2014 IEEE Computer Society Annual Symposium on VLSI

Reducing Energy per Instruction via Dynamic Resource Allocation and Voltage and Frequency Adaptation in Asymmetric Multicores

Arunachalam Annamalai†, Rance Rodrigues‡, Israel Koren and Sandip Kundu
Advanced Micro Devices Inc., California†
Nvidia Corporation, Oregon‡
Department of Electrical and Computer Engineering, University of Massachusetts at Amherst
Email: arunachalam08@gmail.com†, {rodrigues‡, koren, kundu}@ecs.umass.edu

***Abstract*—With the advent of multicore processors the emphasis in computation is moving from sequential to parallel processing. Still, applications that require strong sequential performance do not achieve their highest performance/power when executing on current multicore systems. As the computational needs vary significantly across different applications and with time, there is a need to dynamically allocate appropriate computational resources on demand to suit the applications' current needs, in order to minimize the energy consumption. The Energy per Instruction (EPI) could be further decreased by dynamically adapting the voltage and frequency to better fit the changing characteristics of the workload. Not only can a core be forced to a low power mode when its activity level is low, but the power saved by doing so could be opportunistically re-budgeted to other cores to boost the overall system throughput. To this end, we propose a holistic solution to energy efficiency improvement by seamlessly combining heterogeneity, Dynamic Resource Allocation (DRA) and Dynamic Voltage and Frequency Adaptation (DVFA) capabilities to adapt the core resources to the changing demands of applications. Our results show that the proposed scheme provides an EPI reduction of about 17.9% when compared to the baseline heterogeneous multicore, 14% when compared to the baseline heterogeneous multicore with DVFA only and about 16.5% when compared to the baseline heterogeneous multicore with DRA only.**

I. Introduction

Technology advancements allowed more transistors to be packed in a smaller area while the improved performance helped in achieving higher clock frequencies. This, unfortunately, led to a power density problem, forcing processor industry to lower the clock frequency and integrate multiple cores on the same die [1], [2]. Most current multicores consist of many symmetric cores (SMP) with more modest computational capabilities that are suited for thread level parallelism. Hence, they lose out on performance when sequential or applications with high instruction level parallelism (ILP) are encounterd [3]. High performance for such applications could be achieved either by designing more powerful cores or by morphing the resources of the existing cores on-demand to suit the applications' current needs. The former approach results in some hardware resources being under-utilized, wasting static power when such applications are not encountered. Hence, on-demand resource morphing may be a better alternative.

It is well known that different workloads benefit from different computational resources. Even during the execution of a given workload, resource requirements may vary with time due to changes in program phases [4], [5]. Asymmetric Multicore Processors (AMPs) were introduced as a potential solution to this conundrum due to their capability to cater to the diverse needs of the workloads. A number of recent proposals have shown them to outperform their symmetric counterparts within a given power and area budgets [6], [7]. Hence, a heterogeneous multicore architecture with a dynamic resource allocation (DRA) capability would help in increasing the performance and utilizing the available resources more efficiently, resulting in reduced power consumption.

Power consumption could be further reduced by employing the widely used Dynamic Voltage and Frequency Scaling (DVFS) technique [8]. The voltage and frequency of a core can be lowered when it is idle or is in a low activity mode. For example, a memory bound application typically does not have sufficient ILP to keep the core busy while waiting for the long-latency memory accesses to complete [9]. Reducing the voltage and/or clock frequency of the core in such a case may result in significant power reduction without impacting the performance greatly [8].

We seamlessly combine the benefits of the above approaches and propose a heterogeneous architecture that could dynamically allocate execution resources (DRA) and adapt the frequency and voltage of the cores (DVFA) at runtime to suit the time dependent behavior of the workload. At a base level, we assume an AMP architecture with each core resourced moderately in all areas, while featuring extra-strength in a specific area (e.g., integer or floating-point operations). The strength of the cores is non-overlapping and hence, each core is suited for specific application characteristics. When a thread demands strength in more than one area, the cores are morphed dynamically by realigning their execution resources such that one core gains strength in one or more additional area(s) by trading its moderate resources for stronger resources of other core(s). Such morphing is not always the best solution, if a mismatch between the thread needs and the capabilities of the core executing it is discovered, a thread swap may provide a better alternative. Thus, our AMP architecture supports moving from the *baseline* mode of operation to the *morphed* mode, returning to the baseline, and also supports *thread swap*. Hardware monitors are used to determine the thread to core affinity during runtime and estimate the Energy/Instruction in each configuration of the multicore. Reconfiguration of the cores takes place at fine-grain time slices to minimize EPI. Dynamic core morphing together with dynamic thread swapping constitute the dynamic resource allocation (DRA) mechanism of the proposed scheme. The benefits are further increased with the added DVFA feature, where the frequency and voltage of the individual cores are changed (decreased or increased) dynamically in accordance with the workload behavior to minimize EPI while staying within the defined Thermal Design Power (TDP) limits. The key contributions of this paper are:

1) A holistic energy-efficient scheme that allows the AMP to morph the execution resources and/or operating conditions of the cores at runtime.
2) A novel hardware-based performance monitoring that triggers the reconfigurations and voltage/frequency adaptation to minimize EPI.

978-1-4799-3765-3/14 $31.00 © 2014 IEEE
DOI 10.1109/ISVLSI.2014.110

II. Related Work

Several reconfigurable multicore architectures have been recently proposed. Ipek *et al.* [10] have presented the concept of core fusion where the resources of several cores in a homogeneous chip multiprocessor (CMP) are fused to form a single stronger core at runtime. Kim *et al.* [11] presented another approach to fusion of homogeneous cores where, for example, 32 dual-issue cores can be fused into a 64-issue processor. Both schemes exhibit a high inter-core global communication overhead impacting the potential benefits of fusion. Salverda *et al.* [12] discuss the difficulties in achieving good performance by fusing simple in-order cores into out-of-order (OOO) cores.

Other references [3], [6] show that reconfigurable architectures may improve the benefit of AMPs even further. Das *et al.* [13] have proposed an asymmetric dual-core processor that could fuse a strong integer and a strong floating-point core. Their scheme is static so the cores are either fused or not for the entire program run. Static morphing of the cores can not suit all the different phases in an application. In [14], Rodrigues *et al.* presented dynamic morphing of cores at runtime to maximize performance/Watt. In comparison, our proposed scheme could both dynamically allocate resources and scale/boost the voltage/frequency of the individual cores. The DVFA capability of our scheme not only helps in reducing the energy consumption but also in efficiently redistributing the saved power to the other cores in order to increase the overall system's throughput. Moreover, improving performance and performance/Watt are targeted in [13] and [14], respectively, while the objective of the proposed scheme is to minimize EPI. Finally, we present an in-depth analysis of the impact of reconfiguration overhead in this work.

Dynamically scaling the voltage and frequency of cores in CMPs has been established as an efficient technique for power reduction [8]. Ghasemazer *et al.* [15] minimize energy consumption in CMPs by selectively turning ON or OFF the cores and choosing the optimum voltage and frequency for each core. Intel's Turbo Boost technology allows a core to run at a higher frequency automatically if the multicore is operating below a given rated power and temperature limits [16]. Similarly, AMD's Accelerated Processing Units (APUs) use the Turbo Core Technology to boost the frequency and performance of the cores staying within the defined power envelope [17]. Keramidas *et al.* [8] predict the performance and power consumption of cores and implement a DVFS scheme based on L2 stalls.

III. Proposed Scheme

The proposed scheme implements Dynamic Resource Allocation (DRA) and Dynamic Voltage and Frequency adaptation (DVFA) to provide adequate execution resources and determine the appropriate voltage and frequency of operation to suit the current needs of the workload. To illustrate our approach, we consider two heterogeneous cores (see Figure 1) per tile. A multicore system may consists of as many such tiles as deemed appropriate making the scheme scalable. The first core is a 2-way super-scalar INT core, with strong integer execution units but with low performance for floating-point operations while the second core (FP core) features strong floating-point execution units but low performance integer execution units. The reason for this example architecture is the diversity in the instruction type distribution of the common benchmarks. By focusing on the distinct strength of the integer and floating-point execution units, we would be able to efficiently service a wide variety of non-overlapping benchmarks in the baseline mode of operation.

In the *baseline* configuration (Figure 1), good performance is achieved by the cores while executing parallel workloads with ap-

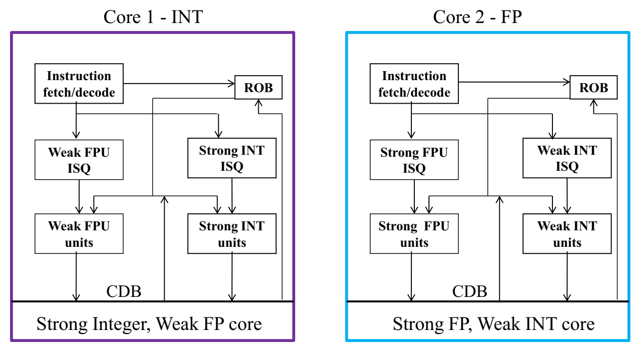

Fig. 1. Baseline configuration for two heterogeneous cores.

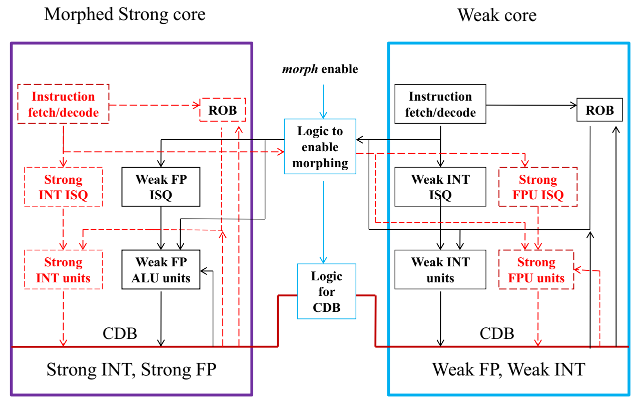

Fig. 2. Morphed configuration for two heterogeneous cores. The red dotted lines/boxes indicate the connectivity for the strong morphed core configuration and the black solid lines/boxes indicate connectivity for the weak core.

propriate resource requirements. However, when there is a need for a strong sequential performance by an application, dynamic resource morphing of the cores takes place. In the morphed mode, the INT core takes control of the strong floating-point unit of the FP core to form a strong "Morphed core" while relinquishing control of its own weak floating-point unit to the FP core. The FP core thus becomes a "Weak core." The stronger Morphed core retains the front-end resources of the INT core. In contrast, the front-end resources of the FP core are appropriately sized down to suit the reduced needs of the application running on the Weak Core in order to save power. Hence, morphing results in two cores: (i) a strong morphed core capable of handling both integer and floating-point intensive operations efficiently; and (ii) a weak core with weak functional units consuming less power. The proposed dynamic morphing of the cores is shown in Figure 2. When the morphed mode is no longer beneficial, the system reconfigures itself back to the baseline mode.

The behavior and characteristics of workloads tend to vary with time. Some applications may be floating-point intensive to start with and may have higher percentage of integer instructions after a certain point. Hence, swapping of the threads between the two baseline cores under such scenarios would help in reducing the execution time significantly [18]. Therefore, in addition to the baseline and morphed modes of operation, we also allow the two tightly coupled heterogeneous cores to swap their execution contexts. The DRA feature is a hardware-based scheme and is totally isolated from the Operating System (OS) level scheduler. Only the initial scheduling is assumed to be done by the OS and from then on, the thread to core assignment is managed autonomously by the DRA feature to minimize EPI at fine-grain time slices.

Furthermore, we leverage the DVFA feature to move each hetero-

TABLE I
EXECUTION UNIT SPECIFICATIONS FOR THE CORES [13] (P - PIPELINED, NP - NOT PIPELINED)

Core	FP DIV	FP MUL	FP ALU
FP	1 unit, 12 cyc, P	1 unit, 4 cyc, P	2 units, 4 cyc, P
INT	1 unit, 120 cyc, NP	1 unit, 30 cyc, NP	1 unit, 10 cyc, NP

Core	INT DIV	INT MUL	INT ALU
FP	1 unit, 120 cyc, NP	1 unit, 30 cyc, NP	1 unit, 2 cyc, NP
INT	1 unit, 12 cyc, P	1 unit, 3 cyc, P	2 units, 1 cyc,P

geneous core individually to either the Low Power (LP) mode or the High Performance (HPerf) mode by monitoring its performance and the frequency of memory reference operations. When the Instructions per Cycle (IPC) of the thread is consistently low (likely due to memory intensive operations), the proposed scheme moves the corresponding core to the LP mode. On the other hand, if the performance of a thread is high, then the corresponding core is moved to the HPerf mode if the other core is either already in the LP mode or is ready to enter the LP mode. Hence, entering HPerf mode is conditioned on the other core being in the LP mode. This is done to ensure that the TDP limit of the multicore is not violated. Adding the DVFA feature to the dynamic allocation of resources (through morphing and thread swapping) further reduces EPI.

IV. METHODOLOGY

The size of the floating-point and integer core parameters were taken from [14], as the proposed scheme is compared against their heterogeneous baseline configuration with *DRA-only* capability. The execution latencies for the cores were taken from [13] (see Table I). Similar to Intel's Turbo Boost technology, the frequency levels of the cores are changed in steps of 133 MHz [19]. In the default mode of operation, the cores operate at 1.1 V/2 GHz. The voltage and frequency level when the core enters the HPerf mode is 1.2 V/2.133 GHz while the voltage and frequency level of the cores in the LP mode is 0.9 V/1.734 GHz (frequency decreased by two steps). To illustrate our proposal, we have considered only three voltage/frequency levels, however, our scheme is scalable to many such levels.

We used SESC as our architectural simulator [20], and power is measured using Wattch [21] and CACTI [22] with modifications to account for *static power* dissipation. For our experiments, we have selected 38 benchmarks: 16 from the MiBench suite [23], 14 benchmarks from the SPEC suite [24], one benchmark from the Mediabench suite [25], and 7 additional synthetic benchmarks.

V. DRA AND DVFA MECHANISMS

DRA and DVFA features of the proposed scheme are accomplished using: (i) online monitors and, (ii) a performance predictor. The online monitors continuously and non-invasively observe certain aspects of the execution characteristics of the committed instructions and the activity of the cores. The performance predictor collects the observed application characteristics and determines whether to continue execution in the current configuration, or transition to another configuration. The possible transitions include dynamic reallocation of execution resources and/or DVFA.

A. Decision making at fine-grain time slices

Since prior knowledge about the computational needs of the applications is normally not available, online monitors are used to characterize the time dependent behavior of the workload. The key program characteristics that impact the performance/power are continuously monitored. Since power is not a property that can be extracted during runtime, we use other application attributes as proxy for power. We use hardware counters to monitor the instruction compositions (integer, floating-point and load-store). In the morphed mode, one thread executes on a stronger core, while the other thread executes on a weaker core. Consequently, the IPC of the thread that executes on a weaker core is limited. To avoid sacrificing the performance of this thread greatly, we monitor the IPC of the threads and assign a thread to the weak core only if its IPC is already low.

We exploit the 'memory boundness' of the program and when the core is busy with memory intensive operations and the IPC is low, we move the core to the Low Power (LP) mode where both the voltage and frequency are lowered to values mentioned in Section IV. On the other hand, if the IPC of the thread is high and the core is busy servicing compute intensive operations, the voltage and frequency of the core are boosted and the core enters the HPerf mode. However, our scheme allows the core to enter the HPerf mode only if the other core is either already in the LP mode or is ready to enter it. This way we keep a check on the TDP limit. Our proposed scheme has the capability to revert back to the default operating conditions when these modes are no longer beneficial, by monitoring the IPC of the threads.

The 'memory boundness' of the program is tracked through our monitoring of the frequency of the load/store instructions and is further strengthened by monitoring the LSQ occupancies of the cores. The hardware counters required for the online monitoring are similar to the ones used by Khan et al. [26] to keep track of instruction type distribution and IPC.

B. Offline Profiling

Offline profiling experiments were done to arrive at the suitable switching conditions for core reconfigurations and scaling conditions for DVFA. For the profiling experiments, 10 benchmarks from the suite of 38 were chosen such that they included some that (i) benefit from morphing or swapping (e.g., *fft, apsi, epic*), and some that (ii) did not benefit (e.g., *art, applu, equake*). The threads were then run for 100 million instructions on each core and at different voltage-frequency combinations. The instruction distribution, EPI, and LSQ occupancy were noted for a fixed number of committed instructions that we call a window (discussed later). To determine the switching conditions for core reconfiguration, two threads from the pool (of 10) were chosen and after every window the core configuration that yields the lowest EPI was noted along with the instruction distribution of both the threads during each window. For example, while running a combination of *apsi* and *applu*, if it is noticed (at the end of the window) that running *apsi* on the morphed core and *applu* on the weak core is better in terms of EPI, then this is marked as a potential switching point from baseline to morphed mode. Similarly, favorable switching points to come out of morphed mode and *thread swap* were also identified. These experiments were repeated for 60 random combinations of the benchmarks (out of 100 possible assignments) and possible trigger points for morphing, swapping and reverting to baseline mode were identified. Once the best core configuration was determined, the experiment was extended to choose the appropriate voltage-frequency level and the switching points to revert back to default operating conditions. The best EPI along with the corresponding instruction distribution and LSQ occupancy of the cores were noted at the end of each window.

The percentage of floating-point instruction (%FP), integer instructions (%INT), and load-store instructions (%LS), the LSQ occupancy and the IPC values obtained through these experiments were averaged out and used to establish the rules for dynamic reconfiguration (for both resource allocation and DVFA) shown in Figure 3.

```
Dynamic Resource Allocation:
1.  Threads $T_1$ and $T_2$ assigned randomly to cores
2.  Do Swap if:
     i.  ($\%INT_{FP} \geq 45$) and ($\%INT_{INT} \leq 27$)
                         OR
     ii. ($\%FP_{INT} \geq 27$) and ($\%FP_{FP} \leq 11$)
3.  Go from baseline to morphed mode if:
     i.  For $T_1$ ($T_2$ )
          a.  $\%(FP + INT) \geq 55$ and
          b.  ($14 \leq \%FP \leq 27$) and ($34 \leq \%INT \leq 47$)
     ii. And $T_2$ ($T_1$ )
          a.  $IPC \leq 0.4$ and
          b.  $\%(FP + INT) < 55$
4.  Come out of morphed to baseline mode if:
     i.  Thread currently on morphed core shows
          a.  $\%(FP + INT) < 50$
          b.  Use swap rules for thread to core assignment
5.  End

Dynamic Voltage & Frequency Adaptation:
1.  If ($IPC < 0.2$) and (($\%LS \geq 38$) or ($LSQ_{occ} \geq 0.57$))
     i.  Corresponding core enters LP mode
2.  If ($IPC \geq 0.7$) and (($\%LS < 26$) or ($LSQ_{occ} < 0.37$))
     i.  Mark the corresponding core suitable for HPerf mode
     ii. Enters HPerf mode only if the other core is in LP mode
3.  Return to default operating conditions:
     1.  If in HPerf mode, when IPC < 0.5
     2.  If in LP mode, when IPC ≥ 0.4

•   $\%INT_{FP}$  - Integer instruction percentage of thread on FP core
•   $\%INT_{INT}$ - Integer instruction percentage of thread on INT core
•   $\%FP_{FP}$   - FP instruction percentage of thread on FP core
•   $\%FP_{INT}$  - FP instruction percentage of thread on INT core
•   %LS    - Percentage of load and store instructions
•   $LSQ_{occ}$   - Fraction of load/store queue entries that are occupied
•   IPC    - Current IPC of the thread
```

Fig. 3. Dynamic resource allocation and Voltage/Frequency adaptation conditions for our scheme

C. Accounting for program phases

Based on the conditions mentioned in Figure 3, tentative decisions regarding the core configuration and the appropriate voltage and frequency levels are made at the end of every committed instruction window. In order to avoid too many reconfigurations (and their associated overhead), we prefer to wait until the new program execution phase has stabilized. To accomplish this, we base our reconfiguration decision on the most frequent tentative decision made during the n most recent instruction windows referred to as history depth.

We conducted a sensitivity study to quantify the impact of the window size and history depth on the quality of reconfiguration decisions. The best choice would be the combination that yields better EPI reduction over the entire program execution over the static baseline configuration (shown in Figure 1). To account for the fairness of the system, we have considered geometric EPI reduction in all our results in this paper. Different window sizes of 200, 250, 500 and 1000 committed instructions were considered and the history depth n was varied from 3 to 20. For each combination of window

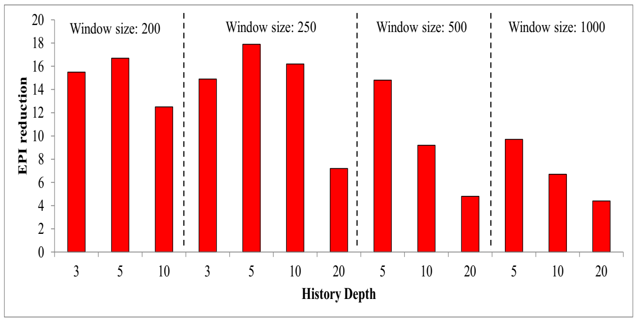

Fig. 4. Sensitivity analysis for determining window size and history depth.

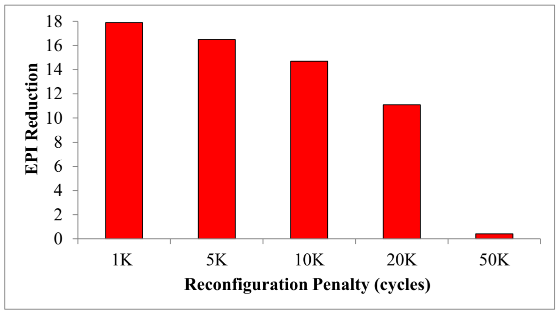

Fig. 5. Impact of reconfiguration overhead on achieved EPI reduction benefits over static baseline heterogeneous configuration.

size and history depth, about 100 random combinations of two-thread workloads were run, assuming an overhead of 1000 cycles per reconfiguration (as described in the next section). The EPI reduction obtained for each individual experiment was then averaged to give a single value that represents the entire set. From Figure 4, it could be seen that window size of 250 instructions and history depth of 5 provides the maximum EPI reduction of about 17.9%. Hence, we chose a window size of 250 instructions and history depth of 5 for our experiments. Our proposed scheme thus relies on the behavior of the threads during the last 1250 (250×5) committed instructions to make the reconfiguration decision about the execution resources and operating conditions.

D. Reconfiguration Overheads

In order to swap threads between the cores, we need to flush the pipelines, exchange architecture states and warm the caches. Similarly, if the frequency of the cores needs to be changed, the platform specific registers should be updated [9]. It is mentioned in [27] that with the use of on-chip voltage regulators, the voltage transitioning time is reduced to few tens of nanoseconds.

To quantify the impact of the reconfiguration overhead, experiments were run with 100 random combinations of benchmarks. The percentage EPI reduction over the static baseline configuration obtained with a overhead of 1K cycles was compared against those obtained with 5K, 10K and 20K cycles. As could be seen from Figure 5, the obtained EPI reduction benefits drop only by about 1.4% with a reconfiguration penalty of 5K cycles. Even when the reconfiguration overhead is as high as 20K cycles (about 10 μs for a 2 GHz processor), our proposed scheme was able to achieve an average EPI reduction of about 11.1% over the static baseline heterogeneous configuration. We observed that only with a penalty of 50K cycles per reconfiguration, the obtained EPI reduction benefits of the proposed scheme are almost nullified. With dedicated support for state swapping (e.g., Intel's Sandy Bridge [16]), far lower overheads can be expected and we used a reconfiguration overhead of 1000 cycles in our experiments.

VI. Evaluation

To illustrate the efficiency of our proposed scheme, we compare the EPI metric of our scheme against three baseline heterogeneous configurations – static, with DVFA capability only and with DRA feature only [14]. Besides the performance/power metric (EPI), we also evaluate the stand-alone performance in terms of weighted improvement in throughput using our scheme. From the pool of all 38 benchmarks, 100 random combinations of two benchmarks were chosen and run on the dual-core until completion of 100 million instructions. For the baseline configurations, the best thread to core

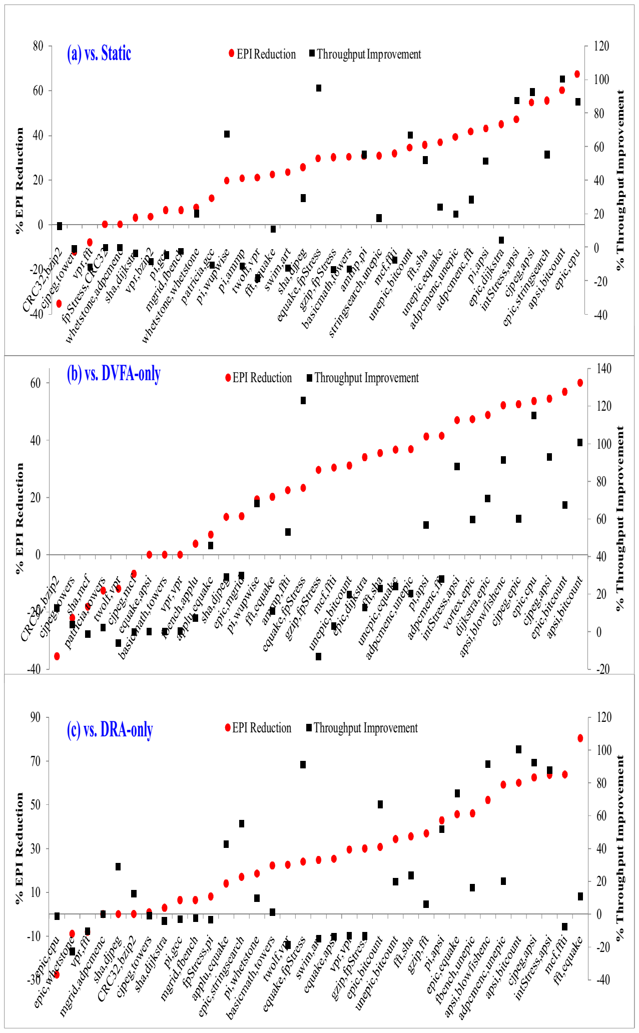

Fig. 6. EPI reduction and throughput improvement due to the proposed scheme over (a) the static baseline heterogeneous configuration, baseline heterogeneous configuration with (b) DVFA, and (c) DRA for different multiprogrammed workloads.

assignment was assumed to be known in advance while for the proposed scheme, a random initial thread to core assignment was made. The hope is that our scheme will detect the best assignment shortly after the programs begin to run.

For the sake of clarity, only 35 combinations (out of the 100) are shown in Figures 6(a), 6(b), and 6(c) which depict the geometric reduction in EPI (in percentage) and weighted improvement in throughput (in percentage) when using the proposed scheme against each of the baseline configurations. The shown 35 combinations include the 10 worse results (out of the 100), the 10 best results and 15 that showed average benefits.

As shown in Figure 6(a), significant reduction in EPI was obtained using the proposed scheme when compared to the static baseline heterogeneous configuration which lacks the capability to adapt to the time-varying behavior of the workload. The only scenario where the static configuration was able to achieve lower EPI than the proposed scheme was when no reconfiguration happened for the two-benchmarks pairs (for example, {*CRC32,bzip2*}, {*vpr,fft*}). Considering the throughput metric, our scheme performs worse than the static baseline for few combinations like {*vpr,fft*} (-19%), {*swim,art*} (-12%) as one of the cores operate in LP mode for most part of the execution to reduce EPI. Using the proposed scheme there was on average (for all the 100 combinations) a 17.9% reduction in EPI and 10% improvement in throughput as compared to the static baseline configuration.

As expected, relatively lower benefits were observed when comparing our scheme against the baseline configuration with either the DVFA only or the DRA only capability. These reference baseline configurations have some capability (either to change the voltage/frequency level or morph the execution resources) to adapt to the time-varying behavior of the workload. It could be noted that for few combinations like {*cjpeg,towers*}, {*sha,mcf*} in Figure 6(b) and {*epic,cpu*}, {*vpr,fft*} in Figure 6(c), the proposed scheme performs worse than the two baseline configurations in terms of EPI. There are three possible reasons for this: (i) The best thread-to-core initial assignment is assumed for the baseline configurations while it is random for the proposed scheme. This gives an added advantage to the baseline configurations when either no or very late reconfigurations happen using DRA. (ii) Due to the dual feature of DVFA and DRA in the proposed scheme, in a few rare cases the earlier transitions (either frequency/voltage adaptation or resource morphing) made by our scheme prevents some useful reconfigurations to happen in the future. For example, a decision to morph the cores could have been turned down by our scheme as the second core had been in the HPerf mode (and is about to come out of it) because of which its performance is slightly better than the defined conditions for a thread to be assigned to the weak core (refer to Figure 3). (iii) The scheme mispredicts. This could happen occasionally for any prediction scheme whose rules are determined by analysing a subset of applications.

We also noticed many combinations that stressed the need for a scheme to have the dual features of both DRA and DVFA. Consider for example, the pair {*basicmath,towers*} for which voltage and frequency were scaled twice during the program execution for both the proposed scheme and the baseline configuration with DVFA. Hence, no significant EPI reduction was seen for this benchmark pair when compared against the baseline with only DVFA. However, the baseline configuration with DRA was able to perform swapping of threads once for {*basicmath,towers*} while the two threads executed without any reconfiguration in the static baseline configuration. Hence, there was about 30.2% and 22.3% reduction in Energy/Instruction for the mentioned workload pair compared to the static baseline configuration and the one with DRA alone, respectively. There were few other workload pairs like {*sha, djpeg*} where the baseline configuration with DRA was able to follow the proposed scheme closely while significant benefits were achieved over the static baseline configuration and the one with DVFA alone. These examples illustrate the capability of our scheme to satisfy the diverse needs of different applications and enable a significant reduction in EPI.

Considerable benefits are achieved by the proposed scheme against both the non-static baseline configurations when workload pairs (like {*equake, fpStress*}, {*epic, bitcount*}, {*fft, sha*}) that require both DVFA and DRA to minimize EPI are encountered. Moreover, the number of combinations that benefit from the proposed scheme and result in a significant reduction in EPI is much higher than the number of those that do not (only 11% of the 100 combinations showed minor degradation compared to the baseline configuration with DVFA while the number of combinations that were slightly degraded was 9% compared to the baseline configuration with DRA). On average, for the considered 100 combinations, there was about 14% (15.5%)

and 16.5% (8.4%) reduction (improvement) in EPI (throughput) using the proposed scheme over the baseline heterogeneous configurations with only DVFA capability and only DRA feature, respectively.

VII. Conclusions

We have presented a novel energy efficient scheme that dynamically allocates computational resources and changes the voltage and frequency levels of cores at runtime. For illustration, we considered a dual-core: one core with support for strong integer code execution and another core that could handle floating-point operations efficiently. Aligning with the time-dependent behavior of the applications and their computational demands, our proposed scheme dynamically swaps the executing threads or morphs the cores at runtime by realigning resources of the given baseline cores to form a strong and a weak core. In addition, appropriate voltage and frequency levels are chosen dynamically to decrease EPI. Our results show that the proposed scheme with DRA and DVFA capabilities provides an average EPI reduction of about 17.9% when compared to the baseline heterogeneous multicore, 14% when compared to the baseline heterogeneous multicore with DVFA only and about 16.5% compared to the baseline heterogeneous multicore with DRA only.

VIII. Acknowledgement

This research was supported in part by grants 0903191 and 1201834 from the National Science Foundation.

References

[1] J. Held, J. Bautista, and S. Koehl, “From a Few Cores to Many: A Tera-scale Computing Research Review,” 2006.

[2] O. Khan and S. Kundu, “A model to exploit power-performance efficiency in superscalar processors via structure resizing,” in *20th Symposium on Great Lakes Symposium on VLSI*, ser. GLSVLSI ’10, 2010, pp. 215–220.

[3] M. Pericas, A. Cristal, F. Cazorla, R. Gonzalez, D. Jimenez, and M. Valero, “A Flexible Heterogeneous Multi-Core Architecture,” in *16th International Conference on Parallel Architectures and Compilation Techniques, 2007. PACT 2007*, Sept. 2007, pp. 13 –24.

[4] R. Kumar, K. Farkas, N. Jouppi, P. Ranganathan, and D. Tullsen, “Single-ISA Heterogeneous Multi-Core Architectures: The Potential for Processor Power Reduction,” in *36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003*, Dec. 2003, pp. 81 – 92.

[5] T. Sherwood, S. Sair, and B. Calder, “Phase tracking and prediction,” in *30th Annual International Symposium on Computer Architecture*, ser. ISCA ’03, 2003, pp. 336–349.

[6] M. Hill and M. Marty, “Amdahl’s Law in the Multicore Era,” *Computer*, vol. 41, no. 7, pp. 33 –38, July 2008.

[7] J. A. Winter, D. H. Albonesi, and C. A. Shoemaker, “Scalable thread scheduling and global power management for heterogeneous many-core architectures,” in *19th International Conference on Parallel Architectures and Compilation Techniques*, ser. PACT 2010, 2010, pp. 29–40.

[8] G. Keramidas, V. Spiliopoulos, and S. Kaxiras, “Interval-based models for run-time DVFS orchestration in superscalar processors,” in *7th ACM International Conference on Computing Frontiers*, ser. CF ’10, 2010, pp. 287–296.

[9] X. Zhang, K. Shen, S. Dwarkadas, and R. Zhong, “An evaluation of per-chip nonuniform frequency scaling on multicores,” in *2010 USENIX Conference on USENIX Annual Technical Conference*, ser. USENIX-ATC’10, 2010, pp. 19–19.

[10] E. Ipek, M. Kirman, N. Kirman, and J. F. Martinez, “Core fusion: accommodating software diversity in chip multiprocessors,” in *34th Annual International Symposium on Computer Architecture*, ser. ISCA ’07, 2007, pp. 186–197.

[11] C. Kim, S. Sethumadhavan, D. Gulati, D. Burger, M. Govindan, N. Ranganathan, and S. Keckler, “Composable Lightweight Processors,” in *40th Annual IEEE/ACM International Symposium on Microarchitecture, 2007*, Dec. 2007, pp. 381 –394.

[12] P. Salverda and C. Zilles, “Fundamental performance constraints in horizontal fusion of in-order cores,” in *IEEE 14th International Symposium on High Performance Computer Architecture, 2008. HPCA 2008*, Feb. 2008, pp. 252 –263.

[13] A. Das, R. Rodrigues, I. Koren, and S. Kundu, “A study on performance benefits of core morphing in an asymmetric multicore processor,” in *IEEE International Conference on Computer Design (ICCD)*, Oct. 2010, pp. 17 –22.

[14] R. Rodrigues, A. Annamalai, I. Koren, S. Kundu, and O. Khan, “Performance Per Watt Benefits of Dynamic Core Morphing in Asymmetric Multicores,” *In 20th International Conference on Parallel Architectures and Compilation Techniques, 2011. PACT 2011*, Oct. 2011.

[15] M. Ghasemazar, E. Pakbaznia, and M. Pedram, “Minimizing energy consumption of a chip multiprocessor through simultaneous core consolidation and DVFS,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, June 2010, pp. 49 –52.

[16] E. Rotem, A. Naveh, D. Rajwan, A. Ananthakrishnan, and E. Weissmann, “Power management architecture of the 2nd generation Intel Core microarchitecture, formerly codenamed Sandy Bridge, Hot Chips 2011, Paper: HC23.19.920,” Aug. 2011.

[17] D. Foley, M. Steinman, A. Branover, G. Smaus, A. Asaro, S. Punyamurtula, and L. Bajic, “AMD’s “LLANO” FUSION APU, Hot Chips 2011, Paper: HC23.19.930,” Aug. 2011.

[18] A. Annamalai, R. Rodrigues, I. Koren, and S. Kundu, “Dynamic thread scheduling in asymmetric multicores to maximize performance-per-watt,” in *Parallel and Distributed Processing Symposium Workshops PhD Forum (IPDPSW), 2012 IEEE 26th International*, may 2012, pp. 964 –971.

[19] “Intel Corporation. Intel Turbo Boost Technology in Intel Core Microarchitecture (Nehalem) Based Processors, White Paper,” 2008.

[20] J. Renau, “SESC Simulator, http://sesc.sourceforge.net.” January 2005.

[21] D. Brooks, V. Tiwari, and M. Martonosi, “Wattch: A framework for Architectural-level Power Analysis and Optimizations,” in *27th Annual International Symposium on Computer Architecture*, ser. ISCA ’2000, 2000, pp. 83–94.

[22] P. Shivakumar and N. P. Jouppi, “Cacti 3.0: An Integrated Cache Timing,Power, and Area Model,” Tech. Rep., 2001.

[23] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, “MiBench: A free, commercially representative embedded benchmark suite,” in *IEEE International Workshop on Workload Characterization, 2001*, Dec. 2001, pp. 3 – 14.

[24] “The Standard Performance Evaluation Corporation (Spec CPI2000 suite). http://www/specbench.org/osg/cpu2000.”

[25] C. Lee, M. Potkonjak, and W. Mangione-Smith, “MediaBench: a tool for evaluating and synthesizing multimedia and communications systems,” in *30th Annual IEEE/ACM International Symposium on Microarchitecture, 1997*, Dec 1997, pp. 330 –335.

[26] O. Khan and S. Kundu, “Thread Relocation: A Runtime Architecture for Tolerating Hard Errors in Chip Multiprocessors,” *In IEEE Transactions on Computers*, vol. 59, no. 5, pp. 651 –665, May 2010.

[27] W. Kim, M. Gupta, G.-Y. Wei, and D. Brooks, “System level analysis of fast, per-core DVFS using on-chip switching regulators,” in *IEEE 14th International Symposium on High Performance Computer Architecture, 2008. HPCA 2008*, Feb. 2008, pp. 123 –134.