AgileRegulator: A Hybrid Voltage Regulator Scheme Redeeming Dark Silicon for Power Efficiency in a Multicore Architecture

Guihai Yan[†], Yingmin Li, Yinhe Han[†], Xiaowei Li[†], Minyi Guo[‡], Xiaoyao Liang[‡]

[†]State Key Laboratory of Computer Architecture, Institute of Computing Technology, Chinese Academy of Sciences

[‡]Department of Computer Science and Engineering, Shanghai Jiao Tong University, China

{yan_guihai, yinhes, lxw}@ict.ac.cn, yingmin.li@gmail.com, {guo-my, liang-xy}@cs.sjtu.edu.cn

Abstract

The widening gap between the fast-increasing transistor budget but slow-growing power delivery and system cooling capability calls for novel architectural solutions to boost energy efficiency. Leveraging the fact of surging "dark silicon" area, we propose a hybrid scheme to use both on-chip and off-chip voltage regulators, called "AgileRegulator", for a multicore system to explore both coarse-grain and fine-grain power phases. We present two complementary algorithms: Sensitivity-Aware Application Scheduling (SAAS) and Responsiveness-Aware Application Scheduling (RAAS) to maximally achieve the energy saving potential of the hybrid regulator scheme. Experimental results show that the hybrid scheme achieves performance-energy efficiency close to per-core DVFS, without imposing much design cost. Meanwhile, the silicon overhead of this scheme is well contained into the "dark silicon". Unlike other application specific schemes based on accelerators, the proposed scheme itself is a simple and universal solution for chip area and energy trade-offs.

1 Introduction

Semiconductor road map shows power delivery and temperature will be more serious constraints to chip performance than the chip area as technology evolves. At 20nm technology node, it is estimated that roughly 20% chip area will become dark silicon [1], which cannot be turned on all the time to fully contribute to the chip performance. This projection indicates that putting more and more logics on a chip and expect higher performance is unsustainable if we do not pay attention to the energy efficiency.

As a major power management technique, Dynamic Voltage and Frequency Scaling (DVFS) promises to greatly save energy with only marginal performance loss, or significantly improve the performance with a given power budget. DVFS technique requires Voltage Regulators (VR) to provide variable voltages corresponding to different power states for each Voltage-Frequency Islands (VFI) in the microprocessor[2]. To achieve the best power efficiency, the DVFS operations should be able to respond to the transition of power phases in a timely way [3][4].

The response time of regulators virtually determines how fast the DVFS can operate. There are two kinds of regulators: the conventional off-chip VR and the more recently invented on-chip VR. Off-chip VR doesn't occupy chip area except power pins and the on-chip power network. It has higher power delivery efficiency, but is not as responsive as on-chip VR. By contrast, on-chip VR has much shorter latency to switch to a new voltage setting, but it has relatively lower power delivery efficiency and it dictates significant amount of chip area.

Unlike off-chip VRs which can only exploit the coarse-grain power phases, on-chip VRs are ideal to exploit fine-grain power phases. However, completely resorting to on-chip VRs are impractical due to two reasons: 1) Area overhead: It is estimated that to deliver 1 watt of power we need to pay $2mm^2$ chip area for the on-chip VRs [5]. Even though the regulator area can be partially amortized by the surge of dark silicon, it is still an overkill to pack per-core on-chip VRs which will occupy similar area as the cores on the chip. 2) Power delivery efficiency (DC-DC converter efficiency): Due to the physical constraints, the power transfer efficiency of the state-of-art on-chip VRs can hardly reach 80% [5] as compared to the 90% efficiency of off-chip VRs [6]. Therefore in some cases the energy benefit gained by using on-chip VRs can be totally offset by the loss of power delivery efficiency.

In this paper, we propose "AgileRegulator", a scheme leveraging dark silicon for a small number of on-chip VRs, rather than specialized computing logics or accelerators[7]. We might refer to the silicon taken as "Power Silicon" since the primary goal is for flexible power delivery and energy conservation. We demonstrate the hybrid scheme with mixed on-chip and off-chip VRs in a multicore architecture running multi-program applications. Since not all applications can benefit from fast DVFS, we only deploy a few on-chip VRs and other cores are still powered up by off-chip VRs. By combining the advantage of both on-chip and off-chip VRs through our smart DVFS and application scheduling algorithm, our scheme provides energy efficiency close to ideal per-core DVFS, with an increased area budget well fitted into the forecasted dark silicon.

[†]To whom correspondence should be addressed. The work was supported in part by National Basic Research Program of China (973) under grant No. 2011CB302503, in part by National Natural Science Foundation of China (NSFC) under grant No.(61100016, 61076037, 60921002).

In particular, this paper makes the following contributions:

1. We found that different cores within a single VFI often exhibit unbalanced program activities. Since each VFI can only have one off-chip regulator and that regulator has to accommodate the program whose performance is most sensitive to power settings, such core-to-core behavior variation will result in unfairness and the degradation of the overall energy efficiency in that specific VFI. In this paper we propose a novel application scheduling algorithm, "Sensitivity-Aware Application Scheduling (SAAS)", to mitigate the unfairness by dynamically grouping the applications with similar energy behavior into the same VFI, under the constraint of other system limitations such as memory bandwidth. This scheme greatly improves the overall energy efficiency for a system powered up by off-chip VRs.

2. Beyond SAAS, we propose to use the dark silicon area for a very limited number of on-chip VRs to maximally explore the fine-grain power phases for the most benefited applications. We show a smart algorithm "Responsiveness-Aware Application Scheduling (RAAS)" to identify and schedule such applications to use on-chip VRs. We also find that an on-chip VR is only helpful for a limited number of applications after considering the lower power delivery efficiency. Given the hefty area overhead, we advise to adopt the on-chip VRs judiciously and selectively.

3. We demonstrate the importance of building a synergy between SAAS and RAAS by combining their advantages and limiting their overhead. In some cases, SAAS may cause memory congestion in a VFI. When the fairness optimization conflicts with the memory bandwidth, we can leverage RAAS to balance the bandwidth utilization, while still maintain the level of fairness required by the SAAS. We evaluate our scheme on a 16-core, a 36-core, and a 64-core systems with multi-program workloads. Experimental results show that such a hybrid scheme can achieve an energy efficiency close to the ideal per-core DVFS case.

The rest of this paper is organized as follows: Section 2 gives the background and clarifies key motivations. Section 3 introduces the principle and framework of SAAS, RAAS, and the synergy between them. Section 4 introduces the key implementation heuristic. Section 5 describes the experiment setup and the workloads we used for this experiment. We show results in Section 6 and introduce related work in Section 7. Finally we summarize the paper in Section 8.

2 Background

2.1 Application Sensitivity to DVFS

Unlike most prior researches focusing on fairness issues introduced by shared resources in a multicore processor[8][9][10][11][12], we find that DVFS operations can also degrade system fairness and therefore result in energy inefficiency. The sensitivity of application execution latency (or time) to power states varies widely across applications. For example, Figure 1 compares

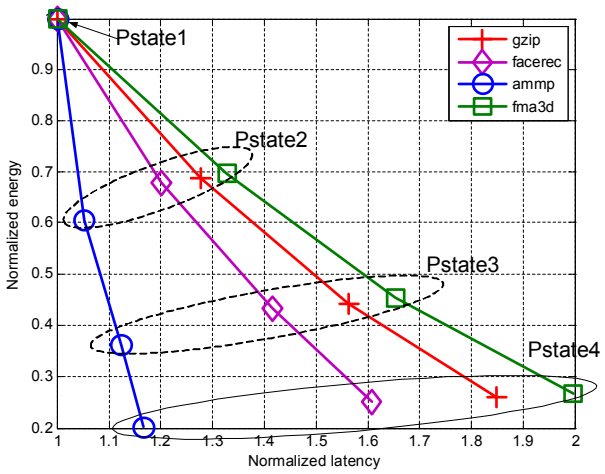


Figure 1. Energy vs. Latency within an execution epoch

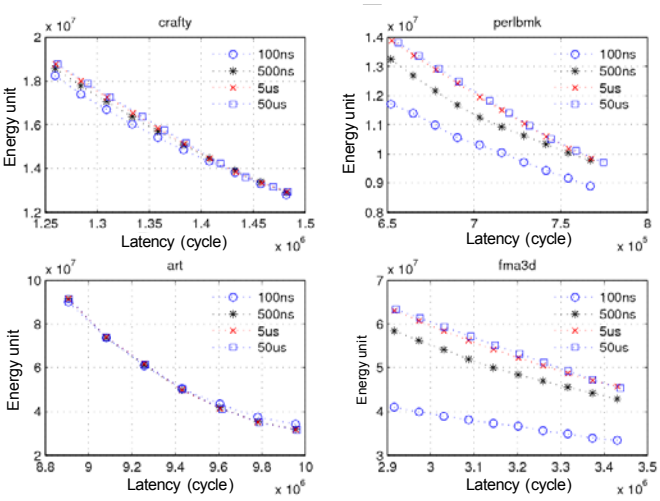


Figure 2. Energy vs. Latency pareto frontier under different DVFS intervals

the trade-off between execution latency and power states for different applications(`gzip`, `facerec`, `ammp`, `fma3d`). We have four power states with $Pstate1$ having the highest operating frequency and voltage. The execution latencies corresponding to all other power states ($Pstate2 - Pstate4$) are normalized to that of $Pstate1$. We find that the execution latency always varies with power states but to a different degree for different applications. The latency increase can vary from 17% for `ammp` to 100% for `fma3d`. Clearly, if the two applications reside in the same VFI powered up by the same voltage regulator, to meet the minimal system performance requirement, the power state has to be tuned based on `fma3d`, putting `ammp` into a sub-optimal operating mode. We call this as *DVFS-Induced Unfairness* in this paper.

2.2 Voltage Regulators

The key hardware support for DVFS operations relies on efficient voltage regulators, either off-chip or on-chip

[13][5]. The conventional regulator is either plugged onto the PCB board via a connector, or permanently embedded onto the board [14], referred as off-chip VR. The output voltage can be tuned according to processor operating mode/power state and typically ranges from 0.5V to 1.5V. A specific voltage is selected by programming a voltage identification register. The switching latency across power states is largely determined by the voltage tuning delay of the off-chip VRs. The latency is usually at the milliseconds range, which results in a relative slow DVFS response in practice. Each off-chip VR takes a large piece of board area and some amount of power supply pins of the chip. Because modern ICs are usually pin limited and the PCB boards are expensive, it is practical to pack only a few off-chip VRs onto the board.

By contrast, on-chip VR was invented for much faster response [13][5] with nano-seconds DVFS operations. However, this benefit does not come for free. Firstly, packing on-chip VRs into the silicon dictates huge area overhead. The area overhead is proportional to the required power delivery capacity. Empirically, delivering 1 watt power corresponds to $2mm^2$ chip area (the specific relationship varies with different technologies and types of regulators). That means up to $60mm^2$ silicon area is required to power up a 30 watt processor core in an Itanium® like processor where each core only takes up $75mm^2$. Clearly, the area of the on-chip VR is close to that of the core it powers. Since most of the regulator area is devoted to on-chip capacitors and inductors (metal), the area overhead is not likely to scale fast with advanced technology. However, the growth of dark silicon area provides us a perfect opportunity to use dark silicon as on-chip VRs for better voltage tuning. On the other hand, even with the dark silicon, the hefty area cost implies that on-chip VRs must be used very selectively to power up the most benefited parts of the microprocessor. Furthermore, the power delivery efficiency of on-chip VR is far from perfect. Recently, Kim et al. demonstrated a nanosecond-scale on-chip regulator with a peak efficiency staying at 77% [5]. On the other hand, the efficiency of off-chip VR is usually better than 90% [6].

2.3 Application Responsiveness to Fast DVFS

Not all applications require fast DVFS to achieve optimal energy efficiency. The application's preference to DVFS tuning latency is clearly demonstrated through analyzing the pareto optima of energy-latency tradeoffs. Figure 2 shows the energy-latency curve of four applications (`crafty`, `perlbmk`, `art`, `fma3d`) with different DVFS tuning latencies. The results show that the benefit of fast DVFS heavily depends on the characteristic of applications: for `crafty` and `art`, the fast DVFS brings little energy benefit. On the contrary, we can greatly improve the energy efficiency for `perlbmk` and `fma3d` with fast DVFS. In this paper, we call applications sensitive to DVFS latency as "sponge" applications, as they are like sponges releasing more water with more pressure (finer DVFS interval).

3 The Framework of "AgileRegulator"

"AgileRegulator" aims to build a synergy between two schemes, SAAS and RAAS, to combine the benefits of both off-chip and on-chip VRs. We assume that the target microprocessor consists of multiple clusters and each cluster consists of multiple cores. The hardware architecture is shown in Figure 3. Each core has a private L1 cache. The last-level cache can be logically shared by all of the cores within a cluster. Each cluster holds its own memory controller for the main memory access.

In our architecture, each cluster forms a VFI and is powered by an off-chip VR. Our scheme includes both on-chip and off-chip regulators. Given the hardware overhead of on-chip VR and the dark silicon area budget (around 20% chip area), we assume one on-chip VR is attached to each cluster. Only one core in a cluster can be opportunistically powered by the on-chip VR, depending on the characteristic of the application running on the core. When all the on-chip VRs are shut down, the target architecture degenerates to an off-chip VR only system.

In this section, we first describe the energy optimization problem in a multi-VFI processor and then explain how "AgileRegulator" can help boost the energy efficiency.

3.1 The Energy Optimization Problem for Multi-VFI Processors

In this paper, a *workload* is composed of multiple applications running on multiple cores. The applications are evenly divided into K consecutive epoches. At the beginning of each epoch, the applications will be rescheduled and re-assigned to VFIs. We assume a microprocessor has N VFIs and each VFI has M cores. $P(i,j)$ denotes the current power state of the ith VFI for the jth epoch. Given that each VFI hosts multiple applications, $D(i,j)$ is defined as the maximum execution latency across all the applications in the ith VFI for the jth epoch due to DVFS. $E(i,j)$ denotes the total energy consumption of all the applications in the ith VFI for the jth epoch. Equation (1) and (2) formulate these definitions.

$$D(i,j) = \max_{m \in \text{the } i\text{th VFI}} \{latency(m,j)\}, \tag{1}$$

$$E(i,j) = \sum_{m \in \text{the } i\text{th VFI}} \{energy(m,j)\}. \tag{2}$$

The energy optimization problem is to determine $P(i,j)$ for all VFIs over all epochs so that the total energy E_{total} is minimized, subject to the constraint that the maximum latency $D_{max} = \max\{\sum_{j=1}^{K} D(i,j)\}$ should not exceed the allowed maximum system latency ($Max_{latency}$). In summary, we have the following optimization problem:

$$\text{Minimize } E_{total} = \sum_{i=1}^{N}\sum_{j=1}^{K} E(i,j), \tag{3}$$
$$\text{Subject to } D_{max} \leqslant Max_{latency}.$$

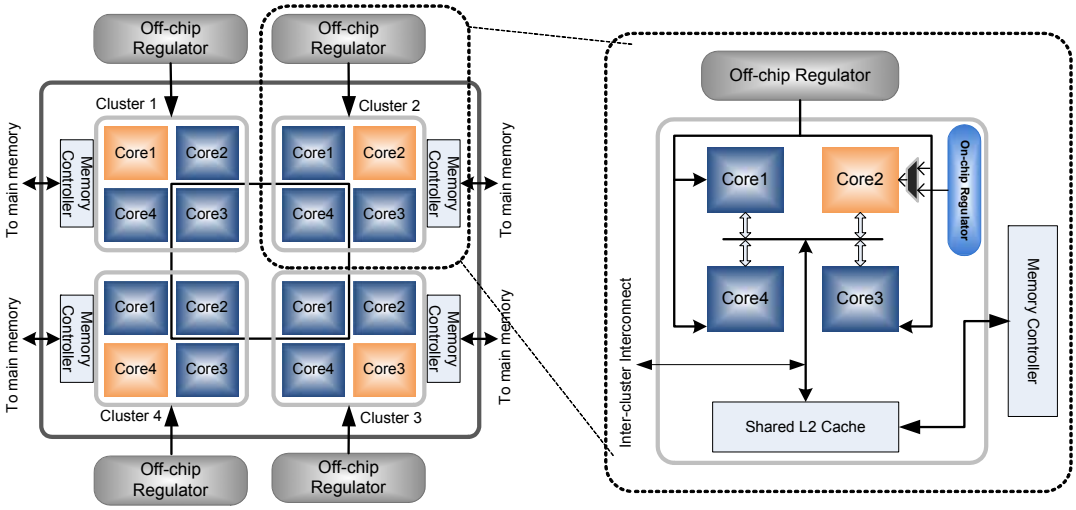


Figure 3. Top view of "AgileRegulator" architecture

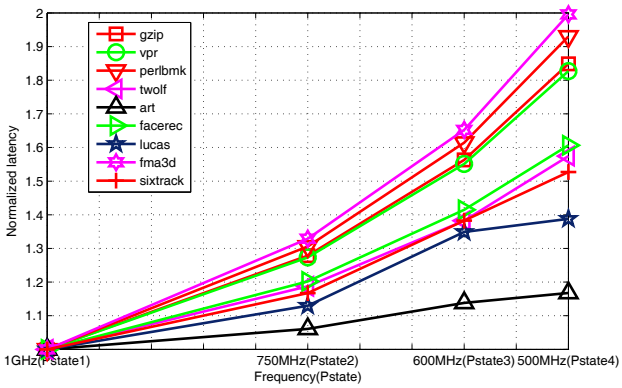


Figure 4. The S-factor for a $4-Pstate$ system

For simplicity, we use a relative $Max_{latency}$ definition, $D_{allowable}$, in the rest of this paper, which is defined as the ratio of actual execution time with DVFS to the running time at full-speed with the highest frequency/voltage setting. For example, $D_{allowable} = 105\%$ (or 1.05X) means the the system can accept 5% extra execution latency (or performance loss) after enabling DVFS for energy savings compared with the full-speed case.

3.2 SAAS: Sensitivity-Aware Application Scheduling

We define the power state with the highest voltage-frequency combination as $Pstate1$ and use it as the reference point. We then define the sensitivity of the execution latency associated with other power states, also called S-factor (S), for each epoch in any application:

$$S(Pstate1 \rightarrow PstateX) = \frac{Latency(PstateX) - Latency(Pstate1)}{Frequency(Pstate1) - Frequency(PstateX)}. \quad (4)$$

We use S-factor to characterize the fairness in a VFI. Figure 4 shows the execution latency with four power states for a bunch of SPEC benchmarks during an epoch of 10M instructions, where the latencies are normalized to the $Pstate1$. We observe that the latency change with power states varies greatly from benchmark to benchmark. For example, the latency of `art` only increases 17% from $Pstate1$ to $Pstate4$ while it varies 100% for `fma3d`. Such a big performance variation will seriously hurt the intra-VFI fairness if applications with big difference in S-factor are grouped into the same VFI. For a fair system with the optimal energy efficiency, the necessary condition is that there is as little intra-VFI imbalance on S-factor as possible. Otherwise, the selection of power state has to be bottlenecked by the application with the biggest S-factor, thereby wasting energy on other applications. This observation motivates the SAAS approach that is devoted to mitigate the intra-VFI imbalance based on the S-factors.

Figure 5 illustrates an example on how SAAS reduces the S-factor imbalance and in turn saves energy. In this example, we assume two VFIs, each hosting two applications during the kth epoch. The applications with dark color(`App2` and `App3`) have big S-factors, while applications with light color (`App1` and `App4`) have small S-factors. In the original case, the required frequency to meet $D_{allowable}$ is 1.6GHz for `App2` and `App3`. Without SAAS, the working frequency for both VFIs has to be set to 1.6GHz and the off-chip VRs have to select a high voltage (1.2V) to meet this frequency. If we take a further look at VFI1, this setting is inefficient because `App1` in that cluster is wasting power with the 1.6GHz/1.2V setting since its performance is insensitive to a lower power state. It can well meet the latency requirement with a much lower voltage and frequency setting. The same situation also applies to `App4` in VFI2. This is what we called S-factor imbalance in a VFI. Our proposed SAAS scheme can identify this situation and exchange the allocation of `App2` and `App4` to balance the intra-VFI S-factors. After the re-grouping process, the working frequency of VFI1 hosting `App1` and `App4` can be reduced to

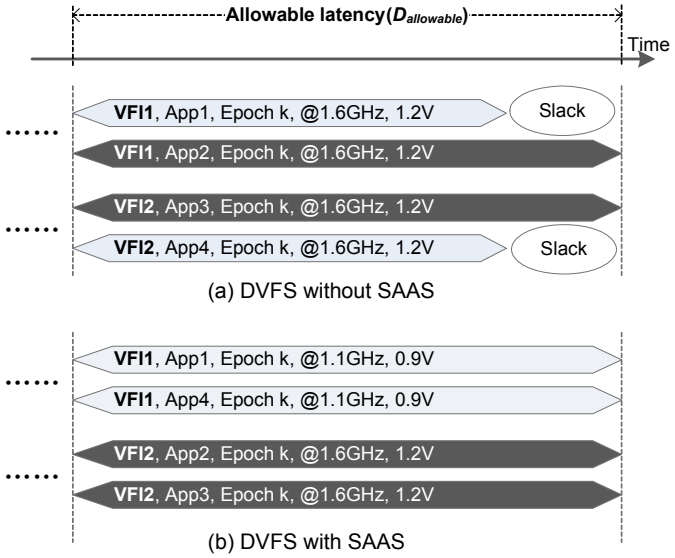


Figure 5. Balancing intra-VFI S-factor through SAAS

1.1GHz and the supply voltage can be reduced to 0.9V accordingly, saving a large amount of energy.

Bandwidth-Limited SAAS. The memory bandwidth for a VFI can be over-utilized if multiple memory-intensive applications are scheduled into the same VFI during the process of SAAS. It is clearly undesirable if the required memory bandwidth exceeds the service bandwidth of the memory controller for this cluster. As an essential optimization to the basic SAAS algorithm described above, we propose the bandwidth-limited SAAS to prevent the memory bandwidth violations. However, the bandwidth-limited SAAS is not a simple compromise of the S-factor balancing within a VFI. The memory bandwidth requirement and the S-factor balance can be both achieved in most cases by combining the SAAS technique and the RAAS technique presented in Section 3.4.

3.3 RAAS: Responsiveness-Aware Application Scheduling

The RAAS technique is designed to opportunistically migrate the most benefited application to the core equipped with on-chip VR for fast DVFS tuning instead of sticking to the slow off-chip VR which is shared by all other cores in a VFI, as shown in Figure 3. As stated in Section 2.3, `perlbmk` and `fma3d` are two sponge applications. Their energy efficiency is very sensitive to fast DVFS. For these applications, the fast DVFS can save up to 16% and 35% of energy without compromising the performance. Slow DVFS driven by off-chip VRs cannot provide this kind of energy savings due to the much coarser granularity of voltage tunings.

3.4 Combining SAAS and RAAS

We find that SAAS or RAAS alone cannot fully exploit the potential of energy savings. In some case, SAAS has to sacrifice the intra-VFI balance to avoid side effects like memory bandwidth contention. Coupled with RAAS, we can greatly reduce the side effects and balance both the memory bandwidth and S-factor. We find the combination of SAAS and RAAS can often achieve the effect of ideal per-core DVFS in terms of energy efficiency.

More specifically, if SAAS causes memory contention in a VFI after grouping applications with similar S-factors, we can exchange one memory-intensive application with a memory-none-intensive application in another VFI to relax the memory bandwidth requirement. However, the newly switched-in application may have very different S-factors that can potentially degrade the overall energy efficiency of both VFIs. To avoid this, the switched-in application will reside in the core powered up by the on-chip VR, without affecting the existing power state.

4 Implementation

4.1 SAAS and RAAS Heuristic

Searching for the optimal SAAS and RAAS strategy is the key technique for "AgileRegulator". We propose a practical heuristic that can be implemented into the hardware for this purpose.

Suppose at the beginning of the kth epoch, ith core is tagged with the ith application. There are altogether $P = N \times M$ applications in VFI_1, VFI_2, $\cdots$, VFI_N where each VFI has M cores. The S-factors of all the applications are denoted as S_1, S_2, $\cdots$, S_P, and their memory bandwidth as B_1, B_2, $\cdots$, B_P. The memory bandwidth threshold for each VFI is B_{th}.

• Step1: *Ranking S-factors*

$\{A_1, A_2, \cdots, A_P\} = RANK(S_1, S_2, \cdots, S_P)$, where A_1 is the index of the application with the largest S-factor in the system.

• Step2: *Initial Application Grouping*

$Group_1$: $\{A_1, A_2, \cdots, A_M\}$. The applications with the index of $A_1, A_2, \cdots, A_M$ are scheduled into the same group. This means applications with the largest M S-factors are scheduled into $Group_1$. Similarly, we have

$Group_2$: $\{A_{M+1}, A_{M+2}, \cdots, A_{2M}\}$

. . . . . .

$Group_N$: $\{A_{(N-1)M+1}, A_{(N-1)M+2}, \cdots, A_P\}$

• Step3: *Assigning Sponge Applications to Cores with On-chip VRs*

After SAAS, we step forward to RAAS. If there is no sponge application identified, the on-chip VR is not used and turned off. Otherwise, we schedule the sponge applications to the cores powered by on-chip VRs. If the number of sponge applications is larger than the number of on-chip VRs, those applications with higher IPC (Instruction per Cycle) take the priority, because those cores tend to be more power-hungry.

• Step4: *Easing Memory Bandwidth Contention*

The required memory bandwidth BW_i of $Group_i$ can be calculated by adding up the bandwidth of all the cores in the group. A bandwidth violation is identified if $BW_i > B_{th}$. The groups can be divided into two categories: without bandwidth violation denoted by a set

of $C_{w/o}$ and with bandwidth violation denoted by a set of C_w. If $C_{w/o} = \varnothing$, then there is no free bandwidth available in the system. We cannot do anything in such a case. However, if $C_{w/o} \neq \varnothing$, we can exchange the most memory-intensive application in the C_w with the most memory-non-intensive application in the $C_{w/o}$. The exchanged applications, if any, can only be scheduled to the cores with on-chip VRs. After this step, the cores with on-chip VRs may be occupied by switched-in applications that are not sponge at all. However, it helps ease memory bandwidth contention.

Clearly, the bandwidth violations cannot always be eliminated if a workload consists of too many memory-intensive applications, but our experimental results show that our bandwidth-aware scheme usually does not exacerbate the contention over the original case.

• Step5: *Scheduling the Application Groups*

To schedule an application group to a VFI, we use an algorithm that can reduce the chip thermal violation. Typically, the groups with large number of L1 misses consume less core power and run cooler. The algorithm always schedule such groups to VFIs with higher temperature so as to balance the chip thermal effect.

After Step5, all the applications find their residing VFIs and cores for the current epoch. The normal DVFS algorithm can be applied to both off-chip and on-chip VRs if applicable.

4.2 S-factor Regression and Inference

One essential knowledge required by our scheme is to identify the S-factor of each application dynamically for each epoch because SAAS relies on S-factors for dynamic grouping. We propose to leverage the regression tree approach to dynamically infer the S-factors of runtime workload based on a set of built-in performance counters.

Regression Tree is a non-parametric technique that recursively partitions groups into smaller subgroups that maximally differ on a desired outcome [15]. Regression tree approach is widely used to 1) predict occurrence of an outcome from a set of predictors, 2) detect threshold effects, and 3) predict censored data, etc. The inference for S-factor just matches the goal of applying the regression tree method. The first step of using the regression tree approach is to build a regression tree by determining a set of *if-then* conditions based on a training set. Those conditions imply a non-linear and complex interactions between predictors.

In this paper, because the application miss events have large impact on the performance degradation, we propose to use the following five miss events: L1 instruction and data cache(il1, dl1), unified L2 cache (ul2), instruction TLB and data TLB as the predictors.

The training set is built from the suite of SPEC benchmarks. Each training sample consists 1 million instructions. We use 100 samples randomly picked out from those benchmarks on SimPoint intervals. We find that 100 samples are enough to achieve a stable tree and avoid over fitting. Figure 6 shows part of the regression tree. Each leaf shows a possible regression result. Given a set

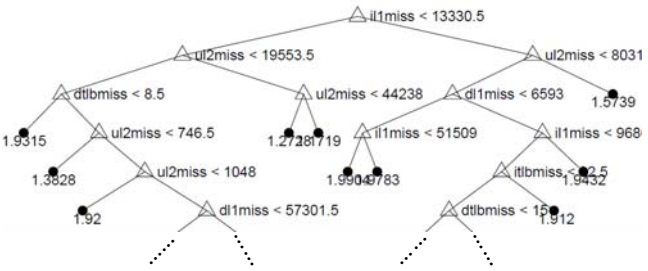


Figure 6. Using regression tree to infer S-factor

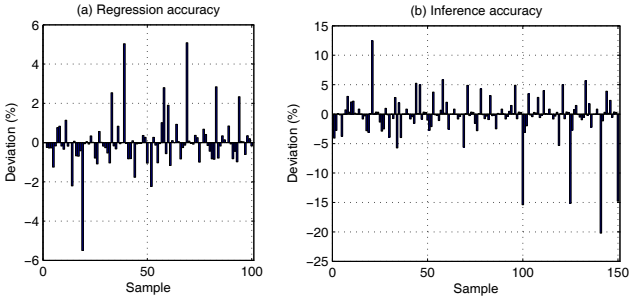


Figure 7. The accuracy of regression and inference for S-factor using the regression tree method

of predictors, the inferred outcome can be figured out by following a path indicated by the split conditions.

The accuracy of regression tree is shown in Figure 7. Figure 7 (a) shows the regression deviation for the training set itself. More than 90% samples have deviation less than ± 2%, and the rest of samples show deviation no more than ± 6%. More importantly, we use this tree to infer another 150 samples not included in this training set. The inference accuracy is over 95% for more than 90% samples. Only 3% samples show more than 10% deviation.

The hardware overhead for applying the regression tree is small, because the training process is conducted off-line. The hardware logic only needs to implement a set of *if-else* conditions for given predictors.

4.3 Identify Sponge Applications

We also need to identify sponge applications dynamically for the RAAS algorithm. We find that the pattern of the miss events has strong correlation with the responsiveness to a fast DVFS. Figure 8 shows part of results for dynamic miss events for two applications. In this figure, each sample is represented as a stacked bars corresponding to the amount of misses in L1 instruction (il1), L1 data (dl1), and L2 (ul2) cache. A sharp difference in the miss pattern between the two applications can be observed. the miss pattern of `fma3d` exhibits prominent periodicity, but the pattern of `crafty` is of erratic fluctuations. We find that `fma3d` is much more responsive to fast DVFS compared with `crafty`.

The on-chip VR has an intrinsic tuning bandwidth within which it can keep up with the pace of the changing environment. When the bandwidth of the application miss trace is too wide and exceeds the capability of

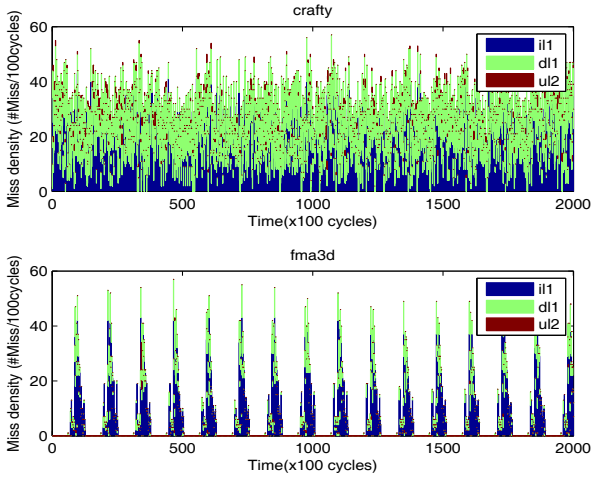


Figure 8. Misses distribution over time for `crafty` and `fma3d`

the regulator, the DVFS will fail to explore the power phases. If the bandwidth of the trace is less than half of the regulator bandwidth (i.e. meeting the Nyquist Sampling Theorem), DVFS has large opportunity to catch up the transition of power phases. In our study, the regular patterns (e.g. `fma3d`) usually have much narrower bandwidth than those white noise-like patterns (e.g. `crafty`). We therefore leverage the pattern recognizer to identify the sponge applications.

For hardware Implementation of RAAS, we propose a simple pattern-recognizer. We combine all the miss events into a single parameter "criticality" [16], defined by Equation (5).

$$Criticality = N(Miss_{L1}) + \frac{P_{L1L2}}{P_{L1}} \times N(Miss_{L1L2}) \quad (5)$$

where, $N(Miss_{L1})$ is the number of memory references that miss in L1 instruction or data cache but hit in L2 cache. $N(Miss_{L1L2})$ is the memory reference miss in both L1 and L2 cache. P_{L1} is the penalty (miss latency cycles) of L1 cache miss with L2 cache miss, and P_{L1L2} is the penalty of L2 cache miss.

The next step is to smooth the "criticality" with moving average operation, which serves as a typical low-pass filter to eliminate outlier disturbance. As Figure 9 (a) and (b) show, after the low-pass filter, `fma3d` exhibits regular pattern, while `crafty` still shows randomness. We further clip this waveform into a binary representation, as Figure 9 (c) and (d) show. Obvious pattern can be identified after the clipper as shown in the figure. We can use several counters to differentiate the repetitive patterns from erratic patterns.

5 Simulation Methodology

Simulation Setup. We use Wattch [17], a generic power simulator built on Simplescalar [18]. The core microarchitectue resembles the Alpha-EV6 processor, with separated L1 cache and shared L2 cache. The detailed configuration are shown in Table 1. We modify Wattch

Parameter	Value
Fetch/Issue/Commit width	4
Issue width	4
Functional units	4-Int/2-FP
Branch predictor	2K-gshare
L1 I-Cache(way)/Latency	64KB(2)/3 cycles
L1 D-Cache(way)/Latency	64KB(2)/3 cycles
L2 cache(way)/Latency	2MB(4)/12 cycles
ITLB,DTLB/Latency	128 entries/30 cycles

Table 1. Processor core configuration

to support four power states. The key focus in this modification is to separate the core portion (i.e. pipeline and L1 cache) from other non-core units such as L2 and I/O, because in practice only the core is enabled with DVFS. The frequency of the core is scaled according to power states. There are four available power states: $Pstate1$ (1.1V, 1GHz), $Pstate2$ (0.95V, 750MHz), $Pstate3$ (0.8V, 600MHz), and $Pstate4$ (0.65V, 500MHz).

We assume each VFI holds a DDR3 memory controller. To study the impact of memory bandwidth, we set the bandwidth threshold to different configurations ranging from 6.4GB/s of DDR3-800 SDRAM to 17.1GB/s of DDR3-2133 SDRAM. The bandwidth requirement of each VFI can be calculated from the total last-level cache miss rate (MR) [19]. One memory access triggers a memory bus transaction to fill a cache line. The last-level cache line size (L) is 128 bytes in our simulated machine. The theoretical bandwidth requirement can be calculated by $MR \times L$.

Voltage Regulators. The typical efficiency of off-chip VR is set to 90%. As for the on-chip VR, the efficiency is usually not constant and depends on the load. Prior study shows that the peak efficiency is about 77% at the optimal point [5]. When shifting from that point, the efficiency will gradually decline. Given that the constant efficiency is the ideal case people pursue and also for simplicity, we assume on-chip regulators can provide constant 75% efficiency. The default tuning latencies of off-chip and on-chip VR are set to 2ms and 100ns respectively, as suggested in [20].

For each 4-core VFI, we assume one on-chip VR. This roughly fits into the forecasted 20% dark silicon area in the 20nm process node given the area overhead for state-of-the-art regulators.

Power Model. Although Wattch is a powerful tool to investigate the performance and dynamic power in a reasonable amount of simulation time, it cannot generate accurate leakage power which accounts for considerable amount of overall power consumption. We approximate the leakage power as a fraction of the dynamic power [21]. By surveying the up-to-date industry data such as Intel® 8-Core Xeon® processors, fabricated using 45nm high-κ dielectric CMOS technology, the overall leakage accounts for about 16% of the total power at typical process corner [22]. Based on these data, we assume that leakage accounts for 20% of total power on average.

DVFS Algorithm. The major focus of this paper is to optimally schedule the applications to VFIs and cores physically equipped with different voltage regu-

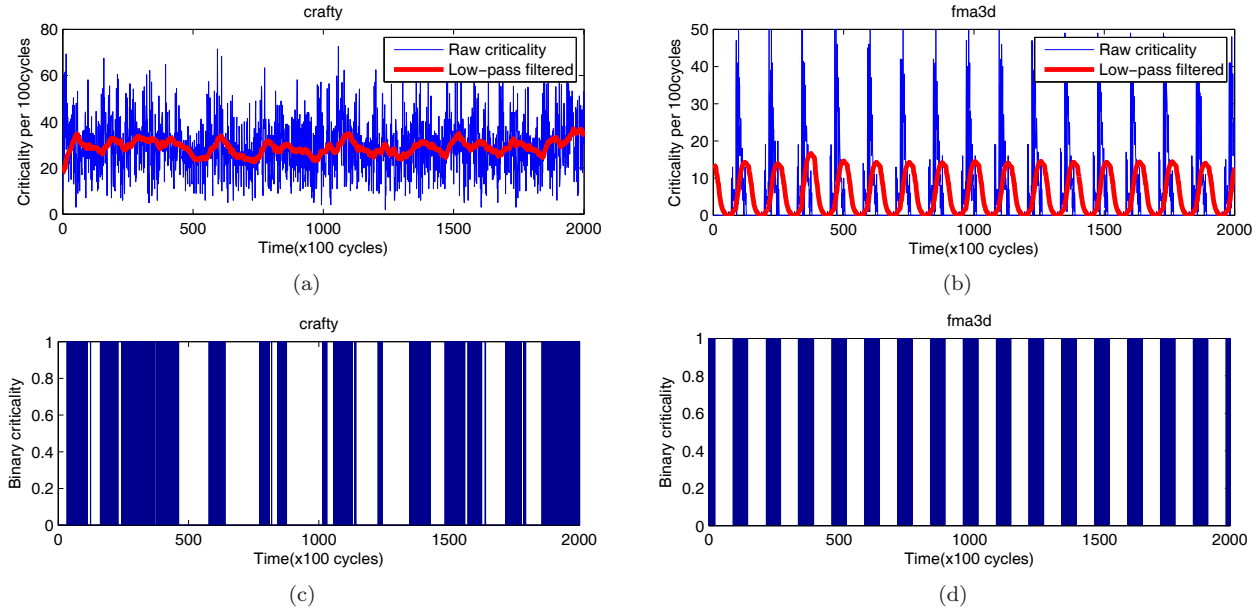


Figure 9. Working mechanism of pattern-recognizer

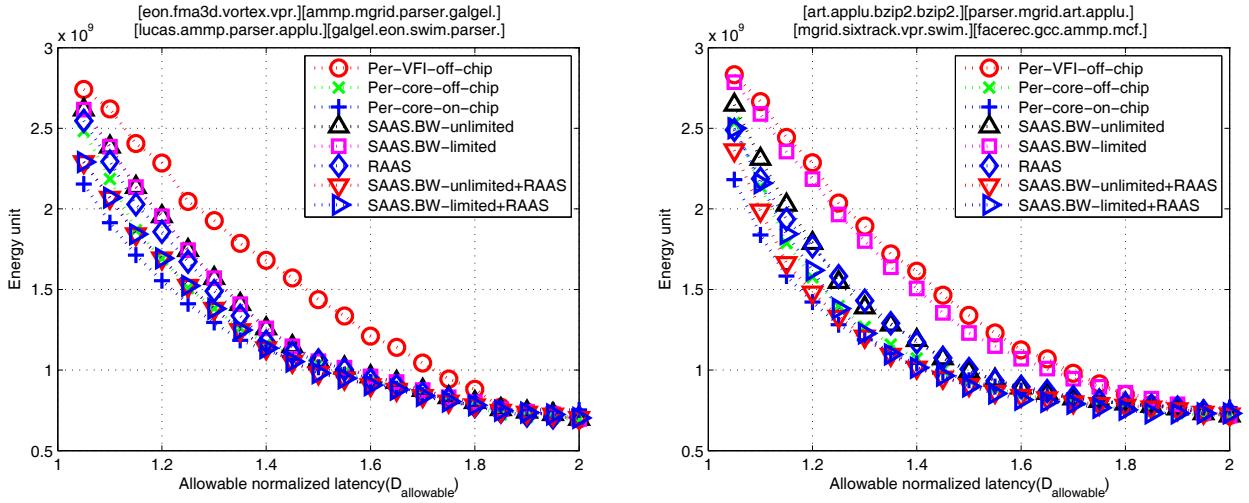
lators. Once the applications are allocated, we simply apply existing DVFS algorithms for energy tuning. We borrow an algorithm assuming power state configuration determined off-line [23]. But other solutions can also be adopted with very small degradation [24].

Workloads. A multi-program workload consists of multiple applications. The basic applications are chosen from all SPEC2000 CPU benchmarks; to add more diversity in the statistical evaluation we add 10 more SPEC2006 benchmarks (`bzip2, gcc, mcf, libquantum, omnetpp, astar, namd, soplex, povray`). We use SimPoint [25] to sample the simulation intervals based on standard single simulation points configuration. A whole application is 100M cycles and is divided into 8 epochs (i.e. 12.5M cycles for each epoch). If not specified, the benchmarks are randomly picked and combined into multi-core workloads. We randomly generate 100 different workloads in our simulation to evaluate the statistical efficacy. We also deliberately build memory-intensive and memory-non-intensive workloads to study the response of different optimization techniques. Other work such as [26][27] applies the similar method to build workloads. These workloads only intend to represent the multi-program environment, which is the primary focus of this paper.

Application Migration. Our scheduling scheme relies on application migration at the beginning of each epoch which has been proved necessary for multicore processors [28] to improve variation and fault tolerance [29]. Thread migration is not only engaged for thermal management [30], but also used to steer the applications running in a more power-efficient way on multicore processors [31]. Each migration involves transferring architecture states from one core to another. The performance and power penalty can be amortized well if the migration interval is kept long enough. Experimental results show that for a multicore processor with private L1 and shared L2 cache with a migration interval of 2.5M cycles, the worst penalty is only 5% and typically 2% for SPEC benchmarks [28]. The migration interval in our scheme is kept at 12.5M cycles and only imposes very marginal performance penalty (less than 1%). We therefore ignore the migration cost in the experiments.

6 Experimental Results

We first analyze the energy-delay pareto frontiers of "AgileRegulator" on a 16-core, 4-VFI system and show the performance of different techniques: 1) Per-VFI-off-chip is the baseline scheme where each VFI is power by one off-chip VR. The applications are randomly assigned to VFIs. 2) Per-core-off-chip assumes an impractical scheme where each core has its own off-chip VR. This structure can achieve coarse-grain per-core DVFS but is not implementable in the future many-core systems due to the large number of pin counts and PCB board cost. 3) Per-core-on-chip is also an impractical system that assumes per-core on-chip VR. This can achieve per-core fine-grain DVFS. Without considering the power delivery penalty, this is the oracle scheme that provides the best energy efficiency. It is not practical mostly due to the unacceptable silicon area cost. 4) SAAS.BW-unlimited assumes proposed SAAS scheme with unlimited bandwidth, hence SAAS never needs to compromise for memory bandwidth violation. 5) SAAS.BW-limited assumes the SAAS scheme under limited memory bandwidth and trades off the VFI fairness for memory bandwidth alleviation. 6) RAAS is built from the baseline per-VFI-off-chip, and assumes one core of each VFI is equipped with an on-chip VR enabling RAAS. 7) SAAS.BW-unlimited+RAAS assumes the ideal combination of SAAS and RAAS without considering memory bandwidth limitations. 8) SAAS.BW-limited+RAAS combines SAAS and RAAS with practical bandwidth requirement. Among all the techniques described in this



(a) Memory non-intensive workload

(b) Memory intensive workload

Figure 10. The pareto frontier for Energy-delay tradeoff, 17.GB/s memory bandwidth

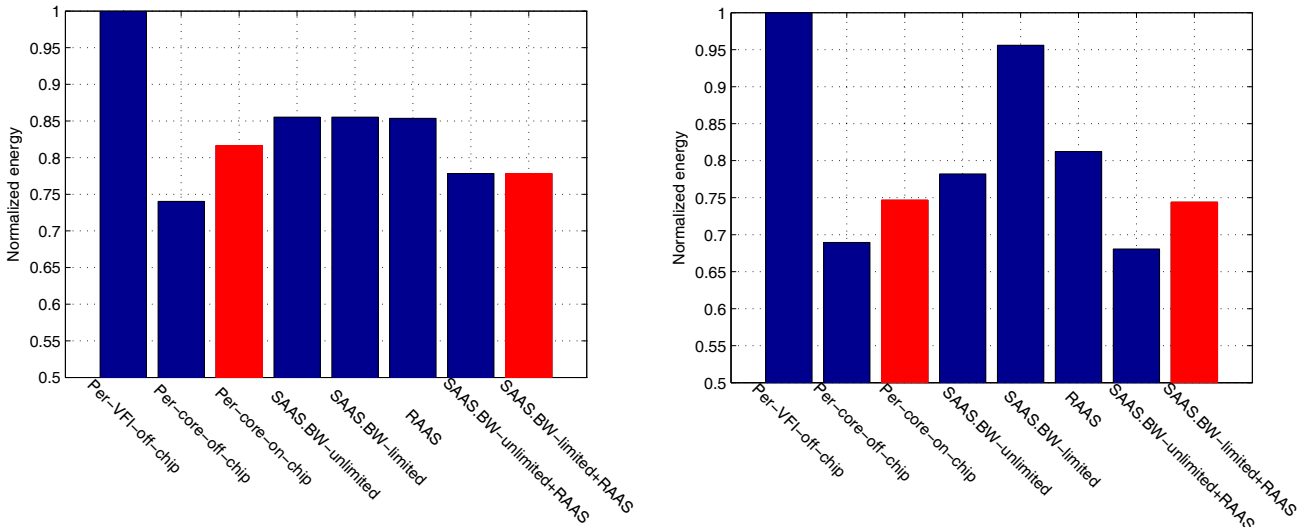


Figure 11. The normalized energy given 75% on-chip regulator efficiency and 90% off-chip regulator efficiency for 4x4 system, 20% allowable degradation

section, we highlight SAAS.BW-limited+RAAS because it is our preferred and practical hybrid solution.

6.1 Energy Latency Frontier Analysis

Figure 10 shows the energy consumption versus allowable application execution latency degradation ranging from 1.05X to 2X for various schemes. The workload shown in Figure 10 (a) is memory-non-intensive hence no bandwidth violations are detected on all VFIs. The second workload shown in Figure 10 (b) is memory-intensive and suffers from serious bandwidth violations. The applications forming a workload are marked at the top of each figure.

For the workload without bandwidth violations, SAAS alone can perform almost as well as the the Per-core-off-chip scheme, but still far from Per-core-on-chip. Per-core-off-chip cannot squeeze out the energy benefit from sponge applications due to the slow response time of off-chip VRs. Per-core-on-chip can fully leverage the advantage of fast-tuning in applications like `fma3d`, `sixtrack` and `facerec`. RAAS performs comparably with SAAS, but cannot beat Per-core-on-chip. However, the combination of SAAS and RAAS (SAAS.BW-limited+RAAS) performs close to the ideal Per-core-on-chip scheme.

For the memory-intensive workload, SAAS degrades badly due to the compromised fairness within a VFI. As Figure 10 (b) shows, SAAS.BW-limited can only perform as well as the baseline. RAAS alone, though working better, is still far from the Per-core-on-chip. However, SAAS.BW-limited+RAAS combines the benefits of both techniques and pushes the energy frontier close to the Per-core-on-chip case.

The above results related to RAAS are based on the assumption that on-chip VR has the same power transfer efficiency as off-chip VR. However, as we stated in

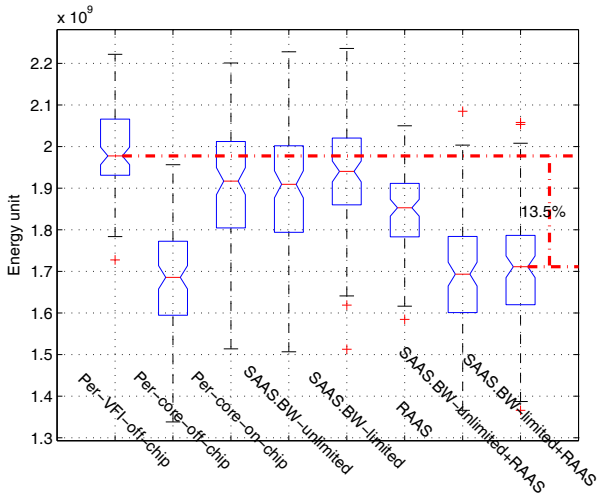


Figure 12. The statistic energy consumption compare with baseline scheme over 100 workloads for 4x4 system, 20% allowable degradation

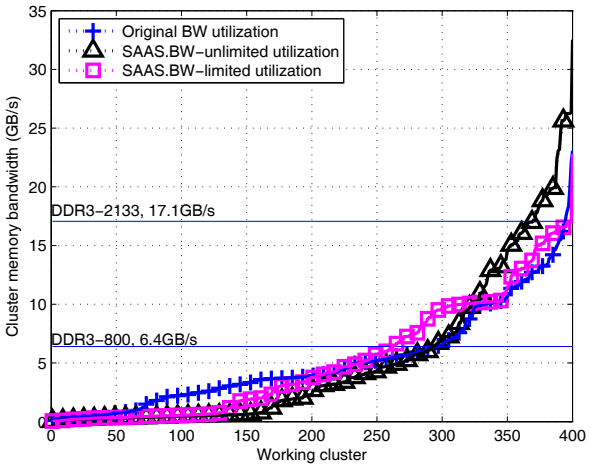


Figure 13. The memory bandwidth utilization over 100 workloads for 4x4-core system, 17.GB/s memory bandwidth

Section 2, current on-chip regulators cannot perform as well as the off-chip regulators in terms of power transfer efficiency. To deliver the same amount of power to the cores, on-chip VRs typically take an extra 15% of power from battery compared with off-chip VRs.

All the results provided in the following sections have taken the imperfect VR efficiency into account. We replot the results for the two workloads in Figure 11, where we highlight the Per-core-on-chip scheme with transfer loss and our preferred SAAS.BW-limited+RAAS scheme. The results show that the once oracle Per-core on-chip scheme cannot even beat the Per-core-off-chip for both workloads. We draw a conclusion here that the on-chip VR may not necessarily bring us the advantage of power savings over off-chip VR when considering the power delivery cost. Given the inherent area overhead of on-chip VRs, totally resorting to fast DVFS is not likely to be the right direction in the future processor design. For both cases, Per-core-off-chip exhibits great advantage because of its better power transfer efficiency and core-level corse-grain DVFS. However, Per-core-off-chip is also not a feasible solution especially in the future many-core processor due to the very limited power supply pins and PCB board area. The proposed SAAS.BW-limited+RAAS scheme is proven to be superior to all other schemes and it is practical for implementation with modern IC technology.

6.2 Statistical Effectiveness

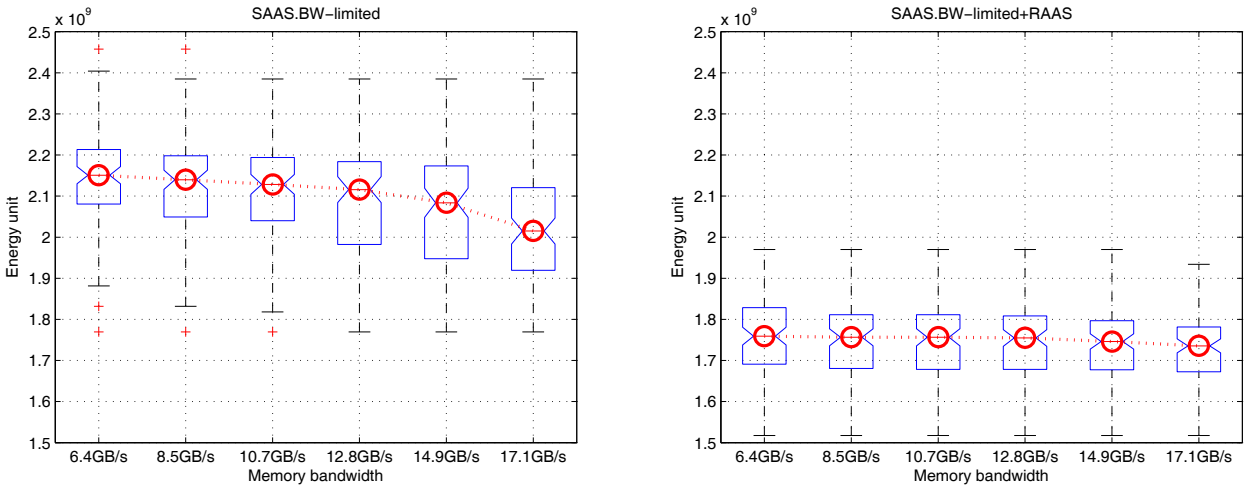
The above results only cover two specific workloads, the following results are devoted to study the statistical effectiveness of these techniques for 100 workloads randomly generated from SPEC2000 and SPEC2006 benchmarks. Figure 12 shows the energy using the eight techniques respectively with 20% allowed latency degradation. The results are presented using "boxplot" which is a statistical illustration used to show the median (the middle notch) and dispersion (the upper and lower bar) of a group of values (the outliers are denoted by "+"). Generally speaking, the proposed SAAS.BW-limited+RAAS scheme can boost the energy efficiency up to 13.5% comparing with the baseline VFI-based-off-chip scheme. This result is better than Per-core-on-chip and close to Per-core-off-chip in both cases.

6.3 The Impact of Memory Bandwidth

SAAS may change the memory bandwidth requirement in VFIs, as explained in Section 3.2. Figure 13 shows the memory bandwidth utilization for a 4-VFI processor across 100 workloads with each VFI holding 17.1GB/s bandwidth. We sort the bandwidth utilization from low to high for better observation. We find that for many cases SAAS.BW-unlimited utilization increases the demand for bandwidth in a specific VFI. This can be clearly identified at the tailing part of the curves ranging from index #300. But if SAAS was conducted in a bandwidth-aware manner, the memory bandwidth of the most memory-hungry VFI will be curbed to stay close to the original case. As a reference, we also mark the available memory bandwidth for two commercial DDR3 memory in Figure 13: DDR3-800 and DDR3-2133 which provide 6.4GB/s and 17.1GB/s bandwidth, respectively.

6.4 Sensitivity of Energy to Memory Bandwidth

Larger memory bandwidth creates more chance for SAAS to balance S-factors, hence leading to more energy savings. As figure 14 (a) shows, SAAS can statistically gain more energy savings with the increase in bandwidth. The dependence on memory bandwidth is greatly relaxed when combining with RAAS. As figure 14 (b) shows, SAAS.BW-limited+RAAS demonstrates an energy level insensitive to memory bandwidth. The overall energy consumption is significantly lowed compared with the SAAS only scheme. This again justifies the synergy between SAAS and RAAS.



(a) SAAS.BW-limited

(b) SAAS.BW-limited+RAAS

Figure 14. The impact of memory bandwidth to energy reduction, with SAAS.BW-limited and SAAS.BW-limited+RAAS

6.5 Scalability Analysis

The SAAS.BW-limited+RAAS technique is scalable with the number of cores. We show the results for 4x4, 6x6, and 8x8 systems in Figure 15. We still configure one on-chip and one off-chip VR for each VFI. We see that although the energy benefit is slightly reduced by 3% from 4x4 system to 8x8 system, the net energy reduction compared with the baseline is still over 10%. From the experience we estimate that an on-chip VR will take approximately the same area as a processor core. Therefore for the same technology node, the on-chip VR takes about 20% of the area of a 4-core cluster while it only takes about 12% area of an 8-core cluster. The large reduction in area overhead only degrades 3% benefit actually shows that our scheme is highly scalable.

With the continued technology scaling, the number of cores integrated onto a single chip will grow quickly according to the Moore's law. However, this is not the case for either on-chip or off-chip VRs. Due to the limited pin counts of the modern IC and the PCB board cost, it is not likely to accommodate large number of off-chip VRs. At the same time, since most of the area of on-chip VR is dedicated to on-chip capacitors and inductors and they are not scaling well with technology, it is also not likely to pack many more on-chip VRs even with the growth of the dark silicon area. This means we will have to leverage the slow-increasing regulators to deal with many more cores in the future. The configurations adopted in this study actually reflect the disproportionate scalability between the number of cores and the number of regulators.

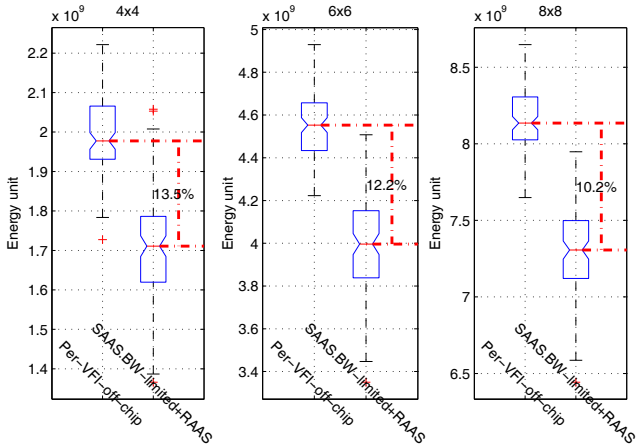


Figure 15. The statistical energy consumption compare with baseline scheme over 100 workloads for 4x4, 6x6, 8x8-core system.

7 Related Work

Power Phase. The adoption of DVFS in essence has to be based on the power phase prediction in terms of duration and power characteristics. Isci et al. found that the power phase can be accurately predicted in practice [3][32]. But they only focused on the coarse-grain (i.e. seconds granularity) power phase and ignored the fine-grain power phase which can be explored with fast DVFS operations. Our scheme studies both coarse- and fine-grain phases by incorporating both on-chip and off-chip VRs.

DVFS Facility. Rotem et al. studied the potential of multiple VFI for chip multiprocessors [33], but without considering the complementary fast DVFS. Although other researchers showed the potential of on-chip regulators [13][5][4], few studies have been published on how to overcome the associated huge hardware cost.

Per-core DVFS Optimization. Li et al. proposed thrifty barrier to improve the energy efficiency of multithread applications [34] and the management heuristic [21]. Isci et al. investigated several global power management policies assuming per-core DVFS facility [35]. Kim et al. investigated the possibility of per-core DVFS using on-chip voltage regulator [20]. Leverich et al. proposed core-level power gating for system-level power reduction [36]. Ma et al. [37] proposed a scalable power control al-

gorithm for per-core DVFS tuning for a mesh-like many-core processors. The design complexity and cost of per-core DVFS for current multicore processors have already been barely affordable. The cost of per-core DVFS can only be prohibitive for the future many-core processors [31]. By contrast, "AgileRegulator" is built on the cost-effective VFI-based solution and is much more scalable than the per-core DVFS design.

Fairness. Unlike the fairness caused by share resource contentions mentioned in previous works [38][10][11][8], we mainly focus on the fairness introduced by DVFS operations, which is a unique contribution of this work.

8 Conclusions

We introduced a novel hybrid architecture "AgileRegulator", by exploring the advantage of both on-chip and off-chip VRs. Two complementary techniques, SAAS and RAAS, were proposed. We showed our scheme performs close to the ideal per-core DVFS in terms of energy efficiency without imposing prohibitive hardware overhead. The area cost in our scheme can well fit into the predicted dark silicon and this servers as a universal way to utilize the chip area for power savings. We found that the synergy of SAAS and RASS solutions can significantly improve the energy efficiency in almost all the cases and we believe the scheme has great potential for the area-energy trade-offs for the future microprocessor design.

References

[1] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *ISCA'11*, pp. 365–376, 2011.

[2] J. Howard, S. Dighe, Y. Hoskote, S. Vangal, D. Finan, G. Ruhl, D. Jenkins, and H. Wilson, "A 48-core ia-32 message-passing processor with dvfs in 45nm cmos," in *ISSCC'10*, pp. 108–109, 2010.

[3] C. Isci, A. Buyuktosunoglu, and M. Martonosi, "Long-term workload phases: duration predictions and applications to dvfs," *Micro, IEEE*, vol. 25, no. 5, pp. 39–51, 2005.

[4] S. Eyerman and L. Eeckhout, "Fine-grained dvfs using on-chip regulators," *ACM TACO*, vol. 8, no. 1, pp. 1–24, 2011.

[5] W. Kim, D. M. Brooks, and G.-Y. Wei, "A fully-integrated 3-level dc/dc converter for nanosecond-scale dvs with fast shunt regulation," in *ISSCC*, pp. 9–10, 2011.

[6] J. Kim and M. Horowitz, "An efficient digital sliding controller for adaptive power-supply regulation," *JSSC*, vol. 37, no. 5, pp. 639–647, 2002.

[7] N. Hardavellas, M. Ferdman, B. Falsafi, and A. Ailamaki, "Toward dark silicon in servers," *IEEE Micro*, vol. Jul./Aug., pp. 6–15, 2011.

[8] S. Kim, D. Chandra, and Y. Solihin, "Fair cache sharing and partitioning in a chip multiprocessor architecture," in *PACT'04*, pp. 111–122, 2004.

[9] J. Chang and G. S. Sohi, "Cooperative caching for chip multiprocessors," in *ISCA'06*, pp. 264 – 276, 2006.

[10] O. Mutlu and T. Moscibroda, "Parallelism-aware batch scheduling: Enhancing both performance and fairness of shared dram systems," in *ISCA'08*, 2008.

[11] E. Ebrahimi, C. J. Lee, O. Mutlu, and Y. N. Patt, "Fairness via source throttling: a configurable and high-performance fairness substrate for multi-core memory systems," in *ASPLOS'10*, pp. 335–346, 2010.

[12] S. Zhuravlev, S. Blagodurov, and A. Fedorova, "Addressing shared resource contention in multicore processors via scheduling," in *ASPLOS'10*, pp. 129–142, 2010.

[13] R. J. Milliken, J. Silva-Martinez, and E. Sanchez-Sinencio, "Full on-chip cmos low-dropout voltage regulator," *Trans. on Circuits and Systems-I*, vol. 54, no. 9, pp. 1879–1890, 2007.

[14] Intel, "Voltage regulator module (vrm) and enterprise voltage regulator-down (evrd) 11.1," tech. rep., Mar. 2009.

[15] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Chapman & Hall, 1 ed., 1984.

[16] A. Bhattacharjee and M. Martonosi, "Thread criticality predictors for dynamic performance, power, and resource management in chip multiprocessors," in *ISCA'09*, 2009.

[17] D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," in *ISCA'00*, vol. 28, pp. 83–94, 2000.

[18] D. Burger and T. Austin, "The simplescalar tool set, version 2.0," tech. rep., Computer Sciences Department, University of Wisconsin-Madison, 1997.

[19] D. Xu, C. Wu, and P.-C. Yew, "On mitigating memory bandwidth contention through bandwidth-aware scheduling," in *PACT'10*, pp. 237–248, 2010.

[20] 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 *HPCA'08*, pp. 123–134, 2008.

[21] J. Li and J. F. Martínez, "Dynamic power-performance adaptation of parallel computation on chip multiprocessors," in *HPCA'06*, pp. 77–87, 2006.

[22] S. Rusu, S. Tam, H. Muljono, J. Stinson, D. Ayers, J. Chang, R. Varada, M. Ratta, and S. Kottapalli, "A 45nm 8-core enterprise xeon processor," in *ISSCC'09*, 2009.

[23] F. Xie, M. Martonosi, and S. Malik, "Bounds on power savings using runtime dynamic voltage scaling: an exact algorithm and a linear-time heuristic approximation," in *ISLPED'05*, pp. 287–292, 2005.

[24] G. Dhiman and T. S. Rosing, "Dynamic voltage frequency scaling for multi-tasking systems using online learning," in *ISLPED'07*, 2007.

[25] T. Sherwood, E. Perelman, G. Hamerly, and B. Calder, "Automatically characterizing large scale program behavior," in *ASPLOS'02*, pp. 45–57, 2002.

[26] J. Donald and M. Martonosi, "Techniques for multicore thermal management: Classification and new exploration," in *ISCA'06*, pp. 78–88, 2006.

[27] K. Meng, R. Joseph, R. P. Dick, and L. Shang, "Multi-optimization power management for chip multiprocessors," in *PACT'08*, pp. 177–186, 2008.

[28] T. Constantinou, Y. Sazeides, P. Michaud, D. Fetis, and A. Seznec, "Performance implications of single thread migration on a chip multi-core," *ACM SIGARCH Computer Architecture News*, vol. 33, no. 4, pp. 80–91, 2005.

[29] G. Yan, X. Liang, Y. Han, and X. Li, "Leveraging the core-level complementary effects of pvt variations to reduce timing emergencies in multi-core processors," in *ISCA'10*, pp. 485–296, 2010.

[30] P. Michaud, A. Seznec, D. Fetis, Y. Sazeides, and T. Constantinou, "A study of thread migration in temperature-constrained multicores," *TACO*, vol. 4, no. 2, pp. 1–28, 2007.

[31] K. K. Rangan, G. Y. Wei, and D.Brooks, "Thread motion: Fine-grained power management for multi-core systems," in *ISCA'09*, pp. 302–313, 2009.

[32] C. Isci, G. Contreras, and M. Martonosi, "Live, runtime phase monitoring and prediction on real systems with application to dynamic power management," in *Micro'06*, 2006.

[33] E. Rotem, R. Ginosar, A. Mendelson, and U. Weiser, "Multiple clock and voltage domains for chip multi processors," in *Micro'09*, pp. 459–468, 2009.

[34] J. Li, J. F.Martínez, and M. C. Huang, "The thrifty barrier: Energy-aware synchronization in shared-memory multiprocessors," in *HPCA'05*, 2005.

[35] C. Isci, A. Buyuktosunoglu, C.-Y. Cher, P. Bose, and M. Martonosi, "An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget," in *Micro'06*, pp. 347–358, 2006.

[36] J. Leverich, M. Monchiero, V. Talwar, P. Ranganathan, and C. Kozyrakis, "Power management of datacenter workloads using per-core power gating," *IEEE Computer Architecture Letters*, vol. 8, no. 2, pp. 48–51, 2009.

[37] K. Ma, X. Li, M. Chen, and X. Wang, "Scalable power control for many-core architectures running multi-threaded applications," in *ISCA'11*, 2011.

[38] M. Kondo, H. Sasaki, and H. Nakamura, "Improving fairness, throughput and energy-efficiency on a chip multiprocessor through dvfs," *SIGARCH Comput. Archit. News*, vol. 35, no. 1, pp. 31–38, 2007.