1284, September 1996.

[3] R. E. Steuer, *Multiple Criteria Optimization: Theory, Computation, Application*, Krieger, 1989.

[4] Anantha Chandrakasan, William J. Bowhill, and Frank Fox, Eds., *Design of High-Performance Microprocessor Circuits*, Wiley-IEEE Press, 2000.

[5] Intel Corporation, *Intel Pentium M Processor with 2-MB L2 Cache and 533 MHz Front Side Bus Datasheet*, July 2005, Reference Number 305262-002.

[6] Murali Annavaram, Ed Grochowski, and John Shen, "Mitigating Amdahl's law through EPI throttling," in *Proceedings of the 32nd International Symposium on Computer Architecture*, June 2005, pp. 298–309.