Dynamic Phase-based Optimization of Embedded Systems

Tosiron Adegbija and Ann Gordon-Ross*
Department of Electrical and Computer Engineering, University of Florida, Gainesville, FL 32611, USA
tosironkbd@ufl.edu & ann@ece.ufl.edu
**Also affiliated with the NSF Center for High-Performance Reconfigurable Computing (CHREC) at UF*

***Abstract*—Phase-based optimization specializes system configurations to runtime application requirements in order to achieve optimization goals. Due to potentially large design spaces in configurable systems, one major challenge of phase-based optimization is determining the best configuration for achieving optimization goals without incurring significant optimization overhead during design space exploration. This work proposes *phase distance mapping,* which uses the correlation between phases and the phases' characteristics to dynamically determine optimal or near-optimal configurations with minimal design space exploration, thereby minimizing optimization overhead.**

Keywords—phase based tuning, phase distance mapping, dynamic optimization, energy savings, low power, configurable hardware

I. Introduction and Motivation

Much research has focused on achieving optimization goals (e.g., reduce cost, energy consumption, time to market, etc.) in embedded systems. However, embedded systems have intrinsic stringent design constraints, imposed by battery capacity, cost, physical size, consumer market competition, etc., that make optimization challenging. These optimization challenges are exacerbated by the dynamic nature of executing applications, requiring optimizations to dynamically specialize (or tune) configurable system parameters (e.g., cache size, associativity, and line size; core frequency and voltage; etc.) to different application execution phases.

A phase is an execution interval where application characteristics (e.g., cache misses, instructions per cycle, branch mispredicts, etc.) are relatively stable. Since same-phased intervals tend to have the same optimal configurations for achieving optimization goals [11], dynamic phase-based optimization increases optimization potential by dynamically determining best (optimal or near-optimal) configurations for different execution phases. However, phase-based optimization increases the potential for optimization overhead (e.g., energy, performance, etc.), especially in systems with large design spaces, since the best configurations for multiple phases must be determined.

To address the challenge of determining the best configurations for different phases, several tuning methods have been proposed. Exhaustive search methods (e.g., [15]) physically explore the design space by executing and evaluating different configurations, however, these methods incur significant optimization overhead while executing non-optimal configurations. Heuristic search methods (e.g., [6]) use intelligent heuristics/algorithms to prune the design space, however, these heuristics still execute non-optimal configurations and incur optimization overhead. Analytical methods (e.g., [9]) significantly reduce optimization overhead by directly determining or predicting the best configurations based on the design constraints and application characteristics. However, most previous analytical methods are either computationally complex or not dynamic.

We propose a computationally simple and dynamic analytical method—*phase distance mapping (PDM)*—to determine the phases' best configurations without significant optimization overhead. Rather than tuning all of the phases in the phase space (i.e., collection of all distinct phases), PDM tunes only one phase—the *base phase*—and uses the correlation between the base phase's characteristics and subsequent new phases' characteristics to predict the new phases' best configurations based on the base phase's best configuration. Our proposed approach offers the following advantages and contributions:

- a low overhead and dynamic analytical method for determining a phase's best configuration,
- significant reduction in optimization overhead as compared to extensive design space exploration,
- minimal designer effort, and
- applicability to various dynamic optimization scenarios.

II. Phase Distance Mapping

Phase distance mapping is based on the hypothesis that the more disparate two phases' characteristics are, the more disparate the phases' best configurations are likely to be. Thus, given a phase *X* and *X*'s best configuration, we can predict another phase *Y*'s best configuration based on the distance between *X's* and *Y*'s characteristics, known as the *phase distance* between both phases. We define the *configuration*

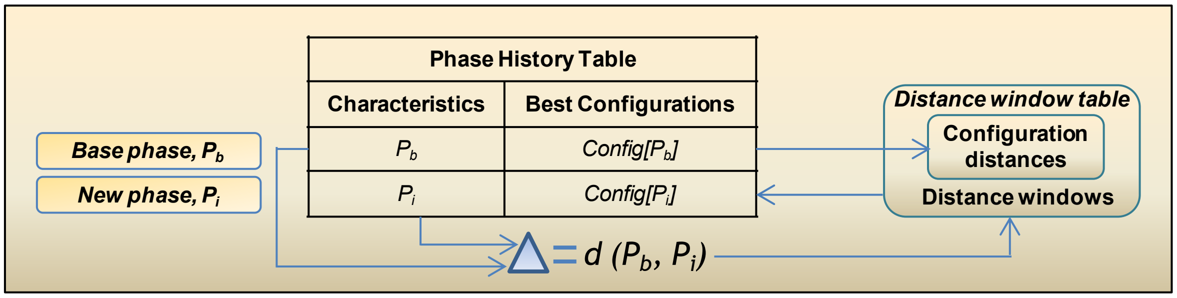


Figure 1. Phase distance mapping (PDM) overview

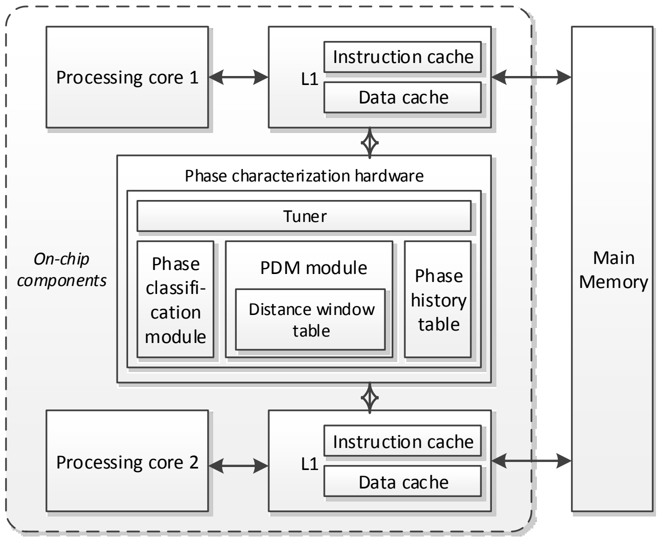


Figure 2. Phase-based tuning architecture for a sample dual-core system [1]

distance as the difference between the two phases' configurations, represented by the difference in configurable system parameter values between the two configurations.

Fig. 1 depicts an overview of PDM. When a new phase P_i is executed, PDM computes the phase distance, denoted as d (P_b, P_i), between P_i's characteristics and the base phase P_b's characteristics. The base phase's best configuration is previously determined using any arbitrary efficient tuning method, such as [6]. To accurately represent the correlation between phase characteristics and configurations, the phase distance must be computed using characteristics that are most impacted by the configurable parameters. For example, since cache characteristics are most impacted by cache configurations, the phase distance could be computed using cache miss rates when optimizing cache configurations.

We evaluated various methods for calculating the phase distance, including *normalization* and *Euclidean distance*. Normalization can be used when only one characteristic is being considered during optimization. For example, when optimizing cache configurations, P_i's cache miss rates can be normalized to P_b's cache miss rates to evaluate d (P_b, P_i). Alternatively, Euclidean distance can be used when multiple configurable parameters are being optimized to impact multiple characteristics. For example, to optimize the data cache, instruction cache, and clock frequency, d (P_b, P_i) can be computed as:

$$\sqrt{(iMR_{Pb} - iMR_{Pi})^2 + (dMR_{Pb} - dMR_{pi})^2 + (IPC_{Pb} - IPC_{Pi})^2}$$

where *iMR, dMR,* and *IPC* represent the instruction cache miss rate, data cache miss rate, and instructions per cycle, respectively.

After computing d (P_b, P_i), PDM searches the *distance window table* for the *distance window* that the phase distance maps to. A distance window is a phase distance range, with a minimum Win_L and maximum Win_U value, representing P_i's configuration distance from P_b, and a phase distance maps to a distance window when $Win_L < d$ (P_b, P_i) $< Win_U$. Each distance window contains a configuration distance from P_b's best configuration (e.g., $L_b * 2$, where L_b is P_b's cache line size), which is used to determine P_i's best configuration when d (P_b, P_i) maps to that distance window.

Distance windows can be created statically or dynamically. Static distance windows require a priori knowledge of the system's applications/application domains, significant a priori analysis of the applications to determine the configuration distances with respect to the base phase, and significant designer effort. To overcome the limitations of static distance windows, we developed a low-overhead, low-designer-effort algorithm that dynamically created distance windows during runtime [1]. This algorithm eliminated designer effort and made PDM more amenable to general purpose systems with unknown applications, without sacrificing the optimization performance.

After P_i's best configuration is determined, the configuration is then stored in the *phase history table* [11] for subsequent executions of P_i.

III. Results

We have evaluated PDM in various optimization scenarios, including *cache tuning* and *thermal-aware phase-based optimization.* This section summarizes the motivations and results obtained from using PDM in these optimization scenarios.

A. Cache Tuning

Since caches have been widely used to bridge the processor-memory performance gap, and the memory hierarchy accounts for a significant portion of an embedded system's microprocessor's overall power consumption, caches are good candidates for dynamic optimization. Cache tuning determines the best cache configuration, such as cache size, associativity, and line size, that best achieves optimization goals.

Fig. 2 depicts a sample phase-based cache tuning architecture for a sample dual-core system, which can be extended to any *n*-core system. On-chip components include the processing cores with private level one (L1) instruction and data caches and the *phase characterization hardware*, comprised of the *tuner, phase classification module, phase history table,* and the *PDM module,* which includes the *distance window table.* The tuner changes configurable parameters and evaluates each configuration, the phase classification module groups similar intervals into phases, and the phase history table stores phase characteristics and best configurations [11].

The PDM module determines a new phase's best configuration and constitutes the only additional hardware overhead imposed by our work. We have implemented the PDM module in synthesizable VHDL and quantified the area and power consumption using Synopsys Design Compiler [14]. Results showed that the PDM module only imposes a 0.30% area overhead and a 0.11% power overhead with respect to a MIPS32 M4K 90nm processor [8], showing that our work constitutes minimal hardware overhead.

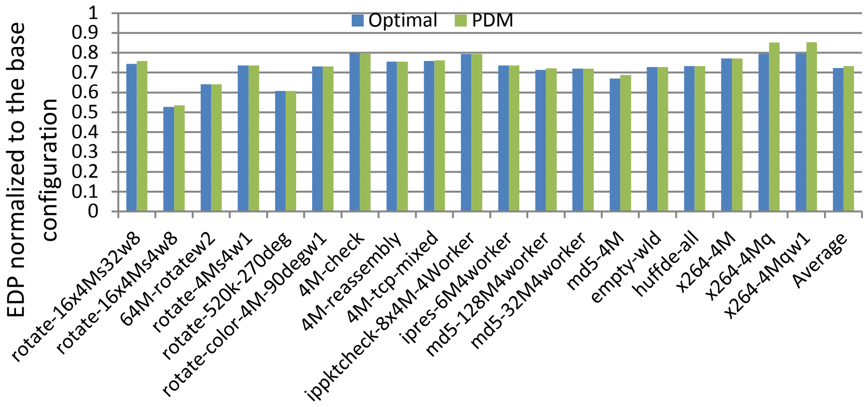


Figure 3. EDP savings as compared to the base configuration

We evaluated PDM's optimization potential using nineteen workloads, representing a variety of realistic real-world applications, from the EEMBC Multibench benchmark suite [5]. We modeled a highly configurable cache architecture similar to [15] with the GEM5 simulator [4] and used the energy delay product (EDP) as our evaluation metric. The cache offered configurable cache size, line size, and associativity ranging from 2 to 8 Kbyte, 16 to 64 byte, and 1- to 4-way, respectively, in power-of-two increments. We quantified PDM's EDP savings with respect to a base cache configuration with cache size, line size, and associativity of 8 Kbyte, 64 byte, and 4-way, respectively. This base cache configuration represents an averagely good configuration for all the phases evaluated in our experiments.

Fig. 3 depicts the EDP savings achieved by the PDM configurations and optimal configurations (determined by exhaustive search) as compared to the base configuration's EDP. On average over all of the phases, PDM achieved EDP savings of 27%, with savings as high as 47%. PDM determined the optimal configurations for twelve out of the nineteen phases, with configurations within 1% of the optimal, on average. Compared to exhaustive search, PDM reduced the exploration time by an average of 95%, and up to 98% for some phases. These results show PDM's ability to achieve significant EDP savings with minimal designer effort while significantly reducing optimization overhead.

B. Thermal-aware Phase-based Optimization

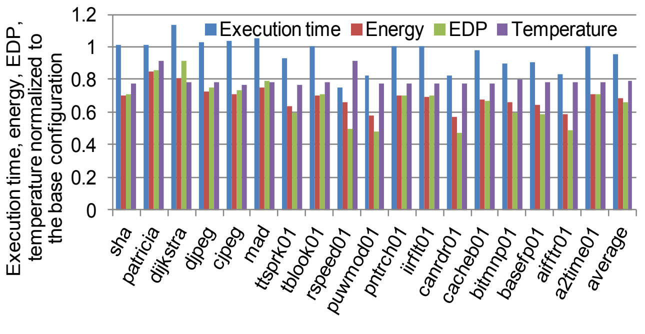
Due to area, cost, and energy constraints, most embedded systems have fewer cooling options as compared to general purpose computers. Thus, much research focuses on optimizing the temperature in embedded systems to prevent thermal emergencies, which could lead to reduced reliability, reduced mean time to failure (MTTF), or even permanent chip damage.

We developed a *thermal-aware phase-based tuning* algorithm (*TaPT*) [2], using PDM concepts, which dynamically determined Pareto optimal cache configurations and clock frequency configurations (using dynamic frequency scaling [12]) that traded off execution time, energy, and temperature in embedded systems. However, optimizing multiple optimization goals presented a multi-objective optimization problem, where one optimization goal could adversely impact other optimization goals (e.g., optimizing the temperature could adversely impact the execution time). Thus, we leveraged the *strength Pareto evolutionary algorithm II (SPEA2)* [16], which is a well-known and effective evolutionary algorithm for solving multi-objective optimization problems.

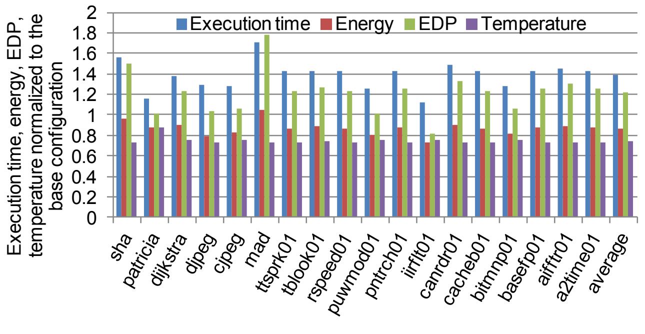
TaPT contained three designer-specified priority settings to prioritize execution time, energy, or temperature minimization during optimization. The priority settings allowed TaPT to trade off the non-prioritized optimization goals in favor of the prioritized optimization goal, thus adhering the optimization to the design constraints. Furthermore, TaPT was computationally simple and imposed no additional hardware overhead.

To evaluate TaPT's optimization potential, we modeled an embedded processor architecture, similar to the ARM Cortex A9 [3], using GEM5. The processor contained configurable L1 instruction and data caches with cache sizes, line sizes, and associativities ranging from 8 to 32 Kbyte, 16 to 64 byte, and 1- to 4-way, respectively, in power-of-two increments. The processor offered seven clock frequencies ranging from 800 MHz to 2 GHz in 200 MHz increments. We used the largest parameter values as the base configuration, representing a modern-day non-configurable embedded system's microprocessor. We used Hotspot 5.0 [13] to measure the temperature using a floorplan and silicon chip area similar to the ARM Cortex A9 processor, and simulated an embedded system without cooling mechanisms, such as heat sink and/or spreader. We used eighteen benchmarks from the EEMBC [10] Automotive and Mibench [7] suites to model a variety of real-world embedded system applications.

Fig. 4 depicts the execution time, energy, EDP, and temperature of the best configurations as determined by TaPT normalized to the base system configuration. For brevity, we only show results for energy and temperature prioritization. Fig. 4 (a) shows that when energy prioritization was specified, TaPT achieved average EDP, energy, execution time, and temperature savings of 34%, 31%, 4%, and 20%, respectively. For temperature prioritization, we specified a 65^{o}C temperature threshold to simulate a highly temperature-constrained embedded system. Fig. 4 (b) shows that when temperature prioritization was specified, TaPT achieved energy and temperature savings of 13% and 25%, respectively. However, the execution time and EDP *increased* by 39% and 22%, respectively, due to the stringent temperature constraints.



(a)



(b)

Figure 4. Execution time, energy, EDP, and temperature normalized to the base configuration for priority settings (a) energy, and (b) temperature

Compared to exhaustive search, TaPT reduced the exploration time by 96% on average.

These results reveal the extent to which some optimization goals may be adversely affected by stringent design constraints. However, the results also show TaPT's ability to trade off optimization goals in order to adhere to design constraints.

IV. Potential PDM Usage Scenarios

To motivate future research, we have identified potential scenarios in which PDM can be used, and we intend to investigate these usage scenarios for future work.

Similar to the scenarios presented in this paper, PDM can be used for fast configuration of other configurable hardware and multi-objective optimization scenarios, including optimizing the issue logic, multi-level caches, etc. To adapt PDM to other configurable hardware, the application characteristics used to evaluate the phase distance must closely relate to the configurable hardware in order to achieve optimization goals.

Furthermore, PDM can be used for scheduling in heterogeneous multicore systems, where the best application/phase-to-core schedules must be determined. PDM can evaluate the phase distance between a previously executed phase and a new phase, and use the correlation between both phases to predict the best core schedule with respect to the optimization goals for the new phase.

Finally, PDM can be used to speed up simulations in computer architecture research, where the best configurations must be rapidly determined for various architectures and compared to a base configuration to evaluate the new architectures' adherence to the optimization goals.

V. Conclusions

This work presents phase distance mapping (PDM), a low-overhead and dynamic analytical approach to dynamic phase-based optimization of embedded systems. PDM significantly reduces optimization overhead and designer effort by dynamically correlating a known phase's characteristics and best configuration with a new phase's characteristics to determine the new phase's best configuration. PDM is suitable for various optimization scenarios and general purpose embedded systems where the system applications are not known a priori.

Acknowledgment

This work was supported by the National Science Foundation (CNS-0953447). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

References

[1] T. Adegbija, A. Gordon-Ross, and A. Munir, "Phase distance mapping: a phase-based cache tuning methodology for embedded systems," Springer Design Automation for Embedded Systems (DAEM), January 2014.

[2] T. Adegbija and A. Gordon-Ross, "Thermal-aware phase-based tuning of embedded systems," ACM Great Lakes Symposium on VLSI (GLSVLSI), May 2014.

[3] ARM, http://www.arm.com/products/processors/cortex-a/cortex-a9.php

[4] N. Binkert, et. al, " The gem5 simulator," Computer Architecture News, May 2011.

[5] S. Gal-On and M. Levy, "Measuring multicore performance," Computer, November 2008.

[6] A. Gordon-Ross, F. Vahid, and N. Dutt, "Automatic tuning of two-level caches to embedded applications," Design Automation and Test in Europe (DATE), February 2004.

[7] M. R. Guthausch et al., "Mibench: a free, commercially representative embedded benchmark suite," IEEE Workshop on Workload Characterization, 2001.

[8] MIPS32 M14K. http://www.mips.com/products/cores/32-64-bit-cores/mips32-m14k/ Accessed 26 July 2013

[9] A. Munir, A. Gordon-Ross, S. Lysecky, and R. Lysecky, "A one-shot dynamic optimization methodology for wireless sensor networks," International Conference on Mobile and Ubiquitous Computing (UBICOMM), October 2010.

[10] J. Poovey, M. Levy, and S. Gal-On, "A benchmark characterization of the EEMBC benchmark suite," International Symposium on Microarchitecture, October 2009.

[11] T. Sherwood, S. Sair, and B. Calder, "Phase tracking and prediction," 30th International Symposium on Computer Architecture (ISCA), May 2003.

[12] K. Skadron, "Hybrid architectural dynamic thermal management," Design Automation and Test in Europe (DATE), February 2004.

[13] K. Skadron, et al., "Temperature-aware microarchitecture: modeling and implementation," Transactions on Architecture and Code Optimization, March 2004.

[14] Synopsys Design Compiler, Synopsis Inc. www.synopsys.com

[15] C. Zhang, F. Vahid, and W. Najjar, "A highly-configurable cache architecture for embedded systems," 30th International Symposium on Computer Architecture (ISCA), May 2003.

[16] E. Zitler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," Swiss Federal Institute of Technology, Dept. of Electrical Engineering, Technical Report 103, 2001.