Exploring the Tradeoffs of Configurability and Heterogeneity in Multicore Embedded Systems

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

Abstract—**Embedded systems, such as smartphones, have become ubiquitous due to the versatility of these devices for various applications, which have varying application resource requirements. Due to these variances, system resources should be specialized to the executing applications' resource requirements in order to adhere to design/optimization goals (e.g., reduced energy consumption, improved performance, etc.). In multicore systems, core heterogeneity and/or configurability affords specialization, however, this design choice exacerbates design challenges and complexity due to an embedded system's stringent design constraints. We evaluate the benefits and tradeoffs of heterogeneous and configurable cores as compared to homogeneous cores for design-constrained multicore embedded systems. Our studies provide valuable insights and guidelines on design choices and show that combining heterogeneity and configurability provides unique opportunities for fine-grained specialization.**

Keywords-heterogeneous cores; homogeneous cores; configurable architecture; low-power architecture; multicore

I. Introduction and Motivation

Multicore architectures are becoming prevalent in ubiquitous embedded systems (e.g., automotive systems, consumer electronics, smartphones, etc.) as an alternative to single-core architectures for achieving design/optimization goals, such as reducing cost, energy consumption, time to market, and/or increasing performance. However, this single- to multicore architecture shift significantly increases design challenges and complexity when coupled with an embedded system's stringent design constraints and resource availability (e.g., energy, power, area, real-time deadlines, size, etc.), which affords challenging design decisions. Additionally, designers must consider the applications' varying resource requirements during execution [12], thus necessitating *specialization* to the applications' unique requirements in order to adhere to design goals, which is becoming increasingly difficult to achieve using traditional homogeneous cores due to widely disparate application requirements.

One method to achieve multicore system specialization is by using disparate—heterogeneous—cores with varying characteristics (e.g., processor family/version, performance, die area, etc.). For example, the Open Multimedia Applications Platform (OMAP) chip contains a microprocessor core (ARM926) and a digital signal processor (DSP) coprocessor core (TMS320C55X) [20]. Even though the designer can select different cores to meet the varying applications' requirements, the design space is limited to the number of core combinations.

To increase adherence to design goals, configurable cores have configurable parameters (e.g., cache size, core frequency and/or voltage, etc.), whose values/configurations can be determined statically at design time or dynamically during runtime. In a configurable homogeneous core system [8], the cores have identical characteristics, but the cores' configurations are specialized to the applications' requirements. Depending on the level of configurability, the cores can have either the same or different configurations. Configurable core systems have large design spaces that consist of all combinations of parameter values, and thus provide more fine-grained specialization.

However, fine-grained specialization exacerbates design complexity, challenges, and decisions. Selecting between a heterogeneous or configurable homogeneous core system affects the level of design goal adherence, but this selection affects competing design goals, including design complexity, energy consumption, performance, runtime overhead, time to market, etc. For example, in heterogeneous core systems, designers have limited configuration options—lower design complexity—but must carefully select the most appropriate cores, and thus adherence to design goals may be limited due to the coarse-grained design space.

Alternatively, configurable homogenous cores may adhere more closely to design goals, but significantly increase design complexity, and time to market, since the cores' configurations must be tuned. Tuning evaluates an application's requirements and determines the best configuration for design goal adherence, but incurs overhead in terms of time, performance and/or energy overhead.

Concomitant to system specialization is application scheduling, which determines the most appropriate core to execute the application on [16]. Scheduling decisions, whether made a priori or at runtime, must consider the cores' characteristics and configurations since this information can significantly affect the system's adherence to design goals. Heterogeneous cores offer less specialization, thus designers must carefully select the cores to maximize the potential for design goal adherence. Configurable homogenous cores alleviate the core selection challenge and increase design goal adherence potential, but complicate scheduling decisions due to a larger design space.

Previous research for general purpose systems showed that heterogeneous and configurable homogeneous cores improve energy consumption and performance as compared to homogeneous cores [11][12], however, there is little research with respect to the unique, and highly constrained, embedded system design goals. Prior scheduling and design space exploration methods [10] for embedded systems did not compare heterogeneity versus homogeneity. Furthermore, to the best of our knowledge, no prior work studied the tradeoffs (with respect to energy consumption, time to market, runtime overhead, etc.) between using heterogeneous and configurable homogeneous cores for achieving specialization, or whether configurable homogeneous cores provide an appreciable increase in design goal adherence to offset the increase in design and scheduling complexity and tuning overhead (e.g., energy, power, performance, etc.).

In this paper, we present an empirical comparison of the tradeoffs between heterogeneous and configurable homogeneous core embedded systems with respect to the energy delay product (EDP) and cache configuration and core frequency specialization. We also evaluate the EDP savings attained by using configurable heterogeneous cores, which leverage the advantages of both heterogeneity and configurability. Our evaluations provide valuable insights and guidelines to assist designers with design challenges and decisions.

The remainder of this paper is organized as follows. Section II presents the related work, Section III identifies the design challenges and studied architectures, Section IV discusses our experimental methodology, and our results are presented in Section V. Section VI posits multicore design suggestions and Section VII concludes the paper and discusses future work directions.

II. Related Work

Kumar et al. [12] proposed a single-instruction set architecture (ISA) heterogeneous multicore system to reduce power in general purpose computers, where each core provided different performance versus power tradeoffs. In [11], Kumar et al. showed that heterogeneous core systems provided power and throughput advantages for applications with varying execution requirements. Balakrishnan et al. [3] investigated the effects of data input size for recurring applications verses core scheduling, and concluded that heterogeneous core systems were beneficial for performance when core scheduling decisions considered the input size/characteristics. Grochowski et al. [9] studied heterogeneous cores with respect to energy and throughput improvements. However, all of these works focused on evaluating heterogeneity in general purpose computers, where throughput typically outweighs energy consumption.

Configurable core systems can be composed of any core with any configurable parameter(s). Zhang et al. [28] showed that applications have varying cache requirements and proposed a configurable cache architecture that determined Pareto optimal cache parameter values trading off energy consumption and performance, showing average energy savings of 40% as compared to a conventional, non-configurable cache. Gordon-Ross et al. [8] showed that configuring the cache to a particular application's requirements reduced memory access energy by 62% with performance improvements in most cases. Semeraro et al. [25] showed that dynamically scaling core voltage using multiple clock domains resulted in EDP savings of 20%. Albonesi [2] presented complexity-adaptive processors where the instruction per cycle (IPC)/clock rate tradeoff could be dynamically altered to match the application's changing requirements, and reduced time per instruction by an average of 9%.

Whereas prior works clearly motivate the benefits of heterogeneous cores and configurable cores, to the best of our knowledge, we are the first to investigate the tradeoffs between heterogeneous and configurable core systems for fine-grained specialization in embedded systems. Our studies and outcomes provide designers with valuable insights into design decisions when choosing an appropriate system configuration for specialization to the applications' requirements.

III. Design Challenges and Architectures

Since incorporating specialization into embedded system design imposes many daunting design challenges, this section details some of the challenges introduced when considering heterogeneity and configurability, and illustrates our evaluated system architectures.

A. *Heterogeneity*

In heterogeneous core systems, the cores' non-uniformity enables designers to statically select different cores that are suitable for different application requirements. If the applications are scheduled to the most suitable cores, performance and energy improvements are possible as compared to a homogenous core system. To provide a wide variety of suitability for diverse application requirements, most traditional heterogeneous core systems contain disparate cores [20]. However, large core disparity can introduce additional overheads and design challenges, especially if the cores have different ISAs, necessitating additional design time, area overheads, more complex scheduling, multiple binaries for each application, etc. Since leveraging cores with the same ISA, but different characteristics, eases system design while still offering specialization, we evaluate single-ISA systems, however, our fundamental tradeoff analysis is applicable to any heterogeneous core system with diverse core ISAs.

Since embedded systems typically have a large design space and several options for heterogeneity (e.g., ISA, core interconnect, memory hierarchy, etc.) one of the major challenges of heterogeneity is determining the key core characteristics that should differ in order to most closely adhere to design goals. The designer must evaluate the system and anticipated applications to select the appropriate cores, which places pressure on the time to market. Oftentimes this process is not straightforward at design time for general purpose embedded systems that execute a wide variety of applications (e.g., smartphones, tablets, etc.), and may require lengthy application evaluation/pre-analysis and design space exploration when the application(s) (or application domain(s)) is/are known a priori. Ideally, the cores would have enough diversity such that there exists a core that would be suitable for any application that could potentially execute on the system.

Scheduling further compounds the core selection challenge since the benefits of core diversity can only be exploited if the scheduler is aware of the cores' tradeoffs with respect to the applications' requirements. We evaluated the scheduling policy's criticality on EDP (Section V) and observed that naïve scheduling decisions (e.g., the scheduler does not consider core tradeoffs, and randomly schedules applications to arbitrary cores) can severely degrade EDP, even resulting in higher EDP than a simple homogeneous core system. Whereas the scheduler must effectively analyze application versus core tradeoffs, this analysis and scheduling must not impose excessive overhead [4]. Prior works showed that effective scheduling can be integrated into the operating system, thus avoiding hardware overhead [7].

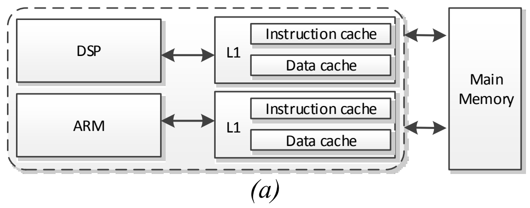
A simple, yet effective, approach to scheduling is the sampling-based method [4][12], which samples different application-to-core mappings at runtime and selects the best schedule based on design goals. This method introduces performance overhead due to periodic application migration across cores for application-to-core mapping evaluation, especially for systems with a large number of cores, and can incur significant overhead when executing an application on an unsuitable core. This overhead is less significant for heterogeneous core systems with replicated cores and in systems with persistent applications (e.g., smartphones), since the best schedule only needs to be determined once and can be reused for each subsequent application execution. Other prior works proposed more complex scheduling methods [4][7][10], however, due to sampling's simplicity, effectiveness, and ease of implementation, and thus appropriateness for embedded systems, we leverage sampling-based scheduling in our experiments, however, our fundamental tradeoff analysis could evaluate any scheduling method.

B. *Configurability*

System configurability is key to adhering to design goals, thus much research has explored configurability with respect to instruction set extensions [14], core voltage [25], issue queue [6], reorder buffer [21], etc. Since research shows large potential EDP savings when combining dynamically configurable caches and core frequency [18], in this work we focus on these parameters, however our fundamental tradeoff analysis is applicable to any configurable parameters.

When using multiple configurable parameters, the design space increases rapidly, especially for interdependent parameters, thus exacerbating design challenges due to potentially intractable design spaces and large tuning overhead. One major challenge when leveraging configurability is specialization granularity, which determines how often the configuration changes. Application-based tuning [28] uses a single configuration that represents the best configuration for the average run of the entire application. Phase-based tuning [8] achieves finer-grained specialization by changing the configuration during application execution based on the application's varying runtime requirements. Whereas phase-based tuning increases design goal adherence potential, phase-based tuning requires identifying phase changes (changes in requirements) and determining the best configuration for each phase, thus increasing tuning overhead.

Dynamically determining the best configurations without incurring significant tuning overhead is especially challenging for large design spaces and fine-grained specialization. Heuristics [8] significantly prune the design space and analytical models/methods [1] can directly determine the best configuration sans design space exploration, thus significantly



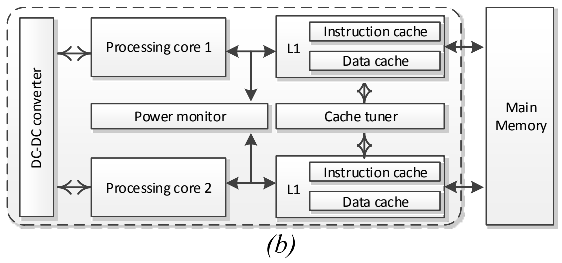


Figure 1. Layout of a sample (a) heterogeneous dual-core system and (b) configurable homogeneous dual-core system

reducing tuning overhead. Despite the overheads, dynamic configurability alleviates costly system/application pre-analysis by the designer, thus resulting in a shorter time to market as compared to heterogeneous core systems [24].

C. *Illustrative System Architectures*

Fig. 1 illustrates our evaluated dual-core system architectures, however, our evaluation methodology is applicable to any system with any arbitrary number of cores and any configurable parameters. Figure 1 (a) depicts a heterogeneous dual-core system with the following on-chip components: two processing cores with different clock frequencies and private level one (L1) instruction and data caches (iCache and dCache, respectively) with different cache configurations for each core. The clock frequencies and cache configurations are tuned at design time and remain static throughout the system's lifetime. Figure 1 (b) depicts a configurable homogeneous dual-core system with the following on-chip components: two identical processing cores with private configurable L1 instruction and data caches, and lightweight, low-overhead tuning hardware—a cache tuner and a DC-DC converter [18][28]. The cache tuner orchestrates dynamic cache tuning by changing the caches' configurations, evaluating and determining the best configurations, and fixing the system to run in those configurations. The DC-DC converter dynamically tunes the core frequency based on power measurements from the power monitor.

IV. EVALUATION METHODOLOGY AND EXPERIMENTAL SETUP

To quantify the EDP variances for heterogeneous, configurable homogeneous, and homogenous core systems, we modeled cache and core frequency configurations common to consumer embedded systems [10] (e.g., the Tegra 2 [26]) and evaluated dual-core systems, which are common in general purpose consumer embedded systems. Even though our experiments in this paper represent state-of-the-art embedded systems [26], our fundamental tradeoff analysis is applicable to future and/or more complex systems (e.g., n-core systems with multi-level caches) because the fundamental design goals and challenges are independent of these characteristics.

TABLE 1. DUAL-CORE SYSTEMS AND CONFIGURATIONS. THE CONFIGURABLE SYSTEM'S PARAMETER VALUES REPRESENT RANGES. THE HETEROGENEOUS SYSTEMS' PARAMETER VALUES REPRESENT CORES' VALUES AS CORE1/CORE2.

System	Cache size	Associativity	Line size	Clock frequency
Homogeneous	32 Kbyte	4 way	64 byte	2 GHz
Configurable	16 – 32 Kbyte	1 – 4 way	16 – 64 byte	1 – 2 GHz
Heterogeneous-1	16/32 Kbyte	4 way	64 byte	1/2 GHz
Heterogeneous-2	8/16 Kbyte	4 way	64 byte	800 MHz/1 GHz
Heterogeneous-3	8/32 Kbyte	4 way	64 byte	800 MHz/2 GHz

TABLE 2. TEST SCENARIOS

Name	Core descriptions
Test scenario 1	Naively-scheduled Heterogeneous-1
Test scenario 2	Optimally-scheduled Heterogeneous-1
Test scenario 3	Configurable homogeneous
Test scenario 4	Configurable heterogeneous

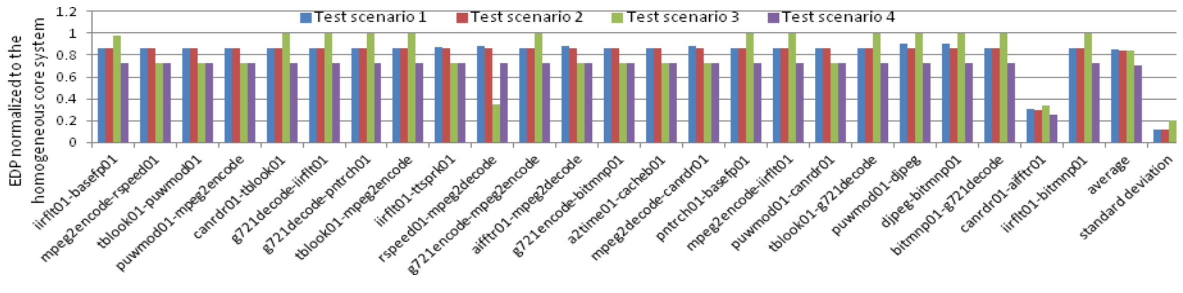


Figure 2. EDP normalized to the homogeneous core system (baseline of one) for the test scenarios in Table 2

Table 1 depicts our experimental dual-core systems and the systems' configurations, which represent actual embedded systems (e.g., Nokia Lumina 620 [19] and Motorola's Atrix 4G smartphones [17]). All cores had separate, private L1 instruction and data caches connected directly to off-chip main memory. Even though our evaluated systems could include a level two (L2) cache, our work evaluates the effects of L1 cache specialization, therefore we do not need to model the L2 cache, however, our work could be easily extended to include an L2 cache. The homogeneous core system served as our *base system* for comparison purposes. The configurable system represented the configurable homogeneous and configurable heterogeneous core systems, and offered cores with varying cache parameter values and operating frequencies (denoted as ranges). The configurable homogeneous core system's cores were tuned simultaneously to a single, homogenous configuration (the cores had the same configurations), resulting in a design space of 108 core configurations. The configurable heterogeneous core system's cores were tuned independently to heterogeneous configurations (the cores could have different configurations), resulting in a design space of 108^n configurations, where n is the number of cores.

In order to compare to static designer-selected cores, we evaluated three different heterogeneous core systems, denoted as Heterogeneous-1, -2, and -3. Based on empirical analysis, Heterogeneous-1 represented the best average configuration for all workloads (Section V), and thus served as the base heterogeneous core system. Heterogeneous-2 and -3 offer different core selection options for situations where designers cannot perform extensive design time analysis, and therefore the designer must make a "best guess" of the applications' requirements.

We exhaustively modeled and evaluated all configurations using GEM5 [5], which we modified to support heterogeneous cores, and McPAT [15] to calculate the systems' cores' EDPs in Joule seconds:

$$\begin{aligned} \text{EDP} &= \text{core\_power} * \text{running\_time}^2 \\ &= \text{core\_power} * (\text{total\_cycles}/\text{core\_frequency})^2 \qquad (1) \end{aligned}$$

where *core_power* includes the core's components, such as the network interface units (NIUs), peripheral component interconnect (PCI) controllers, etc., and the cache, and *total_cycles* is the number of cycles for a single workload execution.

In order to reduce the sensitivity of the results to a particular set of simulated workloads and model real-world embedded system applications, we created twenty-four multiprogrammed workload sets by selecting two random single-threaded benchmarks from seventeen EEMBC [22] Automotive benchmarks (the entire EEMBC Automotive benchmark suite could not be evaluated due to compilation errors) and six random MediaBench [13] benchmarks for image, video, and audio processing. To ensure that both cores executed the same number of cycles and both benchmarks executed at least once to completion, we looped the faster benchmark in each set. The benchmarks were always scheduled to a separate core during execution, such that both cores were active throughout execution. Since embedded systems typically execute small applications with relatively stable characteristics throughout execution, we leveraged application-based tuning in our experiments.

Table 2 depicts our test scenarios. We determined optimal scheduling using sampling such that the lowest EDP schedule represented the best case. Since naïve scheduling randomly schedules applications to cores, we used the highest EDP schedule in order to compare with the worst case schedule. In practice, naïve scheduling would not necessarily achieve the worst case EDP, but comparing to the worse case provides a clearer picture of core selection tradeoffs. We determined the best configurations for the configurable homogeneous and configurable heterogeneous cores using exhaustive search for all of the workloads.

V. RESULTS

Fig. 2 depicts the EDP for the test scenarios in Table 2 normalized to the homogeneous core system (baseline of one) for all experimental workloads, denoted as the x-axis benchmark combinations, and the average and standard

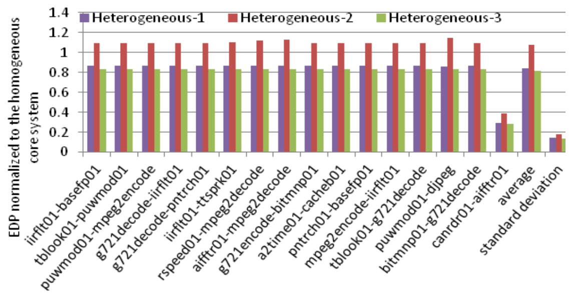


Figure 3. EDP of the heterogeneous systems in Table 1 normalized to the homogeneous core system (baseline of one)

deviation across all workloads. For brevity, we show comparisons with only the base heterogeneous core system, Heterogeneous-1.

Test scenarios 1 and 2: Compared to the homogeneous core system, the naively-scheduled Heterogeneous-1 system achieved similar EDP savings for most workloads, averaging 15% over all workloads, ranging from 9% for *djpeg-bitmnp01* and 69% for *canrdr01-aifftr01*, with a standard deviation of 12%.

The optimally-scheduled Heterogeneous-1 system achieved average EDP savings of 16% over the homogeneous core system, with savings increasing to 13% and 70% for *djpeg-bitmap* and *canrdr01-aifftr01,* respectively. This optimally-scheduled system performed only marginally better than the naïvely-scheduled system, 1% on average over all of the workloads, with differences as high as 4%, but *all* workloads benefited from heterogeneity regardless of scheduling. The EDP savings for these systems were similar because both cores' configurations represented good average configurations for all of the benchmarks (i.e., the cores were averagely-suited for all of the benchmarks), thus neither core resulted in prohibitively high EDP. However, we point out that we exhaustively evaluated our benchmark suite to determine the best average heterogeneous cores' configurations, and in practice, a designer would have to expend considerable design time effort to replicate these results. This comparison shows that scheduling decisions have little effect if the cores' configurations are carefully selected to match the applications' requirements.

Therefore, given limited time to market and potentially unknown applications, we evaluated heterogeneous core systems more representative of actual designer-selected cores. Figure 3 depicts the EDP of the optimally-scheduled Heterogeneous-1, -2, and -3 systems (Table 1) normalized to the homogeneous core system. Heterogeneous-2 *increased* the EDP by 7% on average as compared to the homogeneous core system, and Heterogeneous-3 and -1 decreased the EDP by 19% and 16% on average, respectively. Additional results, not shown for brevity, also showed that scheduling had a significant impact on EDP for these systems, with Heterogeneous-2 and -3 showing EDP *increases* of 65% and 26% on average as compared to the homogeneous core system, respectively, as compared to Heterogeneous-1's 15% decrease in EDP.

Test scenario 3: The configurable homogeneous core system revealed similar EDP savings as the optimally-scheduled Heterogeneous-1 system, with average EDP savings of 16%, ranging from no savings for some workloads to 66% for *canrdr01-aifftr01*, as compared to the homogeneous core system. Even though the configurable homogeneous cores afforded a larger design space, and potentially finer-grained specialization than Heterogeneous-1, since the cores could only select a *single* (same) average best configuration for all the benchmarks, the specialization granularity and EDP savings were reduced.

Test scenario 4: The configurable heterogeneous core system achieved the highest overall average EDP savings of 29% over the homogeneous core system, ranging from 27% to 75% for *djpeg-bitmnp01* and *canrdr01-aifftr01*, respectively, with a standard deviation of 10%. Compared to the configurable homogeneous core system and the optimally-scheduled Heterogeneous-1, -2, and -3 systems, the configurable heterogeneous core system achieved average EDP savings of 11%, 16%, 34%, and 13%, respectively. The configurable heterogeneous core system achieved the maximum EDP savings because each core was tuned independently for each application's requirements, thus affording the finest-grained specialization and the largest EDP savings. We note that these results do not include runtime tuning overhead, but prior work showed that these overheads can be minimal [23].

VI. Multicore Specialization Guidelines

Previous work established that two cores introduce sufficient heterogeneity to attain appreciable EDP reductions [12]. In this paper, we significantly extend prior evaluations by studying the tradeoffs between core diversity and scheduling, and the benefits of incorporating configurability into heterogeneous core systems. Based on our findings, we suggest practical multicore specialization design guidelines intended to aid embedded system designers in selecting appropriate core characteristic and configuration diversity to meet design goals.

Our findings revealed a clear tradeoff between scheduling efficiency/effort and core diversity. Increased core diversity with a good balance of characteristics/configurations (e.g., Heterogeneous-2, as opposed to Heterogeneous-1 or Heterogeneous-3) enhances the benefits of heterogeneity by reducing the EDP, however, to realize these benefits, the scheduling policy must be very effective. With a less effective (i.e., naïve) scheduling policy, less core diversity is advantageous, such as using Heterogeneous-1, which contained good average configurations for the executing applications, such that either core will reveal good average savings, but neither core will reveal the largest savings. Therefore, designers can adjust their core selection efforts based on the system's scheduling policy

Maximum EDP savings is achieved using configurable heterogeneous cores where the cores can be independently tuned and the larger design space reveals greater EDP savings potential. Based on previous work [27], we conjecture that the design space can be significantly reduced with similar EDP savings by using cores with different configuration subsets. Each subset would be specialized to a different application domain's requirements, and applications from a particular domain would be scheduled to the appropriate core. Tuning would therefore only evaluate the reduced design space available on that core, which may be a fraction of the total, unsubsetted design space (only three/four in [27]), thus significantly reducing tuning overhead, however, the tradeoff is extensive design time analysis.

VII. Conclusions and Future Work

Configurable and/or heterogeneous cores specialize a system's configurations to the applications' execution resource requirements to adhere to design goals, however, this specialization exacerbates design complexity, challenges, and decisions. System core selection (e.g., homogenous, heterogeneous, or configurable homogeneous/heterogeneous cores) affects the level of design goal adherence and affects competing design goals. In order to assist designers in core selection, we evaluated and empirically quantified the benefits and tradeoffs of heterogeneous and configurable core systems as compared to homogeneous core systems with respect to cache configuration and core frequency specialization. We provided insights and guidelines for designers and showed that the best energy delay product (EDP) savings can be achieved by using configurable heterogeneous cores, which leverage the advantages of both configurability and heterogeneity. However, since configurable heterogeneous cores result in exponentially large design spaces, our future work will explore and evaluate the impact of reducing the configurable heterogeneous cores' design space by designing heterogeneous core systems with different configuration subsets in each core, where each subset is specialized to a different application domain.

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, "Dynamic phase-based tuning for embedded systems using phase distance mapping," International Conference on Computer Design, October 2012, pp. 284-290.

[2] D. Albonesi, "Dynamic IPC/clock rate optimization," Internation Symposium on Computer Architecture, July 1998, pp. 282-292.

[3] S. Balakrishnan, R. Rajwar, M. Upton, and K. Lai, "The impact of performance asymmetry in emerging multicore architectures," International Symposium on Computer Architecture, June 2005, pp. 506-517.

[4] M. Becchi and P. Crowley, "Dynamic thread assignment on heterogeneous multiprocessor architectures," Computing Frontiers, 2006, pp. 29-40.

[5] N. Binkert, B. Beckmann, G. Black, S. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. Hill, and D. Wood, "The gem5 simulator," Computer Architecture News, May 2011, pp. 1-7.

[6] D. Folegnani and A. Gonzalez, "Energy-efficient issue logic," Internation Symposium on Computer Architecture, July 2001, pp. 230-239.

[7] S. Ghiasi, T. Keller, and F. Rawsom, "Scheduling for heterogeneous processors in server systems," Proc. Computing Frontiers, ACM Press, 2005, pp. 199-210.

[8] A. Gordon-Ross and F. Vahid, "A self-tuning configurable cache," Design Automation Conference, July 2007, pp. 234-237.

[9] E. Grochowski, R. Ronen, J. Shen, and W. Wang, "Best of both latency and throughput," International Conference on Computer Design, October 2004, pp. 236-243.

[10] K. Kim, D. Kim, C. Park, "Real-time scheduling in heterogeneous dual-core architectures," International Conference on Parallel and Distributed Systems, 2006, pp. 91-96.

[11] R. Kumar, D. Tullsen, N. Jouppi, P. Ranganathan, "Heterogeneous chip multiprocessors," Computer, November 2005, pp. 32-38.

[12] R. Kumar, D. Tullsen, P. Ranganathan, N. Jouppi, and K. Farkas., "Single-ISA heterogeneous multi-core architectures: the potential for processor power reduction," International Symposium on Microarchitecture, December 2003, pp. 81-92.

[13] C. Lee, M. Potkonjak, and W. Mangione-Smith, "MediaBench: a tool for evaluating and synthesizing multimedia and communications systems," International Symposium on Microarchitecture, December 1997, pp. 330-335.

[14] R. Leupers, K. Karuri, S. Kraemer, and M. Pandey, "A design flow for configurable embedded processors based on optimized instruction set extension synthesis," Design, Automation and Test in Europe, 2006.

[15] S. Li, J. Ahn, R. Strong, J. Brockman, and D. Tullsen, N. Jouppi, "McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures," International Symposium on Microarchitecture, December 2009, pp. 469-480.

[16] J. Luo and N. Jha, "Battery-aware static scheduling for distributed real-time embedded systems," Design Automation Conference, 2001, pp. 444-449.

[17] Motorola Atrix 4G. http://www.motorola.com/us/consumers/Motorola-ATRIX-4G/72112,en_US,pd.html [retrieved: April, 2013]

[18] A. Nacul and T. Givargis, "Dynamic voltage and cache reconfiguration for low power," Design, Automation, and Test in Europe, February 2004, pp. 1376-1377.

[19] Nokia Lumina 620. http://www.nokia.com/global/products/phone/lumia620/ [retrieved: April, 2013]

[20] OMAP1710 processor architecture and features: http://focus.ti.com/general/docs/wtbu/wtbuproductcontent.tsp?templateId=6123&navigationId=11991&path=templatedata/cm/product/data/omap_1710 [retrieved: April, 2013]

[21] D Ponomarev, G. Kucuk, and K. Ghose, "Reducing power requirements of instruction scheduling through dynamic allocation of multiple datapath resources," International Symposium on Microarchitecture, December 2001, pp. 90-101.

[22] J. Poovey, M. Levy, and S. Gal-On, "A benchmark characterization of the EEMBC benchmark suite," International Symposium on Microarchitecture, December 2009, pp. 18-29.

[23] M. Rawlins and A. Gordon-Ross, "An application classification guided cache tuning heuristic for multi-core architectures," Asia South Pacific Design Automation Cconference, 2012, pp. 23-28.

[24] B. Ristau, T. Limberg, and G. Fettweis, "A mapping framework for guided design space exploration of heterogeneous MP-SoCs," Design, Automation and Test in Europe, March 2008, pp. 780-783.

[25] G. Semeraro, G. Magklis, R. Balasubramonian, D. Albonesi, S. Dwarkadas, H. Dwarkadas, and M. Scott, "Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling," High Performance Computer Architecture, February 2002, pp. 29-40.

[26] Tegra 2 super chip processors. http://www.nvidia.com/object/tegra-superchip.html [retrieved: April, 2013]

[27] P. Viana, A. Gordon-Ross, E. Keogh, E. Barros, and F. Vahid, "Configurable cache subsetting for fast cache tuning," Design Automation Conference, July 2006, pp. 695-700.

[28] C. Zhang, F. Vahid and W. Najjar, "A highly-configurable cache architecture for embedded systems," 30th Annual International Symposium on Computer Architecture, May 2003, pp. 136-146.