Comparison of Single-ISA Heterogeneous versus Wide Dynamic Range Processors for Mobile Applications

Hamid Reza Ghasemi[1], Ulya R. Karpuzcu[2], Nam Sung Kim[3]

[1]hamid@cs.wisc.edu, [2]ukarpuzcu@umn.edu, [3]nskim@illinois.edu

[1]CS Department, University of Wisconsin-Madison, 1210 W. Dayton St., Madison, WI, USA

[2]ECE Department, University of Minnesota, 200 Union St. S.E. Minneapolis, MN, USA

[3]ECE Department, 306 N. Wright St. Urbana, IL, USA

***Abstract*—Mobile computing devices demand processors to offer a wide range of performance/power trade-offs so that they can provide much needed high performance or low power consumption depending on a given operating requirement. While dynamic voltage/frequency scaling (DVFS) has been the most powerful technique to provide such trade-offs, few processor vendors have the capability to provide a sufficient DVFS range requiring joint optimization of devices and circuits. Facing such a challenge, two promising approaches are proposed: scaling the amount of processor resources such as on-chip memory and execution units, i.e., dynamic resource scaling (DRS) and switching between big out-of-order (OoO) and little in-order cores in a single-ISA heterogeneous processor such as ARM's big.LITTLE. In this paper, we compare a single-ISA heterogeneous processor with a wide dynamic range (WDR) processor augmented with DRS in terms of (1) device-, circuit-, architecture-level implications, (2) design challenges, (3) area, (4) performance, and (5) energy efficiency. We evaluate a big.LITTLE processor (as a representative of single-ISA heterogeneous processor) based on Cortex-A15/A7 and a WDR processor based on Cortex-A15 running various mobile and SPEC2006 benchmarks. Our experiments demonstrate that the WDR processor combined with DRS can deliver energy efficiency close to the single-ISA heterogeneous processor, depending on the power overhead of circuit implementation to provide the wide DVFS range.**

I. Introduction

Todays mobile devices are required to execute various applications, which results in diverse platform demands. Form factor limitations restrict the battery capacity. Consequently, to provide longer battery life and to meet different performance demands, a client device must operate efficiently across different power envelopes, and support both high-performance and low-power modes. Dynamic voltage/frequency scaling (DVFS) has been a powerful technique to enable energy efficient computing [1]. However, with technology scaling, the available voltage/frequency (V/F) range for commercial processors has been continuously decreased. This is mainly due to the challenges incurred in scaling the minimum operating V, specifically for on-chip memory [2] [3] [4]. Consequently, future mobile processors may not be able to support a wide V/F range without facing significant area and/or power overhead [5] [6] [7].

Two promising approaches can deliver a wide dynamic power/performance range for mobile platforms without exclusively relying on DVFS. The first approach, dynamic resource scaling (DRS), adjusts, i.e., scales processor resources to match workload characteristics by selectively activating on-chip memory and execution units [13]. DRS can be effectively combined with DVFS to render a wider V/F range. The second approach, on the other hand, provides a wide power/performance range by switching the execution between a complex out-of-order (OoO) core optimized for high-V, and a simple in-order core, optimized for low-V, in the form of a single-ISA heterogeneous processor such as ARM's big.LITTLE [8]. The resulting power/performance trade-off is comparable to a wide dynamic range (WDR) design, which extends DVFS utility by optimizing each core to support operation on a wide V range spanning high-V and low-V corners [9]. Single-ISA heterogeneous designs can also facilitate DVFS to extend the power/performance range further.

As a representative single-ISA heterogeneous processor for mobile applications, big.LITTLE architecture can effectively extend the dynamic power/performance range offered by homogeneous processor configurations. big.LITTLE couples little, i.e., relatively slow but low-power cores, with big, i.e., relatively fast and power-hungry ones. By switching between big and LITTLE cores, the operating V can be dynamically tailored to the computing needs. big.LITTLE architecture does not require major device- or circuit-level changes. However, software and (micro)architectural support should be provided to orchestrate on-demand switching of the workload between big and LITTLE cores at runtime. WDR architectures, on the other hand, optimize each core to enable operation on a wide V/F range. The wide range can cover both super-threshold and near-threshold regimes [9]. Unlike the single-ISA heterogeneous architecture, WDR variants do not require major (micro)architectural changes. However, WDR operation demands careful device- and circuit-level optimization to ensure reliable operation at low Vs (where the susceptibility to noise significantly increases), without sacrificing performance at high Vs.

In this paper, we compare and contrast a representative single-ISA heterogeneous design with a WDR processor augmented with DRS for energy efficient mobile computing. We consider the implications of both approaches for design

978-1-4673-7166-7/15/$31.00 ©2015 IEEE

complexity, device, circuit and architecture level optimization, and software support. We characterize performance, energy efficiency, and area overhead. For several mobile and SPEC2006 workloads, we analyze big.LITTLE as a representative single-ISA heterogeneous processor on pairs of ARM Cortex-A15/A7 cores, and WDR on ARM Cortex-A15. The key contributions of this paper are as follows:

- To the best of our knowledge, this is the first study to compare and contrast single-ISA heterogeneous and WDR architectures for mobile applications in terms of design challenges.
- We provide a detailed analysis of power and area overhead of single-ISA heterogeneous versus WDR designs.
- We demonstrate that single-ISA heterogeneous architectures can be as energy-efficient as WDR processors for both multi- and single-threaded applications, depending on the power overhead incurred by circuit support for wide dynamic range.

The paper is organized as follows: Section 2 explains how single-ISA heterogeneous and WDR approaches differ in providing a wide dynamic power/performance range. Section 3 discusses the design challenges of single-ISA heterogeneous versus WDR architectures. Section 4 provides our evaluation methodology. Section 5 covers the quantitative comparison of single-ISA heterogeneous and WDR approaches. Section 6 concludes the paper.

II. Approaches To Extend The Dynamic Operating Power/Performance Range

Under contemporary technology scaling, the efficiency of DVFS to enable energy-efficient computing is decreasing due to the diminishing dynamic V (and thus F) range. Dynamic resource scaling (DRS) and ARM's big.LITTLE architectures emerged as promising approaches to mimic DVFS utility for sustainable energy efficiency. In the below, we detail how these two approaches differ in providing a wide dynamic power/performance range when compared to WDR architectures designed to extend DVFS.

big.LITTLE represents a single-ISA heterogeneous system encompassing cores of different microarchitectures to form two different processor clusters: One cluster with four big, high performance OoO cores (e.g., Cortex-A15), and another one with four LITTLE, highly power-efficient in-order cores (e.g., Cortex-A7) [8]. To achieve the maximum energy efficiency, the software plays a critical role in determining the right cluster for a given task. Figure 1 shows two different software-based dynamic task scheduling policies: (a) cluster migration, (b) CPU migration [10].

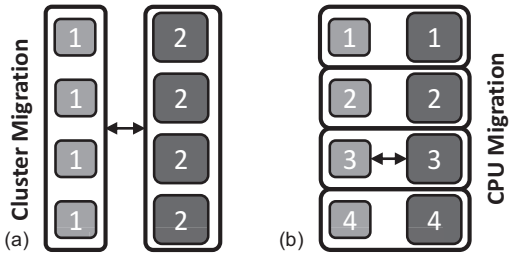


Figure 1: Task scheduling policies of big.LITTLE architecture: (a) cluster migration, (b) CPU migration.

Cluster migration from Figure 1(a) considers all four LITTLE cores as one cluster and all four big cores as another. During regular execution of the workload, only one cluster can be active at a time, thus the inactive cluster is power-gated. As the workload switches from one cluster to the other (i.e., during cluster migration), both clusters become active to communicate all relevant data over the last-level cache (L2). After migration, the operating system activates the desirable cluster, and power-gates the other. The migration time depends on the interconnect between big and LITTLE cores [11]. The power management unit (PMU) of the operating system switches clusters by monitoring the load at the cluster level. This policy is implemented as an extension to DVFS utility [10]: Any time the lowest V/F state of the big core renders sub-optimal energy efficiency, the workload switches from the lowest V/F state of the big core to the highest V/F state of the LITTLE core.

CPU migration from Figure 1(b), on the other hand, pairs each big core with one LITTLE core and exploits task migration within each pair to improve energy efficiency. Each physical big-LITTLE pair forms a single logical core. At any time, only one physical core (big or LITTLE) per logical core can be active. The inactive core of the pair is power-gated. During

Table 1: Design challenges of WDR versus big.LITTLE architecture.

	WDR	big.LITTLE architecture
Architecture	Same ISA	Same ISA
Core microarchitecture	One big OoO core	One big OoO core + one small in-order core
Fully associative resources	Larger SRAM cells; separate voltage domain	No change
Combinational logic	Larger devices; separate voltage domain	No change
Cache design	Separate supply voltage for caches; use of larger 6T or 8T/10T SRAM cells to operate reliably at ultra-low voltages	Two separate L2 caches for each cluster, interconnection network for cache coherency
Software support	No change	Task migration; policies to switch between clusters
Hardware overhead	PMU unit	Coherent interconnection network
Circuit level design	Multi-level shifters; power gating to proactively turn off idle resources	No change
Device level design	Larger devices	No change

execution of the workload, each logical core can switch from its LITTLE core to its big core (and vice versa) independently. During CPU migration, the operating system selects the appropriate physical core by monitoring the load level of each core separately, unlike (a). Whenever the scheduler determines to switch, the system powers up the power-gated core of the pair, communicates the architected state to the core to be activated, power-gates the currently active core, and continues execution on the newly activated core. This policy is also implemented as an extension to DVFS utility [10].

Dynamic Resource Scaling (DRS) can also mimic DVFS functionality, by compensating for the limited V/F range in multi-core processors [13]. The idea is selective activation, i.e., scaling, of the execution resources such as functional units and on-chip memories to match workload demands. To enable dynamic resource scaling, fine grain power gating is necessary. In this manner, exeution resources such as ALU, FPU, ROB, L1 or L2 caches can be controlled separately. A dedicated hardware PMU can be leveraged to implement such fine grain power-gating policies [12]. DRS can also be effectively combined with DVFS.

Wide Dynamic Range (WDR) design enables a processor to operate on a wide V range which can cover both the near-threshold and conventional, super-threshold regions [9] [12]. Unlike big.LITTLE, WDR does not require major (micro)architectural changes. However, to accommodate operation at ultra-low Vs, where reliability becomes a critical concern, WDR relies on complex device- and circuit-level optimization [12] such as providing multiple fine-grain V and power gating domains or multiple level shifting stages. In this case, a dedicated hardware PMU controls not only different V levels, but also clock and power gating schedules as a function of load conditions. WDR can facilitate DRS, as well. DRS in conjunction with WDR (WDR-DRS) demands similar power gating support as basic DRS.

III. Design Challenges of big.LITTLE versus WDR architectures

In the following, we tabulate the major sources of design complexity for big.LITTLE and WDR architectures. Table 1 summarizes our observations.

big.LITTLE architecture needs to ensure coherency of shared data upon migration. The simplest way to achieve this is to disable caches and to send all dirty data to (shared) memory over the AMBA3 bus [11], which incurs a high power and performance overhead. Instead, modern big.LITTLE designs rely on an AMBA4 coherent interface connected to a coherent interconnect (i.e., CCI-400) [11]. When compared to the AMBA3-based solution, the energy efficiency improves as cores can do useful work or enter a lower power state during migration. Other challenges include the sizing of LITTLE cores to deliver maximum possible energy efficiency while supporting very low-V operation at modest area cost. For example, the area of the LITTLE cluster of cortex-A7 is $1.8mm^2$ [14].

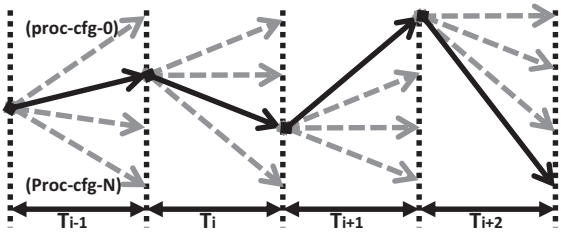


Figure 2: Oracular greedy optimization to find the most energy-efficient processor configuration per interval in big.LITTLE, WDR, and WDR-DRS architecture.

WDR architectures should protect the processor pipeline, the register file and on-chip memories against variation-induced errors due to the higher susceptability to variation at lower Vs, however, without compromising performance at higher Vs. This renders shallow pipelines for logic, and larger SRAM cells, possibly of higher number of transistors than the standard 6T [15] [16]. Even then, the minimum safe operating V of SRAM to exclude errors remains higher than that of logic, accordingly, operating memories at a higher V than logic renders higher energy efficiency [17]. A separate V domain for memories is necessary to achieve this [12]. Moreover, to facilitate data communication between different V domains, preferably configurable level shifters should be added. In this manner, clock signals can be synchronized across V domains, at all Vs covered by WDR. Such optimizations to support lower V operation incur an area overhead, which in turn results in a power overhead at higher Vs. The area overhead can reach 80%, where a comparable design optimized only for low-V operation incurs an area overhead of about 15% [9]. To mitigate the power overhead, fine grain clock or power gating can be adapted, increasing system complexity further. A dedicated PMU is necessary to orchestrate the assignment of V levels along with

Table 2: Summary of WDR and big.LITTLE architecture designs based on A15 and A7 processor configuration.

WDR architecture	big.LITTLE architecture		Memory Subsystem of WDR and big.LITTLE architecture	
Core type: Cortex-A15	Big core type: Cortex-A15	LITTLE core type: Cortex-A7	IL1, DL1	32KB/4-Way/64B 3 cycles
3-wide issue	3-wide issue	2-wide issue	IL1, DL1	32KB/4-Way/64B 3 cycles
160 register file	160-entry register file	32-entry register file	Coherency Protocol	Snoopy-based MESI
128 ROB	128-entry ROB	-	L2, LITTLE cluster	512KB/8-way/64B, 10 cycles
4 cores	4 cores in big cluster	4 cores in LITTLE cluster	L2, big cluster	2MB/8-way/64B, 12 cycles
Regular V/F states: (1.8GHz,0.9V)-(0.8GHz,0.65V)	Regular V/F states: (1.8GHz,0.9V)-(0.8GHz,0.65V)	Regular V/F states: (1.2GHz,0.9V)-(0.4GHz,0.65V)	L2, WDR	2MB/8-way/64B, 12 cycles
Near-threshold V/F states: (0.6GHz,0.6V)- (0.2GHz,0.53V)	-	-	-	-

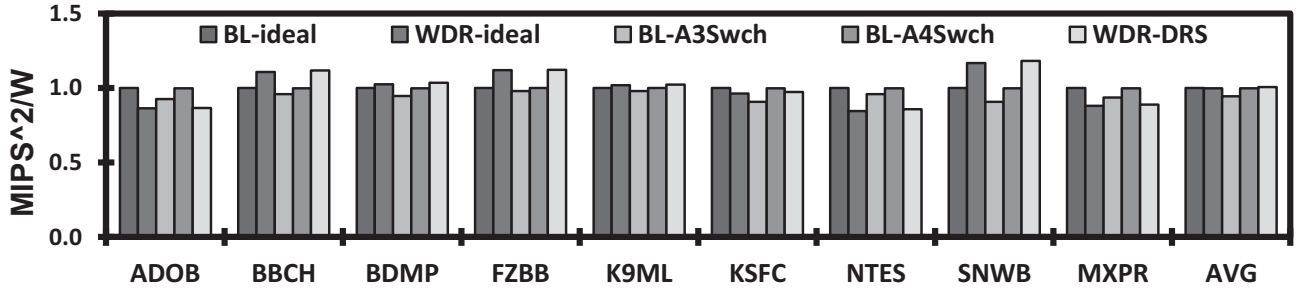


Figure 3: Energy efficiency of big.LITTLE (BL) versus WDR. BL-ideal, BL-A3Swch, and BL-A4Swch correspond to big.LITTLE with no switching overhead, switching overhead modeled after AMBA3, and AMBA4, respectively. WDR-ideal, and WDR-DRS ignore the power overhead incurred by a practical WDR implementation.

Table 3: WDR-DRS configurations

Configuration	Width	ALU	ROB	LSQ	L2 cache
Configuration 1	3	3	36	24	2 MB
Configuration 2	2	2	24	16	1 MB
Configuration 3	1	1	16	8	512 KB

deactivation of idle functional blocks or memory banks [12] [9]. Since low-V operation is more sensitive to any source of noise, V noise due to power management (i.e., power gating) should be taken into account. Similarly, controlling clock skew complicates placement and routing [12] [9].

IV. Evaluation Methodology

A. Power Management Algorithm

The power management algorithm is a key player in the evaluation of big.LITTLE and WDR architectures. The algorithm determines big/LITTLE configuration or DVFS state per execution interval. There are various methods to predict/find the optimal V/F state and core configuration for each interval, based on the execution history. To provide a fair and accurate comparison of big.LITTLE and WDR architectures, independent of the algorithm in use, we assume that the optimal configuration at every interval is known in advance. We employ oracular greedy optimization to extract the best core configuration (i.e., big or LITTLE) along with optimal V/F state for big.LITTLE; the optimal V/F state for WDR; and the optimal V/F state along with best configuration of active resources across all cores for WDR-DRS - all per execution interval (i.e., local optima). To find the configuration of maximum energy efficiency, we use $MIPS^2/W$ as our energy efficiency metric instead of $MIPS/W$ to emphasize performance.

Figure 2 demonstrates the execution trajectory under oracular optimization over consecutive execution intervals. For example, for WDR, in each interval we exhaustively simulate all possible V and F pairs. At the end of each interval, (i) we calculate the $MIPS^2/W$ for each V/F pair; (ii) we extract the V/F pair to deliver the maximum $MIPS^2/W$ in that particular interval; (iii) we checkpoint the architected state corresponding to the selected V/F configuration; (iv) we continue with the simulation for the next interval starting from the checkpoint from (iii). We repeat these steps for each interval, until the application terminates. A similar greedy framework applies for big.LITTLE and WDR-DRS to characterize the most energy efficient configuration on a per execution interval basis.

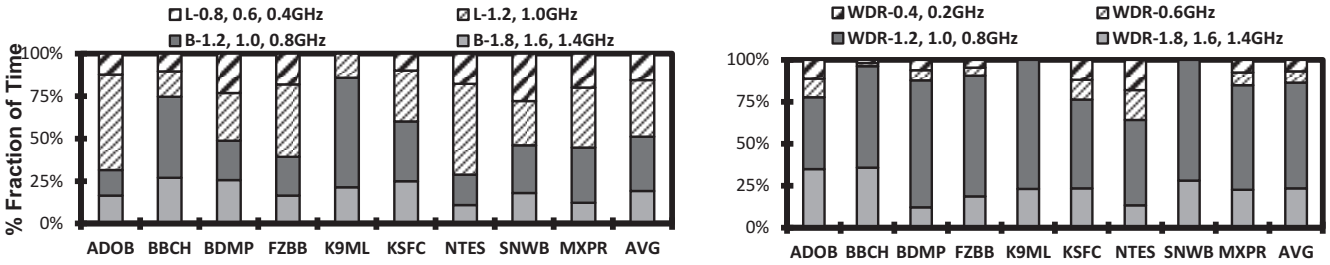
B. Architecture Simulation Environment

To evaluate WDR, we deploy a quad-core Cortex-A15. Each core is three-wide issue with 32KB private L1-D and L1-I caches, and a shared 2MB L2 cache with snoop-based MESI protocol. For WDR-DRS, we add new configurations to the WDR platform by scaling processor resources, such as ALU, ROB, LSQ, and L2 cache. Our platform for big.LITTLE architecture includes two clusters of four cores. The big cluster includes a quad-core Cortex-A15 (similar to the WDR configuration); the LITTLE cluster, four in-order Cortex-A7 cores. Table 2 summarizes the simulation parameters.

We use gem5 full system simulator [18] augmented with our oracular power management algorithm. We rely on McPAT [19] to estimate the power consumption of Cortex-A15 and Cortex-A7 configurations at 22nm. We have modified McPAT to support different V and F levels for accurate power estimation at different DVFS states. Table 2 includes all regular V/F states in WDR (to cover both super- and near-threshold regions), and big.LITTLE architectures. Since we target mobile platforms, we use moby benchmarks [20] that incorporate popular android applications from Google Play Store, and bbench benchmarks Adobe, Bbench, Baidumap, Frozen bubble, K9mail, King soft office, Net ease, Sina weibo, and Mxplayer (denoted by ADOB, BBCH, BDMP, FZBB, K9ML, KSFC, NTES, SNWB, and

Table 4: The LITTLE core configurations in big.LITTLE architecture.

Config.	issue width	# ALU	L2 (KB)	L1D (KB)	L1I (KB)	# INT Reg.	# Float Reg.
cfg1	2	2	512	32	32	32	32
cfg2	2	1	512	32	32	32	32
cfg3	1	1	512	32	32	32	32
cfg4	1	1	256	32	32	32	32
cfg5	1	1	256	16	32	32	32
cfg6	1	1	256	16	16	32	32
cfg7	1	1	256	16	16	16	32
cfg8	1	1	256	16	16	16	16



(a) (b)

Figure 4: Fraction of time spent in different V/F states of big.LITTLE (a) and WDR (b) architectures, to achieve the energy efficiency reported in Figure 3. The shaded boxes show the % time spent in LITTLE cores for big.LITTLE, and in NTV for WDR, respectively.

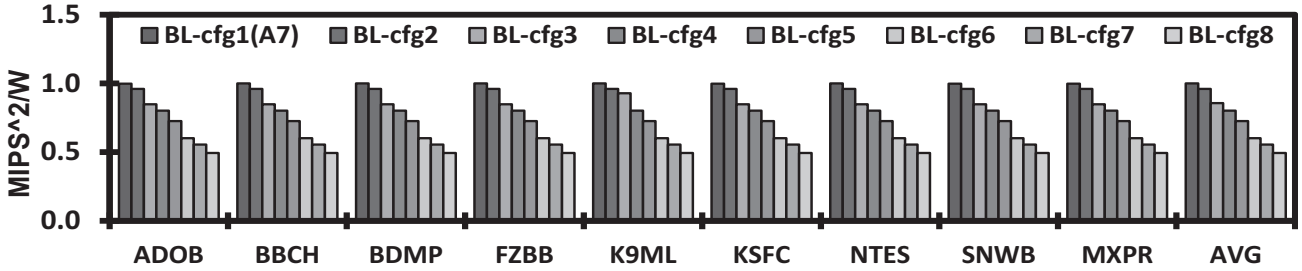


Figure 5: MIPS^2/W of BL implemented using a pair of cortex-A15 as the big core, and different versions of cortex-A7 as the LITTLE core.

MXPR). In addition, we study 10 representetive single-threaded applications from SPEC2006 benchmark suite [21], including both compute- and memory-bound benchmarks: bzip2, gcc, soplex, mcf, libquantum, lbm, milc, hammer, and astar.

V. Evaluation

In this section, we quantitatively compare and contrast the energy efficiency of WDR and big.LITTLE architectures in terms of $MIPS^2/W$.

A. Ideal big.LITTLE versus Ideal WDR

Figure 3 demonstrates energy efficiency in terms of $MIPS^2/W$ for our multi-threaded benchmarks. We consider different configurations: BL-ideal, BL-A3Swch, and BL-A4Swch correspond to big.LITTLE with no switching overhead, switching overhead modeled after AMBA3, and AMBA4, respectively. WDR-ideal, and WDR-DRS ignore the power overhead incurred by a practical WDR implementation. BL-ideal ignores both the power and performance overhead of switching a process between big and LITTLE cores. All data is normalized to the $MIPS^2/W$ of BL-ideal. Depending on the application, BL-ideal can achieve by up to 16% higher energy efficiency than WDR-ideal (for NTES). Similarly, WDR-ideal can achieve by up to 15% higher energy efficiency than BL-ideal (for FZBB). We observe that, ***on average, BL-ideal is as energy efficient as WDR-ideal although they take very different approaches in improving energy efficiency.***

BL-ideal and WDR-ideal achieve similar energy efficiency levels while operating at different V/F states. Figure 4 demonstrates the percentage of processor residency time in each V/F state under BL-ideal and WDR-ideal. Figure 4(a) shows the % time spent at four different V/F states for big.LITTLE. "B-1.8,1.6,1.4GHz" corresponds to p(ower)-states of the big core at 1.8GHz, 1.6GHz and 1.4GHz; "B-1.2,1.0,0.8 GHz", at 1.2GHz, 1.0GHz, and 0.8GHz, respectively. Similarly, "L-1.2,1.0GHz" corresponds to p-states of the LITTLE core at 1.2GHz, and 1.0GHz; "L-0.8,0.6,0.4GHz", at 0.8GHz, 0.6GHz, and 0.4GHz. Figure 4(b) captures the % time spent at the four V/F states for WDR. The high V states of WDR, "WDR-1.8,1.6,1.4GHz" and

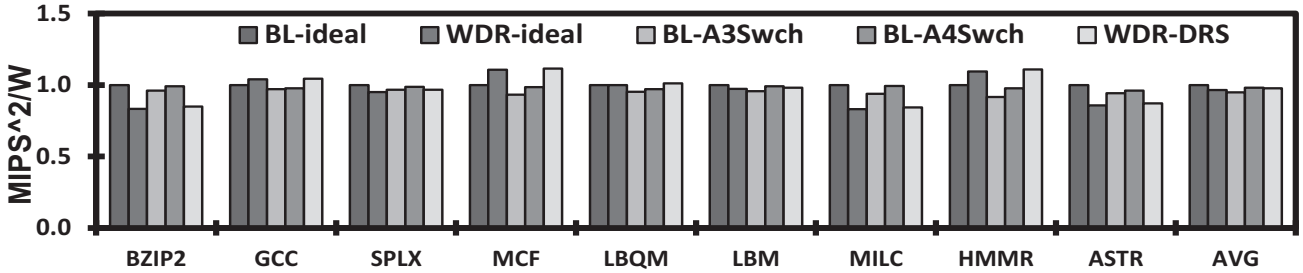


Figure 6: MIPS^2/W comparison of big.LITTLE (BL) architecture and WDR for single-threaded applications.

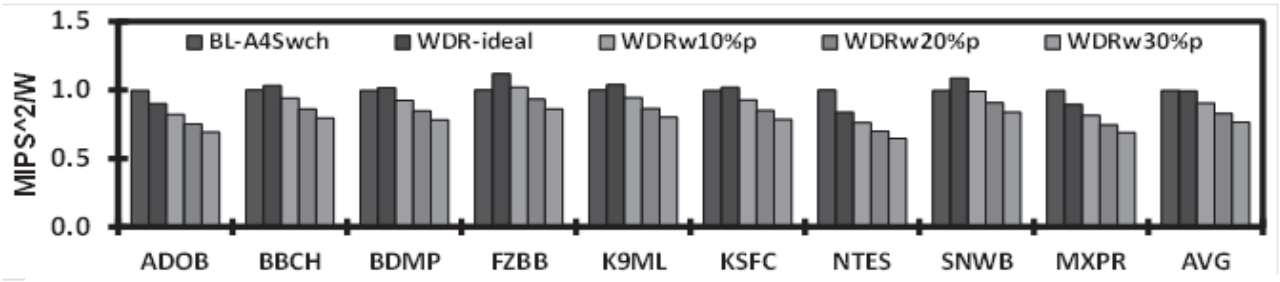


Figure 7: Sensitivity of MIPS^2/W to the power overhead of WDR of 10%, 20% and 30%.

"WDR1.2,1.0,0.8GHz", have direct correspondents in big.LITTLE. "WDR-0.6GHz" and "WDR-0.4,0.2GHz", on the other hand, demarcate the near-threshold V (NTV) states. We observe that, ***overall, WDR achieves higher energy efficiency by spending more time in high performance, high V states which incur higher power consumption than NTV states.*** On average, WDR spends 77% of time in these V/F states with F>=0.8GHz, and only 23% in NTV states with F<=0.6GHz. The power consumption in NTV states is low, however, so is the performance due to the low operating F. Accordingly, NTV states cannot improve the energy efficiency in most intervals. WDR boosts the energy efficiency mainly by running the applications faster in high V states most of the time, to compensate for the relatively higher power consumption. On the other hand, ***big.LITTLE achieves a similar level of energy efficiency by spending more time than WDR in lower power/performance states.*** The big.LITTLE architecture spends 52% of time in its big core V/F states, and 48% in the LITTLE core V/F states (i.e., using the LITTLE cores). Although the execution on the LITTLE cores takes longer, the power savings dominate. Thus, the overall energy efficiency improves. ***In terms of performance, WDR always outperforms the big.LITTLE architecture, on average by 14%.*** However, both achieve similar energy efficiency in terms of $MIPS^2/W$.

WDR with Dynamic Resource Scaling (WDR-DRS): Figure 3 characterizes the energy efficiency of dynamic resource scaling (DRS) at NTV states of WDR-ideal, WDR-DRS. To evaluate WDR-DRS, we added three new configurations, listed in Table 3, which represent an extension to the original WDR configuration (i.e., configuration 1). For configurations 2 and 3, we scale the issue width, the number of ALUs, the number of ROB and LSQ entries by a factor of 2/3 and 1/3 in comparison to the configuration 1, respectively. Although it is possible to apply DRS on both high V and NTV states, we confined our analysis to NTV states to conduct a fair comparison with big.LITTLE. The new NTV configurations include the baseline NTV states and the combination of each V/F state with each DRS configuration, resulting in 6 new NTV states. As shown in Figure 3, we observe that ***applying dynamic resource scaling on WDR (WDR-DRS) improves $MIPS^2/W$ by 2% compared to WDR-ideal, on average, across all configurations.***

Single-Threaded Applications: Figure 6 shows the energy efficiency of single-threaded applications from SPEC2006 under different big.LITTLE and WDR configurations. In big.LITTLE, only one big and one LITTLE core are active, hence the switching happens between these two cores. For WDR, only one (big core correspondent) core is active. We observe that ***the energy efficiency of single-threaded applications follows a similar trend as multi-threaded applications.*** On average, WDR-ideal and WDR-DRS achieve 2% and 1% lower $MIPS^2/W$ than BL-ideal, respectively.

B. Energy Overhead of big.LITTLE versus WDR

Impact of switching overhead of big.LITTLE architecture on MIPS^2/W: Figure 3 demonstrates the energy efficiency of big.LITTLE architecture when we account for the overhead of switching between LITTLE and big clusters. During the switching period, both clusters are on and consume power, although they do not execute the application. Switching impairs energy efficiency by consuming power during the transition, and by increasing the execution time due to the cluster migration. There are two mainstream implementations for inter-cluster interconnect: AXI interconnect using AMBA3 and CCI-400 interconnect using AMBA4. We studied the overhead under both. Figure 3 shows $MIPS^2/W$ of BL-A3Swch and BL-A4Swch that use AMBA3 and AMBA4 buses, respectively.

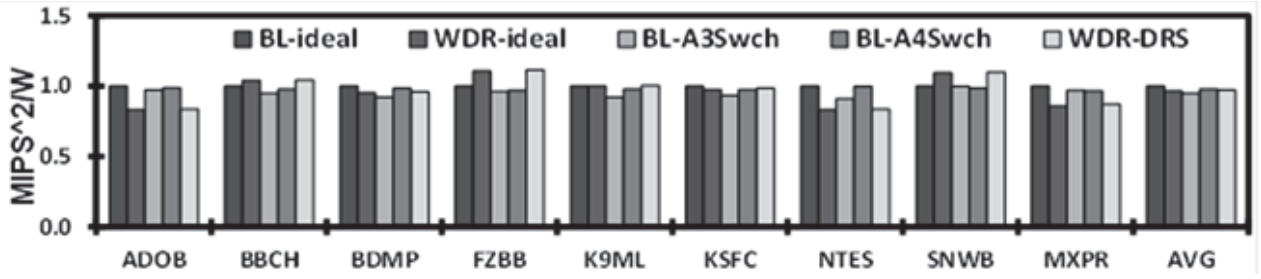


Figure 8: MIPS^2/W comparison of big.LITTLE (BL) supporting CPU migration and WDR supporting per-core voltage domains

The switching overhead under AMBA3 is higher than under AMBA 4, as it needs software intervention to orchestrate the writeback of dirty cache lines to memory. This takes on average about 1.4ms for a 2MB L2 cache when switching from the big to the LITTLE cores, and about 0.4ms for a 512KB L2 cache when switching from the LITTLE to the big cores. AMBA4, on the other hand, relies on hardware-based cache coherence which eliminates communication with memory during the transition, thus takes less time -- 33us on average. Accordingly, ***the switching overhead of big.LITTLE reduces its energy efficiency (under ideal conditions) by 4% (3%) for AMBA3 and 2%(1%) for AMBA4, on average, for multi- (single-) threaded applications,*** as depicted in Figure 3 (Figure 6).

Impact of the size of the little core on MIPS^2/W: In Figure 5, we study the impact of the size of the LITTLE cores on big.LITTLE's energy efficiency. We study eight different configurations, as shown in Table 4, where BL-cfg1 corresponds to the original cortex-A7. As shown in Figure 5, ***as the size of the LITTLE core decreases, the energy efficiency of big.LITTLE architecture degrades, resulting in upto 47% lower $MIPS^2/W$ when compared to BL-ideal.***

Impact of power overhead of a practical WDR implementation on MIPS^2/W: In Figure 7, we study the impact of the power overhead of a practical WDR implementation on the energy-efficiency. We study three different cases to incur 10%, 20% and 30% power overhead. As shown in Figure 7, ***as the power overhead due to implementation complexity increases, the energy efficiency of WDR decreases.*** It results in up to 25% lower $MIPS^2/W$ compared to the WDR-ideal.

MIPS^2/W comparison of big.LITTLE supporting CPU migration and WDR supporting per-core V domains: Up to this point, we analyzed a big.LITTLE architecture with cluster migration, as shown in Figure 1(a), and a WDR design with a single V domain. Figure 8 depicts the energy efficiency of big.LITTLE under CPU migration (Figure 1(b)), and WDR, under per-core V domains. We observe that ***the energy efficiency of big.LITTLE with CPU migration and WDR with per-core V domains follows a similar trend to big.LITTLE with cluster migration and WDR with single V domain.***

VI. Conclusion

This paper compares and contrasts single-ISA heterogenous and wide dynamic range (WDR) architectures in terms of energy efficiency, focusing on mobile applications. We observe that WDR processors facilitating dynamic resource sharing can deliver energy efficiency close to single-ISA heterogeneous processors, depending on the power overhead of circuit implementation to support wide-range DVFS.

VII. Acknowledgements

The authors would like to acknowledge Somayeh Sardashti, and Srinivasan Narayanamoorthy and our anonymous reviewers for their comments on the paper.

References

[1] C. Isci, A. Buyuktosunoglu, P. Bose and M. Martonosi, "An analysis of efficient multi-core global power management policies: maximizing performance for a given power budget," in *IEEE/ACM International Symposium on Microarchitecture (MICRO)*, 2006.

[2] J. Croon, S. Decoutere, W. Sansen and H. Maes, "Physical modeling and prediction of the matching properties of MOSFETs," in *IEEE European Solid-State Device Research Conference (ESSDERC)*, 2004.

[3] S. Ohbayashi, M. Yabuuchi, K. Nii, Y. Tsukamoto, S. Imaoka, Y. Oda, T. Yoshihara and M. Igarashi, "A 65-nm SoC Embedded 6T-SRAM Designed for Manufacturability With Read and Write Operation Stabilizing Circuits," *IEEE Journal of Solid-State Circuits (JSSC),* vol. 42, no. 4, pp. 820-829, Apr 2007.

[4] [Online]. Available: http://www.src.org/calendar/e003676/finalreport.pdf. 2009.

[5] S.-T. Zhou, S. Katariya, H. Ghasemi, S. Draper and N. Kim, "Minimizing total area of low-voltage SRAM arrays through joint optimization of cell size, redundancy, and ECC," in *IEEE Conference on Computer Design (ICCD)*, 2010.

[6] H. Ghasemi, S. Draper, N. Kim, "Low-voltage on-chip cache architecture using heterogeneous cell sizes for high-performance processors," in *IEEE International Conference on High Performance Computer Architecture (HPCA)*, 2011.

[7] S. Rusu, S. Tam, S. Muljono, H. Stinson, J. Ayers, D. Chang, J. Varada, R. Ratta and M. Kottapalli, "A 45nm 8-core enterprise Xeon® processor," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2009.

[8] P. Greenhalgh, "ARM," ARM, September 2011. [Online]. Available: http://www.arm.com/files/downloads/big_LITTLE_Final_Final.pdf.

[9] R. Gregory, D. S., J. S., and S. Vangal, "IA-32 Processor with a Wide-Voltage-Operating Range in 32-nm CMOS," *IEEE Micro,* vol.33, no.2, pp.28-36, 2013.

[10] M. Poirier, "Linux Foundation," 2013. [Online]. Available: https://events.linuxfoundation.org/images/stories/slides/elc2013_poirier.pdf.

[11] System.LSI, "Evaluation on Exynos.bL Processor," [Online]. Available:http://events.linuxfoundation.org/images/stories/pdf/klf2012_yu.pdf. 2012.

[12] S. Jain, S. Khare, S. Yada, P. Salihundam, S. Ramani, S. Muthukumar, A. K. S. M, S. K. Gb, H. Wilson, N. Borkar, V. De and S. Borkar, "A 280mV-to-1.2V Wide-Operating-Range IA-32," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 2012.

[13] H. Ghasemi and N. Kim, "RCS: runtime resource and core scaling for power-constrained multi-core processors," in *IEEE International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2014.

[14] K. Flautner, "Heterogeneity to the rescue," [Online]. Available: https://www.bscmsrc.eu/sites/default/files/media/arm-heterogenous-mp-november-2011.pdf. 2011.

[15] G. Chen, D. Blaauw, T. Mudge, D. Sylvester and N. Kim, "Yield-driven near-threshold SRAM design," in *," IEEE/ACM International Conference on Computer-Aided Design (ICCAD) 2007*.

[16] L. Chang, R. Montoye, Y. Nakamura, K. Batson, R. Eickemeyer, R. Dennard, W. Haensch and D. Jamsek, "An 8T-SRAM for Variability Tolerance and Low-Voltage Operation in High-Performance Caches," *IEEE Journal of Solid-State Circuits (JSSC),* vol. 43, no. 4, pp. 956-963, 2008.

[17] R. Dreslinski, B. Zhai, T. Mudge, and D. Sylvester, "An energy efficient parallel architecture using near threshold operation," in *IEEE International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2007.

[18] N. Binkert, B. Beckmann, G. Black, S. Reinhardt, A. Saidi, T. Krishna, S. Sardashti, K. Sewell, N. Vaish, M. Hill and a. D.Wood, "The gem5 simulator," *ACM SIGARCH Computer Architecture News,* vol.39, no.2, pp.1-7, 2011.

[19] [Online]. Available: http://www.hpl.hp.com/research/mcpat.

[20] M. C. Yongbing Huang, "Moby: A Mobile Benchmark Suite for Architectural Simulator," in *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, 2014.

[21] A. Jaleel, W. Hasenplaugh, M. Qureshi, J. Sebot, J. S. Steely and J. Emer, "Adaptive Insertion Policies for Managing Shared Caches," in *IEEE International Conference on Parallel Architectures and Compilation Techniques (PACT)*, 2008.