ElasticCore: A Dynamic Heterogeneous Platform With Joint Core and Voltage/Frequency Scaling

Mohammad Khavari Tavana, *Student Member, IEEE*, Mohammad Hossein Hajkazemi, Divya Pathak, Ioannis Savidis, and Houman Homayoun

***Abstract*—Heterogeneous architectures have emerged as a promising solution to address the dark silicon challenge by providing customized cores for each running application. To harness the power of heterogeneity, a critical challenge is simultaneously fine-tuning several parameters at the application, architecture, system, as well as circuit levels for heterogeneous architectures that improve the energy-efficiency envelope. To address this challenge, an ElasticCore platform is described where core resources along with the operating voltage and frequency settings are scaled to match the application behavior at run-time. A quantile linear regression model for power and performance prediction is used to guide the adaptation of the core resources, along with the operating voltage and frequency, to improve the energy efficiency. In addition, the dynamically scalable partitions of the ElasticCore are powered with multiple on-chip voltage regulators with high-power conversion efficiency that are able to realize fast dynamic voltage/frequency scaling. The results indicate that ElasticCore predicts application power and performance behavior with a small error at run-time across all studied benchmarks and achieves, on average close to 93% energy efficiency, as compared to an architecture with the Oracle power and performance predictor.**

***Index Terms*—Dynamic voltage scaling, energy efficiency, heterogeneous systems, reconfigurable architectures.**

I. Introduction

MOORE'S law scaling continues to provide increased transistor counts for each successive processor generation. In recent generations, these transistors are primarily devoted to increased core counts. However, the increase in the number of transistors is not accompanied by an increased power budget. Therefore, the "dark silicon era" [1] has arrived, where more transistors are available on an integrated circuit than can be possibly turned ON at any given time. Dark silicon requires a re-evaluation of the tradeoffs between area (transistor count) and power, as transistors are all but free, while power becomes critically important. As a consequence, both academic and industrial studies are focused on the development of new software and hardware paradigms and novel architectures to sustain Moore's law in the presence of dark silicon. One traditional approach is scaling voltage and frequency [i.e., dynamic voltage/frequency scaling (DVFS)] depending on the application demand. The power dissipation reduces quadratically and linearly when reducing the voltage and frequency, respectively. Another approach that effectively improves the energy efficiency of computation is the use of heterogeneous processing cores. Heterogeneous multicore systems are comprised of cores with varying architectural features as well as power and performance characteristics. Each core is specialized and, therefore, efficient for executing one type of application [2], or an aggregation of different generations of cores with the same instruction set architecture (ISA) is implemented [3], [4]. In heterogeneous systems, a given application is executed on the most efficient core based on system optimization objectives (power, energy, energy delay product, and so on). Whenever the execution phase of the application changes, the thread is migrated to the heterogeneous core most fit for the application. Traditionally, the best core is selected based on a small sampling of applications on each core [3], [5]. However, the sampling and migration of threads among cores imposes a timing penalty to the system. Migration requires the transfer of the architectural state of the current core to the selected core. In addition, the overheads imposed by warming up branch predictor and memory subsystems impose extra time overhead to the system. Therefore, to alleviate the various overheads, the decision is generally made at the granularity of the operating system time slice (i.e., tens to hundreds of millions of instructions). In the ARM big.LITTLE [4] platform, the decision is made at the operating system level such that, based on the feedback of the core utilization, the operating system decides to change the voltage/frequency or the core type. Voltage/frequency scaling incurs a time penalty of a few tens of microseconds [6], [7]. In a big.LITTLE system, the time overhead of task migration from an A15 to A7 and the reverse from an A7 to A15 is up to 3.75 and 2.10 ms, respectively [8]. Write back of dirty cache lines and the warming of cache and other state variables impose additional overheads, which limits the migration to fine-grained execution.

The ElasticCore reconfigures resources and the voltage/frequency dynamically at runtime with fine granularity and small overhead based on the demand of the applications.

Manuscript received March 4, 2017; revised June 8, 2017 and August 8, 2017; accepted September 16, 2017. Date of publication November 16, 2017; date of current version January 19, 2018. This work was supported in part by the National Science Foundation under Grant CNS 1526913. *(Corresponding author: Mohammad Khavari Tavana.)*

M. Khavari Tavana and M. H. Hajkazemi were with George Mason University, Fairfax, VA 22030-4422 USA. They are now with the Department of Electrical and Computer Engineering, Northeastern University, Boston, MA 02115 USA (e-mail: mkhavaritavana@ece.neu.edu; hajkazemi@ece.neu.edu).

D. Pathak and I. Savidis are with the Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA 19104 USA.

H. Homayoun is with the Department of Electrical and Computer Engineering, George Mason University, Fairfax, VA 22030-4422 USA.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2017.2759219

The primary enabling components that realize fine grain adaptation with high efficiency are the voltage regulators (VRs) and power delivery system of the proposed platform. A suitable power delivery system based on available state-of-the-art on-chip VR topologies is designed. The effect of on-chip and off-chip VRs on the total energy efficiency of the system is evaluated. In addition, the proposed platform consists of dynamically scaling the bandwidth as well as the capacity. Different tradeoffs in power and performance are achieved by reducing or expanding the size of various resources to construct cores of different sizes and configurations such as big, medium, little, or tiny. The tradeoff for each core size is highly affected by the operating voltage and frequency. Therefore, a joint core and DVFS optimization is required, particularly where the application performance is sensitive to frequency. Joint DVFS and core scaling are employed during different phases of the execution of an application. In particular, it is shown that increasing bandwidth and capacity reduces the sensitivity of an application to frequency. For example, while a core configuration with a higher bandwidth and capacity dissipates more power, there is significant opportunity to use DVFS for reducing power with a minor performance loss. Optimization of energy efficiency is explored based on core and voltage/frequency scaling. Note that with the reduction of the voltage, the frequency is reduced accordingly for the correct operation of the underlying circuit. In the remainder of this paper, the reference to scaling of frequency or voltage implies the simultaneous scaling of both parameters.

The basic idea of ElasticCore was proposed in [9]. The main contributions of [9] are as follows.

1) The performance and power sensitivity of various standard benchmarks for different core sizes (big, medium, little, and tiny) and frequencies are investigated. The performance of standard benchmarks in terms of execution time is shown to be a linear function of frequency while no clear trend is observed as a function of core size.
2) The impact of core size on the frequency sensitivity of applications is explored. It is shown that modifying the core size changes the sensitivity of an application to the frequency.
3) Based on the characterization results, an ordinary least-squares linear regression model (OLSLRM) is used to estimate the power and performance of an application at run-time. Based on performance counters' data captured at run-time for a small monitoring interval of the application, regression models predict the energy efficiency of an application for various core sizes and operating voltages/frequencies. The results are compared with an oracle predictor when used to guide the scaling of the core size and the operating voltage/frequency to maximize efficiency.
4) ElasticCore is proposed, which is a platform capable of scaling resources including bandwidth, capacity, voltage, and frequency based on the application performance requirements at run-time while improving energy efficiency. The ElasticCore platform eliminates the need of migration that is required in big.LITTLE like architectures (i.e., two separate cores with diverse power and performance characteristics) and outperforms the big.LITTLE architecture by enabling fine-grained resource scaling.
5) Circuit design challenges to realize fast core and voltage/frequency scaling at run-time are examined. The ElasticCore is provisioned with multiple on-chip VRs (OCVRs) to not only scale the voltage quickly but also sustain high power conversion efficiency (PCE) over varying load. Based on these results, a two-tiered power delivery scheme is proposed.

This paper substantially extends [9] as follows.

1) In previous work, regression coefficients are trained based on a few million instructions of a given benchmark and used for power and performance estimation at different operating points. The coefficients of the linear model are customized for each application based on the profiling information. In other words, the approach is not general, cannot be applied to an unknown application, and requires application profiling to determine the coefficients of the model. In this paper, the limitations of dedicated linear models are eliminated by using a unified regression model for all applications, moving toward a more general-purpose platform.
2) An OLSLRM was proposed in [9] to guide resource adaptation. In this paper, two other methods [robust LRM (RLRM) and quantile LRM (QLRM)] are evaluated and the prediction accuracies are compared. It is shown that quantile linear regression significantly outperforms the LRM used in [9].
3) In this paper, the analyses and evaluations of ElasticCore are extended through extensive simulation; in particular, the impact of dynamic heterogeneity and DVFS on energy efficiency is investigated. The results indicate that exploiting only DVFS for energy efficiency is not an effective approach, whereas exploiting multiple heterogeneous cores (without DVFS) drastically improves energy efficiency drastically. This observation is in line with [10] and [11].

The rest of this paper is organized as follows. In Section II, related work is described. The motivation of the work, the sensitivity of the applications with respect to core size and frequency is provided in Section III. The ElasticCore architecture and the effectiveness of various LRM are described in Section IV. Evaluation and results are presented and discussed in Section V. Finally, Section VI concludes this paper.

II. Related Work on Heterogeneous Architectures

Heterogeneous architectures are classified into two groups: 1) static heterogeneous architectures and 2) dynamic heterogeneous architectures. A discussion of some of the heterogeneous architectures in each category is provided in this section. Additional information is described in a recent survey on heterogeneous systems and architectures [12].

ElasticCore is a dynamic heterogeneous architecture in the sense that the heterogeneity is within a core, and the core itself is capable of providing diverse performance/power tradeoffs.

A. Static Heterogeneous Architectures

Multicore systems in which the cores are diverse but the characteristics of each core are fixed at design time are described as statically heterogeneous. Examples of commercial products include the AMD Fusion APUs [2] in the high-performance domain and the TIOMAP 5 [13] in the embedded system domain. A static heterogeneous architecture enables efficient thread-to-core mapping and permits a change in the mapping across phases of execution through thread migration [14], [15]. Prior research has shown that the potential benefit of a static heterogeneous architecture is greater with fine-grained thread migration than with coarse-grain migration [16]. In [14], an Intel Xeon is integrated with an Atom processor. Code instrumentation is used at the function or loop level to schedule different phases of the application on each processor. However, the separate core and memory subsystems in static heterogeneous architectures incur power and performance overheads for application migration, which makes dynamic mapping ineffective for fine-grained migration [16].

The composite core proposed in [16] integrates heterogeneous cores to overcome the high migration overhead. Low-overhead thread migration is realized by pairing an in-order and an out-of-order μ*Engine* in the same core. The controller decides at run-time which phase of the application is more suited to execute on the out-of-order or the in-order portion of the core to reduce power with a minor effect on performance.

The ARM big.LITTLE architecture [4] integrates high-performance cores with smaller energy-efficient cores. Despite having the same ISA, the microarchitectures are significantly different with diverse power and performance characteristics. The popularity of static heterogeneity is evident from the many commodity products on the market that implement the big.LITTLE architecture. The Qualcomm Snapdragon 810 uses quad-core Cortex A57s and quad-core A53s, while the Snapdragon 808 uses quad-core A57s and dual-core A53s [17]. The Nvidia Tegra X1 integrates quad-core Cortex-A57s with quad-core Cortex-A53s [18]. The Samsung Exynos 5 Octa uses four Cortex-A15s along with four Cortex-A7s [19].

One of the most important decisions for heterogeneous architectures is application mapping. The challenge of dynamic thread mapping in static heterogeneous many-core systems is addressed in [20] and [21]. Prior research aimed to maximize performance under power constraints. Work presented in this paper differs as the first goal is a dynamic heterogeneous architecture where core size can be adapted at run-time, and the second goal is to maximize the energy efficiency by reducing the energy delay. It is important to note that the power and performance of an application on different cores at various frequencies must be known for proper mapping. Traditional designs suggest selecting the best core based on a small sampling of applications on each core [3]. Other techniques estimate core performance without running applications on a particular core type. A model for performance estimation on two core types (i.e., big and little cores) is provided in [16] and [22]. The complexity of application mapping on a heterogeneous architecture increases exponentially with an increasing number of core types and applications [21], [23]. In response, the solution proposed in this paper applies application mapping with minimal overhead. In this paper, an estimation engine is described that provides an estimation of the performance and power for four core types at four operating frequencies, for a total of 16 operating points. The estimation engine and on-chip VR enables ElasticCore to perform fine-grained adaptation with minimal overhead.

B. Dynamic Heterogeneous Architectures

Static heterogeneous architectures achieve heterogeneity through static or fixed cores. However, a particular heterogeneous design that is fixed may not match the needs of an arbitrary workload. In addition, the resources required for an application vary during different phases of the execution, but the core remains fixed in size, resulting migration overhead that further limits the benefits of heterogeneity. Dynamic heterogeneity is exploited in finer granularity in [24], where 3-D stacking is proposed as a means to share some of the structures among the different stacked cores that cause a performance bottleneck including the register file (RF) and load store queue (LSQ). The 3-D stacking, however, suffers from the high cost of integrating dies vertically and elevated temperature of operation [24]. Core Fusion [25] and TFlex [15] allow for the doubling or quadrupling of the size of the core by aggregating cores together to improve performance whenever the instruction-level parallelism (ILP) is high. Major challenges with Core Fusion and TFlex include data migration with changing core size and the additional pipeline latency imposed by the fused cores. Alternatively, large out-of-order cores are used in MorphCore [26] for workloads with high ILP while the architecture is reconfigurable at run-time to many in-order cores for workloads with high thread-level parallelism. While MorphCore is an architecture that switches between out-of-order and wide in-order operation, the focus of the work described in this paper is to dynamically adapt the out-of-order core window size (capacity), execution bandwidth, and frequency to the workload requirements at run-time to improve the energy efficiency. More recent work applies front end throttling and resource scaling in superscalar processors to tradeoff performance for energy [5]. The core periodically resizes to full resources and samples the instructions per cycle (IPC) to guarantee the performance. Sampling imposes overhead and scalability issues when the operating points increase. ElasticCore adapts the core size and frequency at the same time to optimize energy efficiency without the need for frequent sampling.

III. Motivation

In this section, the performance sensitivity of various standard benchmarks to frequency and core size is analyzed. The concept of dynamically scaling the core architecture with respect to the application behavior is also described. The aim is to design a platform which is capable of dynamically detecting

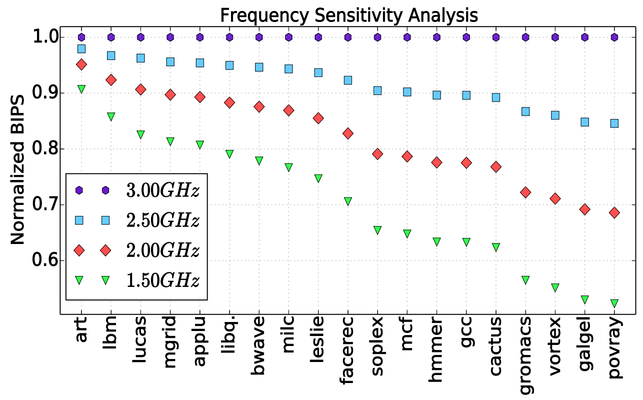

Fig. 1. Normalized performance of different benchmarks with respect to frequency. Note that all the values are normalized to a core with maximum frequency and resources.

the sensitivity of an application to both frequency and core size within a few kilo instructions, and then scale the frequency and core size with respect to the behavior of the application.

A. Application Sensitivity to Frequency

A wide range of applications with diverse behavior from SPEC2000 and SPEC2006 benchmarks are selected and simulated on various core configurations and across various frequencies. The normalized billion instructions per second (BIPS), used as a performance metric, for the studied benchmarks when the operating frequency is swept from 1.5 to 3 GHz with a step of 500 MHz is shown in Fig. 1. The sensitivity of the application to the operating frequency increases when moving from the left to the right in Fig. 1. For instance, the performance loss of *art* is less than 10% when reducing the frequency by half (far left). However, close to a 50% performance loss occurs when reducing the frequency by half when executing the *povray* benchmark (far right). Reducing the frequency slows the CPU–bound applications noticeably, while no significant impact is observed for memory-bound applications. Sensitivity of applications to frequency variation is observed and studied in previous work [27].

B. Application Sensitivity to Core Size

Applications exhibit a different performance behavior as a function of the size of the core resources. In this section, the performance sensitivity of the applications to the size of the core (capacity and bandwidth) is examined. For capacity, the LSQ, integer/floating point RF, reorder buffer (ROB), and integer/floating point instruction queue (IQ) are considered as these four units define the instruction window size of the processor. For bandwidth, the fetch, decode, issue, and commit width are considered. Four balanced cores with different capacity and bandwidth are modeled to reduce the design exploration space of the system. The detailed parameters for each of the configurations are listed in Table I.

The normalized BIPS for each application is shown in Fig. 2. The BIPS for four configurations with different core sizes is reported. It is important to note that the sensitivity to core size is not uniform across applications. For instance, *lbm* shows less than 0.1% performance loss when comparing a

TABLE I
RESOURCE SPECIFICATIONS FOR THE ELASTICCORE PLATFORM

Core voltage/frequency	
Frequency (GHz)	1.5, 2.0, 2.5, 3.0
Vdd (V)	0.750, 0.914, 1.132, 1.350
Bandwidth *(tiny, small, medium, big)*	
Decode/Issue/Commit width	1, 2, 3, 4
Fetch width	2, 4, 6, 8
Capacity *(tiny, small, medium, big)*	
IIQ/FIQ/LSQ size	16, 32, 48, 64
IREG/FREG size	24, 48, 64, 96
ROB size	32, 64, 96, 128
Memory subsystem	
L1 cache	32KB, 4-way
L2 cache	512KB, 8-way

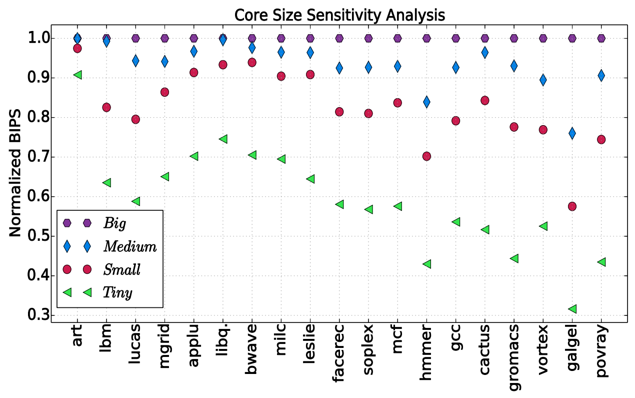

Fig. 2. Normalized performance of different benchmarks with respect to core size. Note that all the values are normalized to a core with maximum resources.

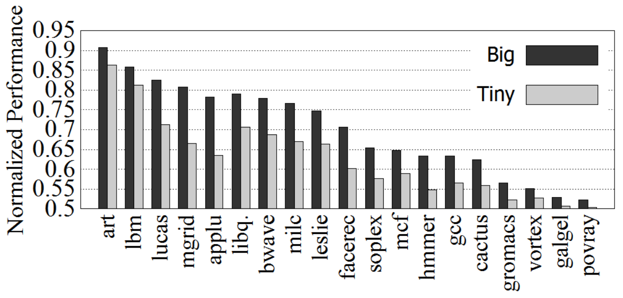

Fig. 3. Impact of core size on frequency sensitivity.

big- to-medium core size, whereas comparing big to small and tiny reveals a performance loss of 17% and 36%, respectively. On the other end of the spectrum, *galgel* shows continuous and uniform performance reduction when changing the core size from big to medium, small, and tiny. A 3x performance loss is observed when reducing the core size from big to tiny for the *galgel* benchmark.

C. Impact of Core Size on Frequency Sensitivity

Simulations indicate that the performance sensitivity of the application to frequency changes noticeably depending on the core size, as shown in Fig. 3. The normalized performance for big and tiny core size configurations when the application is executed at half the maximum frequency, as compared to the case when the application is executed at the maximum

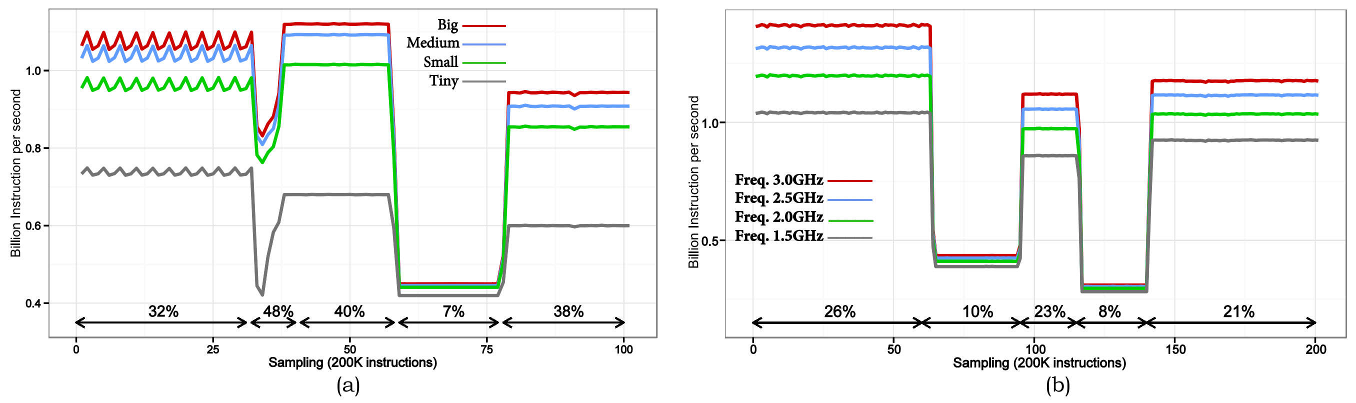

Fig. 4. Sensitivity of *applu* benchmark to (a) core size and (b) frequency at run-time over 20 and 40 million instructions, respectively. The maximum performance loss as a percentage is included for each phase.

frequency, is plotted in Fig. 3. For a number of benchmarks, when the core size varies, the impact on the sensitivity of an application to frequency is significant. However, for other benchmarks, less of an impact is observed. For instance, core size has the largest impact on frequency sensitivity for *mgrid* and the least impact on the *povray* benchmark. As core size potentially affects the frequency sensitivity of applications, the performance estimation engine must be capable of capturing the behavior of an application to properly select the most energy efficient core size and frequency.

D. Application Sensitivity at Run-Time

As shown in Figs. 1 and 2, applications exhibit different levels of sensitivity to frequency and core size. Over-provisioning resources to assure a core functions for the most demanding applications leads to inefficiency in energy usage. The many categories of applications that execute on different cores with various sizes highlight the need for a heterogeneous architecture for improved energy efficiency. Heterogeneous designs that reduce over-provisioning without necessarily sacrificing performance enable significant gains in performance per Watt. However, even within an application, the behavior (sensitivity to frequency and core size) changes at run-time. The BIPS trace for the *applu* benchmark is plotted for various core size and frequency values in Fig. 4(a) and (b), respectively. Even for a range of 20 million instructions, the variation in performance changes significantly from 7% to 48% when the core size is changed from big to tiny. A similar dynamic behavior is also observed when changing the frequency. As shown in Fig. 4(b), the different phases of an application experience a variation in the performance when the frequency is reduced to half. The main goal of ElasticCore is to detect the upcoming phase of an application at run-time, expand or contract resources (capacity and bandwidth), and set the frequency/voltage to match the core resources for the currently executing workload.

IV. ElasticCore Architecture

A. Microarchitecture Feasibility

Prior research has explored the microarchitecture and design modifications required to allow for resource adaptation in the pipeline, where components such as an RF, IQ, ROB, LSQ, fetch width, issue width, and commit width are dynamically resized [24], [28]–[31]. The overhead of microarchitecture as well as the power and area overheads to implement resource resizing is minimal [28]–[30].

Redesigning core components such as the RFs, ROB, IQ, and LSQ to implement adaptive resource resizing was studied in [29]. The RF and ROB are static random access memory (SRAM) structures. Due to the circular FIFO nature of the ROB, two pointers are required to dynamically adjust the size.

A modular architecture to dynamically resize both the ROB and RF is proposed in [28]. Structures such as the LSQ and IQ are designed as content addressable memory + SRAM-based architectures. The components hold the instructions until they are ready to issue. It is possible that two or more partitions of the IQ/LSQ are combined to form a larger partition without impacting cycle time [5], [24]. Prior research also considered bandwidth adaptation based on the ILP in the application [30], [31]. The ElasticCore exploits previous research to implement fine-grained resource adaptation.

Instruction dispatch is stalled when transitioning from more resources to less resources (e.g., a transition from a medium to a tiny core size), to allow for the completion of the current instructions. However, the core is completely stalled during voltage scaling. Stalling the instruction dispatch or stalling the core during voltage scaling incurs performance loss. Most instruction dispatch stalls incur a penalty of less than 100 cycles, which is negligible [5]. However, in this paper, assuming a conservative implementation, 1K cycles of overhead is assumed to account for the reconfiguration of the core. The voltage scaling overhead depends on the VR. If an on-chip VR is implemented, the performance overhead is in the range of a few tens of nanoseconds [6], [23].

Note that ElasticCore is designed for the maximum capacity and bandwidth (in this case, a pipeline width of 4 and window size of 128) to account for the worst case scenario where all resources are needed. The ElasticCore specifications are listed in Table I. The core includes four voltage/frequency settings. The memory subsystem remains unchanged; however, the core size (capacity and bandwidth) is resized to four different configurations.

B. Efficiency Metric

The efficiency is measured in BIPS^3/W, which is a metric for joint power and performance optimization for

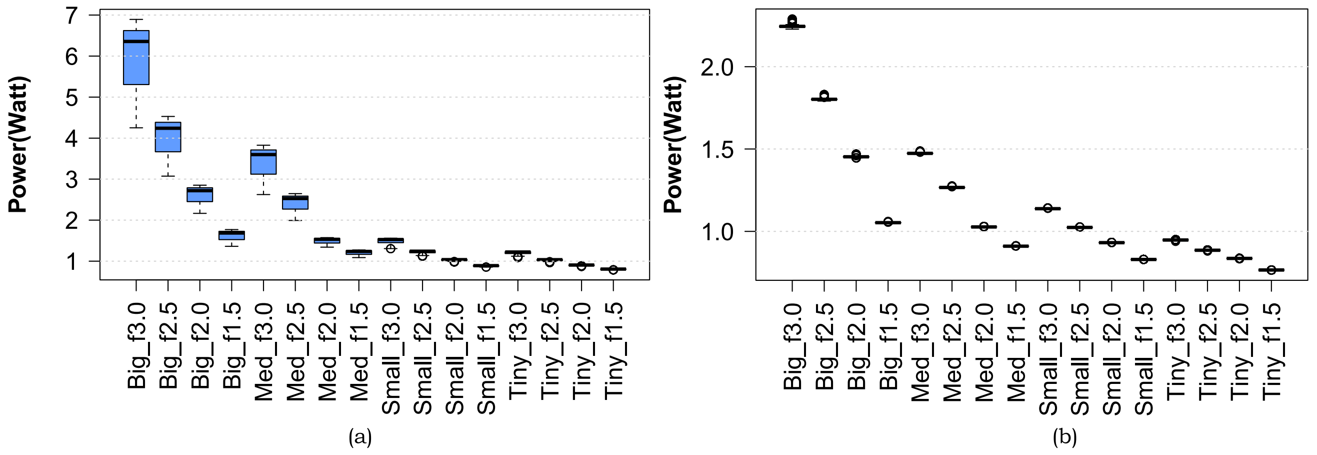

Fig. 5. Power dissipation of (a) *gcc* and (b) *lbm* benchmarks for different voltage/frequency pairs and core sizes.

high-performance processors [12], [24], [32]. The BIPS$^3/W$ is proportional to the inverse of the energy × delay^2 product (ED^{2}P). An estimation of the performance and power is required to dynamically adapt the core and the operating frequency to maximize BIPS$^3/W$. The average and variance of the power dissipation for the *gcc* and *lbm* benchmarks for various frequency/voltage pairs and for each core size is shown through the boxplots in Fig. 5. Each application has a different power footprint. The *gcc* application is more power hungry as compared to the *lbm*. In addition, the variation in power consumption is larger for *gcc* as compared to *lbm*, which experiences low variation in power dissipation. The variation in power is more pronounced in the *gcc* when the core has more resources and operates at higher frequencies.

For a phase of an application that is more sensitive to core size (rather than frequency), execution of the phase with more bandwidth (larger core) and at a lower frequency optimizes the BIPS$^3/W$. The goal is to accurately estimate the power and performance and find the best solution that maximizes BIPS$^3/W$.

C. Power and Performance Estimation

Recent work has proposed OLSLRM to estimate the power [14] and performance [16], [33] of a processor at run-time. In [9], the OLSLRM was also used for resource adaptation of ElasticCore in order to improve the energy efficiency. It was shown that OLSLRM is not the best suited algorithm for performance and power estimation as outliers can deceive the model, as applications experience different phases with different behaviors. In addition, superscalar processors are complex, which makes it difficult to develop a general model for power/performance estimation. OLSLRM is highly sensitive to the outliers and potentially produce misleading results as even a single point of data substantially impacts the accuracy of the regression model. Two separate linear regression algorithms in addition to OLSLRM are evaluated in this paper: 1) RLRM [34] and 2) QLRM [35]. RLRM is more effective as compared to OLSLRM when the data includes significant number of outliers. There are different methods

TABLE II
PERFORMANCE COUNTERS USED FOR THE REGRESSION MODEL

Category	**Hardware performance counter**	
Memory subsystem	L1 data access	L1 data miss
	L2 cache access	L2 cache miss
	D-TLB miss	
Instruction Mix	Integer instruction issue	
	Integer floating point issue	
	Load/Store instruction issue	
Branch	Branch misprediction	

for robust linear regression; however, all give less weight iteratively to the outlier samples that influence the regression model. Out of several proposed techniques for robust regression, the bisquare estimator [34] is used in this paper, a method applied in statistics to determine the coefficients of a linear model. Using the bisquare estimator has been mathematically proven to be at least as efficient as OLSLRM if the distribution of samples is normal; however, it is robust to influential outlier samples [36].

Similar to RLRM, the main advantage of QLRM as compared to OLSLRM is robustness against outliers. All three regression models are applied to the studied benchmarks and the absolute error is measured in each case. The parameters for estimating performance are listed in Table II. Note that removing any of the selected parameters potentially reduces the accuracy of the average case by a few percent. It is therefore possible to trade accuracy for a reduction in the number of parameters. However, as the regression is only performed for each control period, resulting in a small overhead, all parameters are kept to obtain higher accuracy.

The LRM used for performance estimation is given by

$$\mathrm{LRM} = \left(\beta_0 + \sum_{i=1}^{i=9} \beta_i \mathrm{Pc}_i\right) + e_i \tag{1}$$

where β_0 is the intercept, β_i are coefficients of the regression model, and Pc_i are target performance counters. The error term e_i is used by both the OLSLRM and RLRM to minimize the mean-square error. For the quantile regression model,

TABLE III
AVERAGE ERROR OF THE THREE IMPLEMENTED LINEAR REGRESSION ALGORITHMS FOR DIFFERENT CORE CONFIGURATIONS

		Core size			
		Big	Medium	Small	Tiny
		OLSLRM			
Frequency	3.0 GHz	29.46 %	24.45 %	21.24 %	14.89 %
	2.5 GHz	26.41 %	21.71 %	18.63 %	12.93 %
	2.0 GHz	23.23 %	18.82 %	15.93 %	10.85 %
	1.5 GHz	19.86 %	15.74 %	13.04 %	8.76 %
		RLRM			
	3.0 GHz	21.98 %	18.03 %	15.08 %	10.66 %
	2.5 GHz	20.11 %	16.33 %	13.33 %	9.54 %
	2.0 GHz	17.98 %	14.51 %	11.55 %	8.22 %
	1.5 GHz	15.63 %	12.48 %	9.77 %	6.93 %
		QLRM			
	3.0 GHz	16.28 %	13.14 %	11.96 %	9.49 %
	2.5 GHz	15.34 %	12.12 %	11.01 %	8.65 %
	2.0 GHz	14.22 %	11.10 %	9.94 %	7.62 %
	1.5 GHz	12.86 %	9.93 %	8.86 %	6.50 %

a specific quantile of data is set instead of the mean value. The quantile is set to 0.5 in this paper, which results in minimizing the median of the error values. Note that despite using different algorithms, the objective of all three methods is to find the coefficients of the LRM presented in (1).

Incurring a minimum cost in hardware and complexity, implementing LRM in ElasticCore or similar adaptive architectures is desirable. Therefore, the primary objective is to explore and optimize a linear model that exploits different schemes, rather than finding a complex nonlinear model. Note that the overhead area, power, and performance overhead to implement an LRM in hardware are negligible [16]. We show that the QLRM [35] achieves the highest accuracy among the three regression algorithms.

The average error of the three LRMs for performance estimation at different operating points is listed in Table III. For a big core running at 3-GHz frequency, the average error when estimating performance is 29.46%, 21.98%, and 16.28% when using OLSLRM, RLRM, and QLRM, respectively. Scaling down the frequency or the core size leads to an improvement in performance. The error is 6.5% with QLRM when the core is tiny and running at the minimum frequency, as reducing the frequency or core size bounds the maximum achievable performance and reduces the variations in each control period. For all cases, QLRM outperforms OLSLRM and RLRM. The QLRM is therefore used in this paper.

For the training of the coefficients in QLRM, the first 200 million executed instructions of each benchmark are selected and combined to generate a training set. Statistically, 25% of all data are used for training the model. Performance and power are estimated at the end of each control period (i.e., every 200k instructions). The absolute error in performance and power estimation for different operating points is presented as boxplots in Figs. 6 and 7, respectively. The worst case median of the absolute error is less than 18%, which implies that the quantile regression model can estimate half of the operating points with less than 18% error. The median

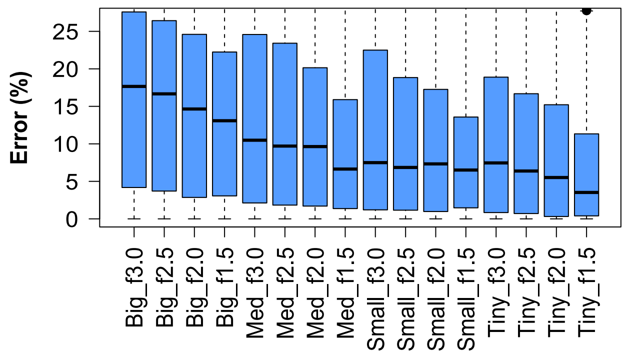

Fig. 6. Error in performance estimation at different operating points.

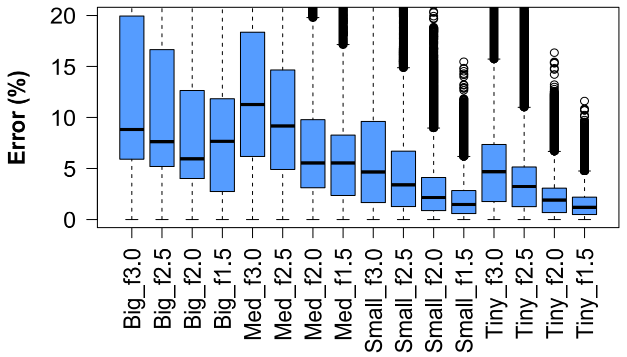

Fig. 7. Power estimation error at different operating points.

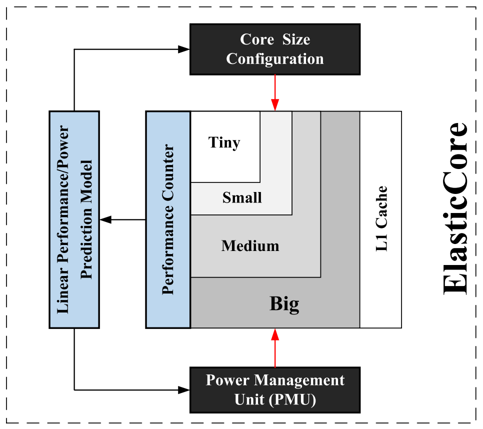

Fig. 8. ElasticCore architecture.

of the error is less than 5% when using a tiny core operating at the minimum frequency.

For the estimation of the power dissipation, only the IPC is used as an input parameter for the regression model. The performance at different operating points is calculated first, and then the power dissipation is determined at these operating points. The same training is performed for the power estimation coefficients with the results shown in Fig. 7. There are many outliers in the data. However, for most of the operating points, the median of the error is less than 10%.

The final architecture of the ElasticCore is shown in Fig. 8. At the end of each control period, which occurs every 200k instructions, the inputs of the regression model (metrics provided in Table II) are fed through the hardware performance counters. The performance and power of the system are estimated for different core configurations based on the linear model built by QLRM. The configuration corresponding to the highest energy efficiency is determined based on the model. The power management unit (PMU) and core size configuration unit then set the corresponding voltage and core size to optimize efficiency.

D. Power Delivery to the ElasticCore

The current demand of the ElasticCore fluctuates significantly at run-time due to dynamic core scaling. The power delivery topology, therefore, plays a crucial role in the energy efficiency of the ElasticCore. The ElasticCore not only requires a power delivery system with high PCE but also a fine-grained, low-overhead voltage control for scaling the core, and the voltage at run-time. The ElasticCore is, therefore, designed with OCVRs that offer fast DVFS [6], a reduced printed circuit board footprint, and a reduction in parasitic losses [23].

The fully integrated VR (FIVR) in the fourth-generation Intel core microprocessors (Haswell) [37], [38] offers up to three times higher peak power as well as higher performance for a given power level as compared with previous generations of Intel core microprocessors. The power delivery system of the ElasticCore is modeled using a configuration similar to the Intel Haswell processor. The block diagram of the proposed two-tiered power delivery system is shown in Fig. 9. Instead of serving the ElasticCore with a single OCVR that provides a high output current rating, each partition (CW1 through CW4) is served with a dedicated OCVR with a maximum output current rating proportional to the peak current demand of the partition CW1. The current rating is defined in terms of the maximum output current supplied by the VR. The PCE of a switching type OCVR is directly proportional to the load current and inversely proportional to the switching frequency [6], [39]. Implementing multiple OCVRs of lower current rating, each serving a different partition (CW1 through CW4) of the ElasticCore instead of a single OCVR with a high rating provides multiple advantages.

1) Variation in the PCE is reduced as each OCVR supplies current to a load with smaller current variation.
2) OCVRs with moderate peak current density ($I_{\max}$ per unit area) and low switching frequency are used to serve each partition if a higher weight is assigned to energy efficiency rather than area efficiency.
3) OCVRs with lower rating provide faster voltage transitions and, therefore, improved performance with DVFS.
4) Reduction in the leakage power of the ElasticCore by selectively shutting down the OCVRs serving unused partitions (CW2 through CW4).

A multitiered power supply hierarchy is implemented where an off-chip VR first converts the power supply voltage (3.3 to 12 V) or Li–Ion battery voltage (3.7 V) to a fixed output voltage V_{VR} of 1.35 V. The OCVRs then convert the V_{VR} to one of the four discrete voltage levels V_{core} (1.35, 1.132, 0.914, and 0.75 V) controlled by the PMU. The L2 cache is on a separate voltage domain of 1.13 V and is served by a dedicated on-chip low dropout (LDO) VR. The LDO provides a high PCE (η_{LDO}) of close to 90% [39] due to the small voltage drop from the input voltage V_{VR}.

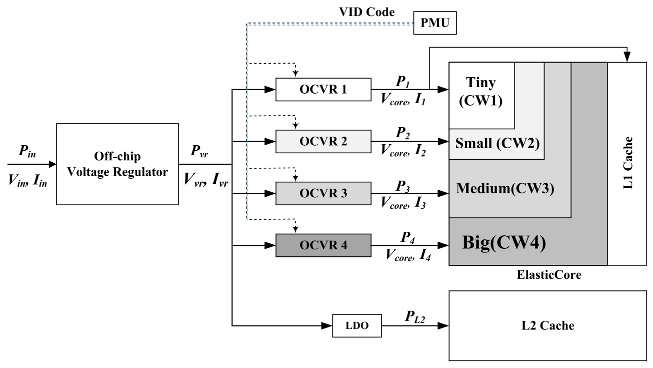

Fig. 9. Two-tiered power delivery configuration for the ElasticCore with multiple OCVRs.

The peak load current consumed by the core across all four voltages, as obtained from McPAT [40], varies from 8 to 30 A. Most VRs offer a PCE in the range of 85%–95% to meet 8 to 30 A load current demand. For example, the PCE of a Texas Instruments LM27403 [41] (voltage mode synchronous buck controller) is 93% with an input voltage of 12 V. The variation in the PCE for the LM27403 is less than 3% for a load current in the range of 5–40 A. The LM27403 is selected as the VR for the ElasticCore. The PCE of the VR is assumed to remain constant across the different modes of operation of the ElasticCore. The current ratings of the OCVRs are selected according to the peak load current of each core partition, where a three-phase buck converter is required for CW1 and an eighteen-phase converter for CW4. Each OCVR has a similar topology to the FIVR [38]. The current density of the FIVR is 31 A/mm^2 [38]. The proposed power delivery system with dedicated OCVRs modeled as multiphase FIVR circuits, therefore, occupies an on-chip area of 1.87 mm^2. The peak PCE of each OCVR remains 90% for the given number of phases and load current. The PCE for the two-tiered power delivery configuration shown in Fig. 9 is given by (2). The energy efficiency of the power delivery system η_{system} is equal to the product of the energy efficiency of the off-chip (η_{VR}) and energy efficiency of the on-chip regulators (η_{FIVR}), where η_{FIVR} is assumed equal to η_{LDO}. Based on the given PCEs of the VR and FIVR, the two-tiered power delivery system of the ElasticCore offers a peak PCE of 84%

$$\eta_{\mathrm{system}} = \frac{P_{\mathrm{out}}}{P_{\mathrm{in}}} = \frac{\sum_{i=1}^{4} P_i + P_{L2}}{P_{\mathrm{VR}}} = \eta_{\mathrm{VR}} \times \eta_{\mathrm{FIVR}}. \quad (2)$$

With the proposed implementation, the power delivery network of the ElasticCore offers not only fine-grained DVFS but also a constant and high PCE across application behavior at run-time.

V. Methodology and Results

The SMTSIM simulator [42] and McPAT [40] are integrated to obtain the power dissipation of each component

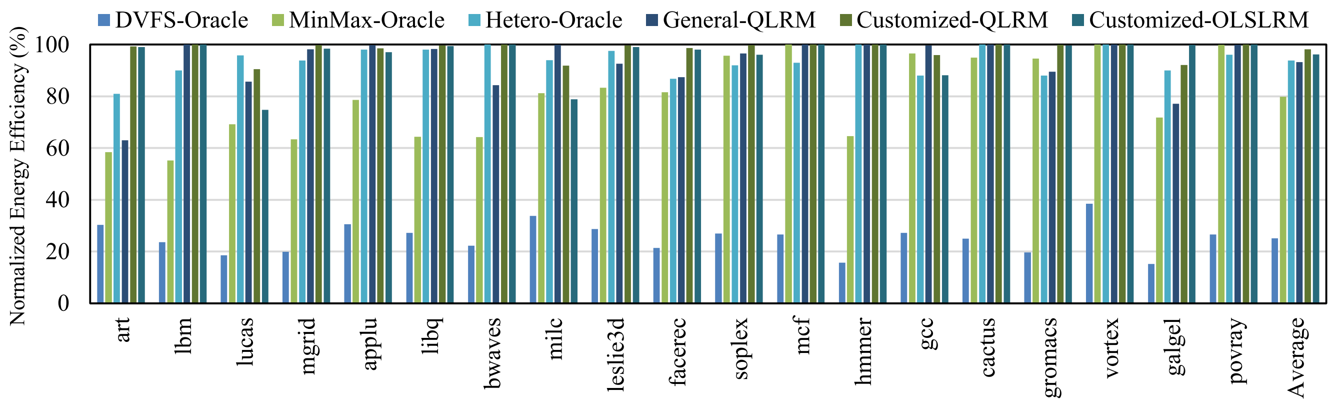

Fig. 10. BIPS^3/W of the applications on various architectures normalized to the ElasticCore-Oracle.

while simulating the target architectures. Each benchmark is simulated for 800M instructions after fast forwarding past the first two billion instructions, and the performance counter data are captured every 200k instructions thereafter. DVFS and core scaling are also performed during the same interval. The DVFS performance of the proposed two-tiered power delivery system for the ElasticCore is compared with a single-tier off-chip VR. A time overhead of 100 ns and 100 μs is assumed for the proposed power delivery scheme for the OCVRs and the off-chip VR, respectively [15]. In addition, the time overhead of implementing the LRM in hardware and calculating values at each interval is assumed negligible [16]. Note that the coefficients are calculated offline and the only overhead at run-time is multiplication and summation of coefficients for the selected performance counters. The power overhead of implementing a LRM is estimated as 5 μW, which is further reduced by gating idle units during each interval [16]. The following schemes are evaluated for comparison.

1) *DVFS-Oracle:* This scheme implements a big core with DVFS capability. It is an oracle scheme as the exact application behavior is known in advance and the best DVFS setting is used at each control period to improve the energy efficiency. The DVFS-Oracle scheme was the primary approach used by industry prior to the development of heterogeneous architectures to respond to the demand of the most computing intensive applications.
2) *MinMax-Oracle:* Uses DVFS similar to the DVFS-Oracle, however, another core is added to realize heterogeneity in the system. In other words, MinMax-Oracle employs the configuration with the highest performance (big) and the one with the greatest power efficiency (tiny). The selected configuration shows diverse power and performance tradeoffs, but is limited to only the two core configurations.
3) *Hetero-Oracle:* This scheme has no DVFS capability at all and instead exploits heterogeneity in a wide range of core configurations (i.e., big, medium, small, and tiny) to improve energy efficiency. Comparing Hetero-Oracle with DVFS-Oracle provides insights on the impact of DVFS and heterogeneity through resizing of the core on the energy efficiency.
4) *General-QLRM:* This is the ElasticCore architecture implemented with the QLRM described by (1) to estimate the BIPS$^3/W$ for various core sizes and frequency/voltage pairs. The first 200M instructions for each benchmark are used as training data. As opposed to Customized-QLRM, the coefficients are fixed for all executing applications in this scheme. Therefore, general-QLRM better represents general-purpose architectures.
5) *Customized-QLRM:* Similar to the general-QLRM, this is the ElasticCore architecture with the QLRM described by (1) to estimate the BIPS$^3/W$ for various core sizes and frequency/voltage pairs. Note that the coefficients are determined per application with training from the first 200M instructions for each benchmark. This scheme is not general and requires reprogramming the coefficients for each application, which is unsuitable for general-purpose computing. However, for many embedded systems, where the type and the behavior of applications are known in advance, customized-QLRM is applicable.
6) *Customized-OLSLRM:* This scheme is presented in [9] and uses OLSLRM as an LRM. Similar to customized-QLRM, the coefficients of the model are determined per application and, therefore, coefficient reprogramming is necessary to obtain an optimal result for each application.
7) *ElasticCore-Oracle:* This is the ElasticCore architecture with a perfect application predictor, where all future behavior of the application as well as the power and performance for various configurations are known in advance. The ElasticCore-Oracle, therefore, adapts the core resources and frequency, and exploits all opportunities to maximize energy efficiency. It provides the upper bound for energy efficiency and is, therefore, used to normalize and compare all the other schemes.

The results of all the benchmarks normalized to the ElasticCore-Oracle are shown in Fig. 10. Using only DVFS limits the improved energy efficiency of the core as compared to using a heterogeneous architecture (4× compared to ElasticCore-Oracle). In addition, Hetero-Oracle, which only exploits heterogeneity without DVFS, improves the energy efficiency by 3.7× on average as compared to DVFS-Oracle. The only difference between Hetero-Oracle and ElasticCore-Oracle is that the latter uses DVFS to further improve energy efficiency. However, the results indicate that DVFS only

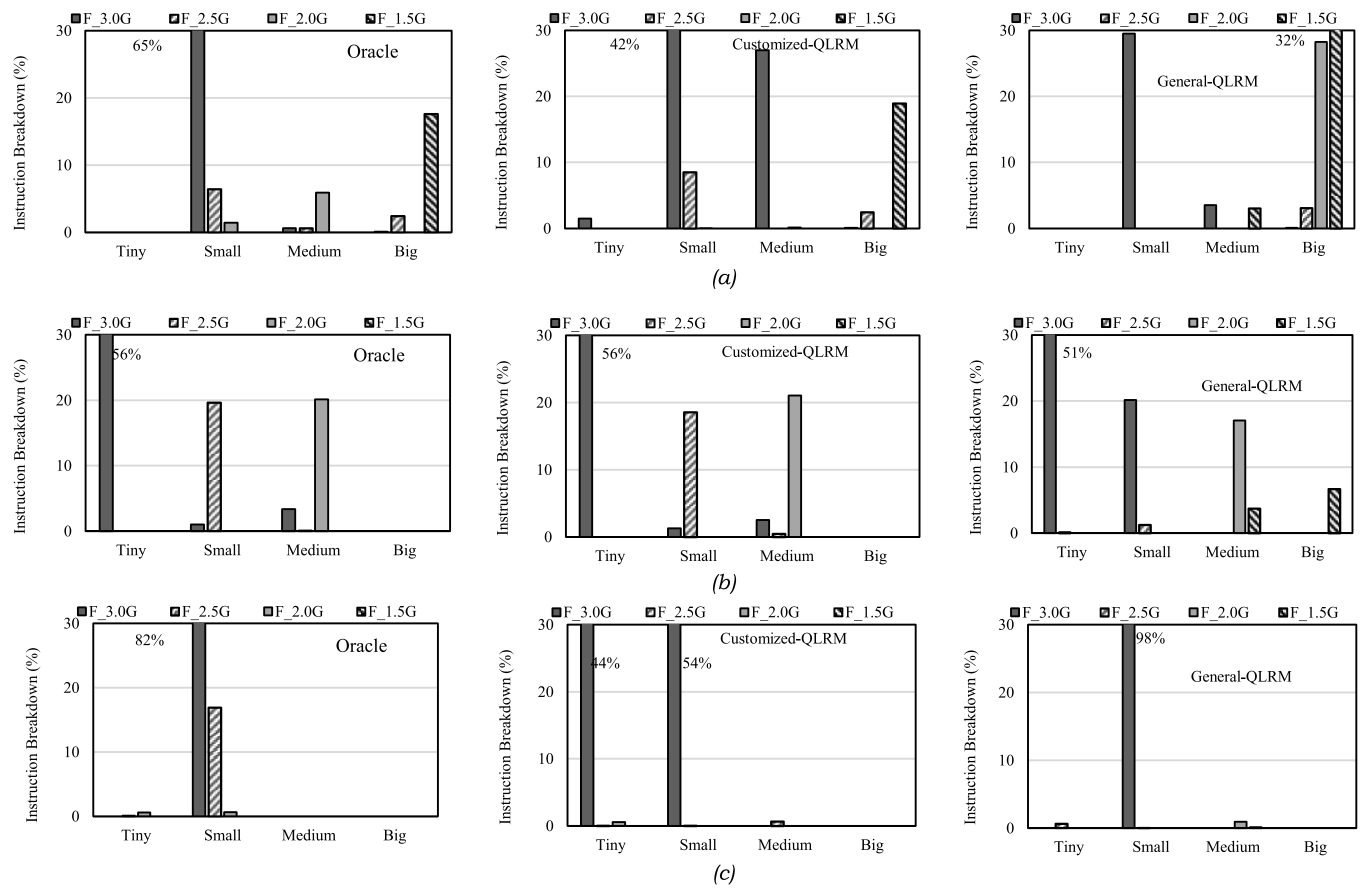

Fig. 11. Distribution of executed instructions on the various core sizes and frequencies for (a) *lucas*, (b) *art,* and (c) *milc* benchmarks.

improves energy efficiency by less than 7%, on average. Note that even adding a tiny core with a big core in the DVFS-Oracle to implement the MinMax-Oracle scheme reduces the efficiency gap from 75% to 20% on average. For some benchmarks, such as *povrary* and *mcf*, the MinMax-oracle performs close to the ElasticCore-Oracle. For other benchmarks such as *art* and *lbm*, the MinMax-Oracle is approximately 40% less energy efficient than the ElasticCore-Oracle. Some benchmarks such as *gcc* and *gromacs* execute more efficiently on a MinMax-Oracle rather than a Hetero-Oracle. The results indicate that these benchmarks are better suited for MinMax-Oracle, which includes eight operating points (four frequency and two core type configurations), as compared to Hetero-Oracle, which includes four operating points.

The general-QLRM performs similar to the ElasticCore-Oracle for many of the benchmarks and on average produces a 93% energy efficiency as compared to ElasticCore-Oracle. General-QLRM produces a poor efficiency only for the *art* benchmark due to a lack of accurate estimation of performance and power. The behavior of general-QLRM for a subset of the studied benchmarks is shown in Fig. 11.

Customized-QLRM performs best as compared to the other schemes, with close to 98% energy efficiency as compared to the ElasticCore-Oracle. However, reprogramming the coefficients based on the application is a significant drawback of customized-QLRM.

The distribution of executed instructions among various core sizes and operating frequencies is shown in Fig. 11. Each column provides the breakdown of instructions for the ElasticCore-Oracle, customized-QLRM, and general-QLRM. The results indicate that each selected benchmark requires a different core size and operating frequency to optimize energy efficiency. As an example, *art* mostly uses the tiny core configuration, whereas *lucas* and *milc* are optimized on the small configuration. The *lucas* benchmark performs poorly as compared to other benchmarks for both the customized-QLRM and general-QLRM. The customized-QLRM uses a medium core configuration instead of the small for some portion of applications, which leads to increased inefficiency. The general-QLRM tends to use big core at minimum frequencies instead of running applications on a small core configuration at the highest frequency. For the *lucas* benchmark, the customized-QLRM and general-QLRM are approximately 10% and 15% less energy efficient, respectively, as compared to ElasticCore-Oracle.

The *art* benchmark is also shown in Fig. 11. The general-QLRM performs poorly as compared to the customized-QLRM for the *art* benchmark. The customized-QLRM closely follows ElasticCore-Oracle in terms of instruction distribution at different operating points for the *art* benchmark. In addition, the general-QLRM inaccurately selects the frequency for the small core configuration and executes 7% of the

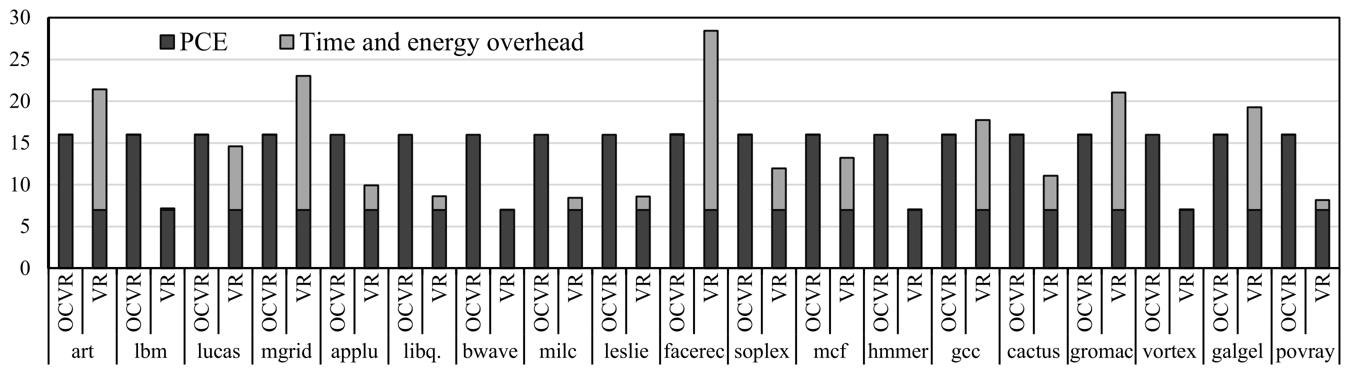

Fig. 12. Impact of the PCE of the OCVR and off-chip VR on the energy efficiency.

instructions on a big core at a minimum frequency. For the *milc* benchmark, general-QLRM outperforms customized-QLRM as general-QLRM is trained with more benchmarks as compared to customized-QLRM and the first 200M instructions of the *milc* benchmark do not accurately represent the entire benchmark. The quantile regression model is accurate enough to capture the variation in the application behavior at run-time and produce accurate estimates of the power and performance to adapt the core size and frequency.

The impact of an on-chip (OCVR) and off-chip VR on the energy efficiency at the application level is shown in Fig. 12. Two factors reduce the energy efficiency at the front-end. The first factor is the PCE of the VRs. As described in Section IV-D, 7% and 16% of the input power is lost in the off-chip VR and OCVR, respectively. The second factor is the time and energy overheads of scaling the core or the operating voltage/frequency. During voltage scaling and turning the OCVRs on/off, the ElasticCore is stalled. For the former case, the voltage must be stable for reliable operation of the ElasticCore. For the latter case, the OCVRs turn ON or OFF according to the required core size.

Even though the PCE of the OCVR is less than the off-chip VR, the fast DVFS and phase adaptation as well as more control on leakage power provide additional advantages when using OCVRs. In comparison, for the off-chip VRs, the use of frequent DVFS in the range of a few nanoseconds is not possible. There is, therefore, a tradeoff such that for some applications where frequent core and frequency scaling are required, the OCVR is more beneficial, and in contrast, for other applications where a few changes are required, the off-chip VR is better suited. For instance, the time and energy overheads imposed by the off-chip VR for the *facerec* benchmark reduces the total energy efficiency as compared to an OCVR. The proposed two-tiered power delivery topology offers high energy efficiency even for applications that do not benefit from frequent frequency and core scaling by introducing high-speed switches that *switch* the supply of current to each partition of the ElasticCore between the OCVRs and off-chip VR.

VI. Conclusion

This paper presents ElasticCore, a dynamic heterogeneous architecture that permits adaptation of its resources at run-time based on the application requirements to enhance energy efficiency. ElasticCore concurrently scales the bandwidth, capacity, and voltage/frequency based on the behavior of the applications at run-time. To guide the adaptation of the core resources along with the operating voltage and frequency, the performance and power sensitivity of various standard benchmarks to different core sizes (big, medium, little, and tiny) and frequencies are first investigated. Based on the characterization results, three regression-based models to accurately estimate the power and performance of the applications at run-time are studied. The results indicate that the QLRM is the most accurate when estimating power and performance. In addition, the dynamically scalable partitions of the ElasticCore are powered with multiple OCVRs with high PCE that are able to realize fast DVFS. Both on-chip and off-chip VRs are analyzed to determine the total energy efficiency at the application level. This paper further analyzes various tuning knobs of the ElasticCore architecture to understand the effectiveness on enhancing energy efficiency. The results indicate that DVFS by itself does not perform well in terms of energy efficiency, whereas core scaling enhances energy efficiency noticeably for many of the studied applications. However, when DVFS is used in conjunction with core scaling, energy efficiency is further improved by 7% on average. The ElasticCore with a general estimator and a customized estimator per benchmark are shown to achieve close to a 93% and 98% efficiency, respectively, as compared to an architecture implementing the Oracle power and performance predictor, where the application behavior is perfectly matched at run-time.

References

[1] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," *ACM SIGARCH Comput. Archit. News*, vol. 39, no. 3, pp. 365–376, Jun. 2011.

[2] A. Branover, D. Foley, and M. Steinman, "AMD fusion APU: Llano," *IEEE Micro*, vol. 32, no. 2, pp. 28–37, Mar./Apr. 2012.

[3] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen, "Single-ISA heterogeneous multi-core architectures: The potential for processor power reduction," in *Proc. Annu. IEEE Int. Symp. Microarchit.*, Dec. 2003, pp. 81–92.

[4] P. Greenhalghl. (2011). *big.LITTLE Processing With ARM Cortex-A15-Cortex-A7*. [Online]. Available: http://www.eetimes.com/document.asp?doc id=1279167

[5] W. Zhang, H. Zhang, and J. Lach, "Dynamic core scaling: Trading off performance and energy beyond DVFS," in *Proc. 33rd IEEE Int. Conf. Comput. Design (ICCD)*, Oct. 2015, pp. 319–326.

[6] W. Kim, M. S. Gupta, G.-Y. Wei, and D. Brooks, "System level analysis of fast, per-core DVFS using on-chip switching regulators," in *Proc. IEEE 14th Int. Symp. High Perform. Comput. Archit.*, Feb. 2008, pp. 123–134.

[7] M. K. Tavana, N. Teimouri, M. Abdollahi, and M. Goudarzi, "Simultaneous hardware and time redundancy with online task scheduling for low energy highly reliable standby-sparing system," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 4, 2014, Art. no. 86.

[8] M. Pricopi, T. S. Muthukaruppan, V. Venkataramani, T. Mitra, and S. Vishin, "Power-performance modeling on asymmetric multi-cores," in *Proc. Int. Conf. Compil., Archit. Synth. Embedded Syst. (CASES)*, 2013, pp. 1–10.

[9] M. K. Tavana, M. H. Hajkazemi, D. Pathak, I. Savidis, and H. Homayoun, "ElasticCore: Enabling dynamic heterogeneity with joint core and voltage/frequency scaling," in *Proc. 52nd Design Autom. Conf. (DAC)*, 2015, Art. no. 151.

[10] S. Srinivasan, N. Kurella, I. Koren, and S. Kundu, "Dynamic reconfiguration vs. DVFS: A comparative study on power efficiency of processors," in *Proc. 29th Int. Conf. VLSI Design Int. Conf. Embedded Syst.*, 2016, pp. 563–564.

[11] A. Lukefahr, S. Padmanabha, R. Das, R. Dreslinski, Jr., T. F. Wenisch, and S. Mahlke, "Heterogeneous microarchitectures trump voltage scaling for low-power cores," in *Proc. Int. Conf. Parallel Archit. Compilation*, 2014, pp. 237–250.

[12] S. Mittal, "A survey of techniques for architecting and managing asymmetric multicore processors," *ACM Comput. Surv.*, vol. 48, no. 3, 2016, Art. no. 45.

[13] Texas Instrument. (2015). *OMAP 5 Applications Processors*. [Online]. Available: http://www.ti.com/product/OMAP5432/technicaldocuments

[14] J. Cong and B. Yuan, "Energy-efficient scheduling on heterogeneous multi-core architectures," in *Proc. ACM/IEEE Int. Symp. Low Power Electron. Design*, 2012, pp. 345–350.

[15] C. Kim *et al.*, "Composable lightweight processors," in *Proc. 40th Annu. IEEE/ACM Int. Symp. Microarchitecture*, Dec. 2007, pp. 381–394.

[16] A. Lukefahr *et al.*, "Composite cores: Pushing heterogeneity into a core," in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2012, pp. 317–328.

[17] R. Whitwam. (2014). *Qualcomm Unveils 64-Bit Snapdragon 808 and 810 SoCs: The Apple A7 Stop-Gap Measures Continue*. [Online]. Available: http://www.extremetech.com/computing/179987-qualcomm-unveils-64-bit-snapdragon-808-and-810-socs-the-apple-a7-stop-gap-measures-continue

[18] L. Gil. (2015). *NVIDIAs Tegra X1 Crushes the Competition*. [Online]. Available: https://liliputing.com/2015/02/nvidias-tegra-x1-crushes-the competition.html

[19] Samsung. (2013). *SAMSUNG Highlights Innovations in Mobile Experiences Driven by Components, in CES Keynote*. [Online]. Available: http://www.samsung.com/us/news/20353

[20] G. Liu, J. Park, and D. Marculescu, "Dynamic thread mapping for high-performance, power-efficient heterogeneous many-core systems," in *Proc. IEEE 31st Int. Conf. Comput. Design (ICCD)*, Oct. 2013, pp. 54–61.

[21] G. Liu, J. Park, and D. Marculescu, "Procrustes^1: Power constrained performance improvement using extended maximize-then-swap algorithm," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 34, no. 10, pp. 1664–1676, Oct. 2015.

[22] K. V. Craeynest, A. Jaleel, L. Eeckhout, P. Narvaez, and J. Emer, "Scheduling heterogeneous multi-cores through performance impact estimation (PIE)," *ACM SIGARCH Comput. Archit. News*, vol. 40, no. 3, pp. 213–224, 2012.

[23] M. K. Tavana, D. Pathak, M. H. Hajkazemi, M. Malik, I. Savidis, and H. Homayoun, "Realizing complexity-effective on-chip power delivery for many-core platforms by exploiting optimized mapping," in *Proc. 33rd IEEE Int. Conf. Comput. Design (ICCD)*, Oct. 2015, pp. 581–588.

[24] V. Kontorinis, M. K. Tavana, M. H. Hajkazemi, D. M. Tullsen, and H. Homayoun, "Enabling dynamic heterogeneity through core-on-core stacking," in *Proc. 51st Annu. Design Autom. Conf.*, 2014, pp. 1–6.

[25] E. Ipek, M. Kirman, N. Kirman, and J. F. Martinez, "Core fusion: Accommodating software diversity in chip multiprocessors," *SIGARCH Comput. Archit. News*, vol. 35, no. 2, pp. 186–197, 2007.

[26] K. Khubaib, M. A. Suleman, M. Hashemi, C. Wilkerson, and Y. N. Patt, "MorphCore: An energy-efficient microarchitecture for high performance ILP and high throughput TLP," in *Proc. 45th Annu. IEEE/ACM Int. Symp. Microarchit.*, 2012, pp. 305–316.

[27] K. K. Rangan, M. D. Powell, G.-Y. Wei, and D. Brooks, "Achieving uniform performance and maximizing throughput in the presence of heterogeneity," in *Proc. IEEE 17th Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2011, pp. 3–14.

[28] D. Ponomarev, G. Kucuk, and K. Ghose, "Dynamic resizing of superscalar datapath components for energy efficiency," *IEEE Trans. Comput.*, vol. 55, no. 2, pp. 199–213, Feb. 2006.

[29] D. H. Albonesi *et al.*, "Dynamically tuning processor resources with adaptive processing," *Computer*, vol. 36, no. 12, pp. 49–58, Dec. 2003.

[30] A. Buyuktosunoglu, T. Karkhanis, D. H. Albonesi, and P. Bose, "Energy efficient co-adaptive instruction fetch and issue," in *Proc. 30th Annu. Int. Symp. Comput. Archit.*, Jun. 2003, pp. 147–156.

[31] A. Baniasadi and A. Moshovos, "Instruction flow-based front-end throttling for power-aware high-performance processors," in *Proc. Int. Symp. Low Power Electron. Design*, 2001, pp. 16–21.

[32] O. Azizi, A. Mahesri, B. C. Lee, S. J. Patel, and M. Horowitz, "Energy-performance tradeoffs in processor architecture and circuit design: A marginal cost analysis," *ACM SIGARCH Comput. Archit. News*, vol. 38, no. 3, pp. 26–36, 2010.

[33] B. C. Lee and D. M. Brooks, "Accurate and efficient regression modeling for microarchitectural performance and power prediction," *ACM SIGOPS Oper. Syst. Rev.*, vol. 40, no. 5, pp. 185–194, 2006.

[34] J. Cohen, P. Cohen, S. G. West, and L. S. Aiken, *Applied Multiple Regression/Correlation Analysis for the Behavioral Sciences*. Evanston, IL, USA: Routledge, 2013.

[35] R. Koenker, *Quantile Regression*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[36] W. H. Greene, *Econometric Analysis*. London, U.K.: Pearson Education, 2003.

[37] P. Hammarlund *et al.*, "Haswell: The fourth-generation intel core processor," *IEEE Micro*, vol. 34, no. 2, pp. 6–20, Mar. 2014.

[38] E. A. Burton *et al.*, "FIVR—Fully integrated voltage regulators on 4th generation Intel Core SoCs," in *Proc. IEEE Appl. Power Electron. Conf. Expo. (APEC)*, Mar. 2014, pp. 432–439.

[39] I. Vaisband and E. G. Friedman, "Heterogeneous methodology for energy efficient distribution of on-chip power supplies," *IEEE Trans. Power Electron.*, vol. 28, no. 9, pp. 4267–4280, Sep. 2013.

[40] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchit.*, Dec. 2009, pp. 469–480.

[41] *Voltage Mode Synchronous Buck Controller With Temperature-Compensated DCR Current Sensing*, document LM27403 3-20V, Texas Instrument, Dallas, TX, USA, 2014. [Online]. Available: http://www.ti.com/lit/ds/symlink/lm27403.pdf

[42] D. M. Tullsen, "Simulation and modeling of a simultaneous multithreading processor," in *Proc. 22nd Int. Conf. Resource Manage. Perform. Eval. Enterprise Comput. Syst.*, 1996, pp. 819–828.

Mohammad Khavari Tavana (S'17) received the M.Sc. degree in computer engineering from Sharif University of Technology, Tehran, Iran, in 2011. He is currently working toward the Ph.D. degree at Northeastern University, Boston, MA, USA.

He received the Chair's Scholarship at Northeastern University, where he is currently a member of the Northeastern University Computer Architecture Research Laboratory. His current research interests include computer architecture, emerging memory systems, and energy-efficient computing.

Mr. Khavari Tavana is a Student Member of the Association for Computing Machinery (ACM).

Mohammad Hossein Hajkazemi received the B.Sc. degree in computer engineering from Shahed University, Tehran, Iran, in 2008 and the M.Sc. degree from the University of Victoria, Victoria, BC, Canada, in 2013. He is currently working toward the Ph.D. degree at Northeastern University, Boston, MA, USA.

His current research interests include emerging storage technologies and distributed storage systems.

Divya Pathak received the M.S. degree in computer engineering from Drexel University, Philadelphia, PA, USA, in 2015, where she is currently working toward the Ph.D. degree in electrical and computer engineering.

Her current research interests include circuits, systems, and microarchitecture design for post-Moore computing including algorithms, modeling, optimization, and very large scale integration design.

Ioannis Savidis received the B.S.E. degree in electrical and computer engineering and biomedical engineering from Duke University, Durham, NC, USA, in 2005 and the M.Sc. and Ph.D. degrees in electrical and computer engineering from the University of Rochester, Rochester, NY, USA, in 2007 and 2013, respectively.

In 2007, he joined the 3-D Integration Group, Freescale Semiconductor, Austin, TX, USA. From 2008 to 2011, he was with the System on Package and 3-D Integration Group, IBM T. J. Watson Research Center, Yorktown Heights, NY, USA. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Drexel University, Philadelphia, PA, USA, where he directs the Integrated Circuits and Electronics Design and Analysis Laboratory. His current research and teaching interests include analysis, modeling, and design methodologies for high-performance digital and mixed-signal integrated circuits, power management for system-on-chip and microprocessor circuits, hardware security, and interconnect related issues for heterogeneous 2-D and 3-D circuits.

Dr. Savidis is a member of the Association for Computing Machinery (ACM), the IEEE Circuits and Systems Society, the IEEE Communications Society, and the IEEE Electron Devices Society. He serves on the organizing committees of the IEEE International Symposium on Hardware Oriented Security and Trust and the International Verification and Security Workshop. He also serves on the editorial boards of the IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, the *Microelectronics Journal*, and the *Journal of Circuits, Systems and Computers*.

Houman Homayoun received the B.S. degree in electrical engineering from Sharif University of Technology, Tehran, Iran, in 2003, the M.S. degree in computer engineering from the University of Victoria, Victoria, BC, Canada, in 2005, and the Ph.D. degree from the Department of Computer Science, University of California, Irvine, CA, USA, in 2010.

He is currently an Assistant Professor with the Electrical and Computer Engineering Department, George Mason University, Fairfax, VA, USA. He also holds a joint appointment with the Computer Science as well as Information Science and Technology Departments. He is the Director of GMU's Green Computing and Heterogeneous Architectures Laboratory. He was with the University of California, San Diego, La Jolla, CA, USA, as National Science Foundation Computing Innovation Fellow awarded by the CRA and CCC. His current research interests include the design of next generation heterogeneous multicore accelerator for big data processing, nonvolatile STT logic, heterogeneous accelerator platforms for wearable biomedical computing, and logical vanishable design to enhance hardware security.