J Supercomput (2014) 68:414–442
DOI 10.1007/s11227-013-1044-2

Managing power constraints in a single-core scenario through power tokens

Juan M. Cebrián · Daniel Sánchez · Juan L. Aragón · Stefanos Kaxiras

Published online: 20 November 2013

Abstract Current microprocessors face constant thermal and power-related problems during their everyday use, usually solved by applying a power budget to the processor/core. Dynamic voltage and frequency scaling (DVFS) has been an effective technique that allowed microprocessors to match a predefined power budget. However, the continuous increase of leakage power due to technology scaling along with low resolution of DVFS makes it less attractive as a technique to match a predefined power budget as technology goes to deep-submicron. In this paper, we propose the use of microarchitectural techniques to accurately match a power constraint while maximizing the energy-efficiency of the processor. We will predict the processor power dissipation at cycle level (*power token throttling*) or at a basic block level (*basic block level mechanism*), using the dissipated power translated into tokens to select between different power-saving microarchitectural techniques. We also introduce a two-level approach in which DVFS acts as a coarse-grain technique to lower the average power dissipation towards the power budget, while microarchitectural techniques focus on removing the numerous power spikes. Experimental results show that the use of power-saving microarchitectural techniques in conjunction with DVFS is up to six

J. M. Cebrián (✉) · D. Sánchez · J. L. Aragón
University of Murcia, Murcia, Spain
e-mail: jcebrian@ditec.um.es

D. Sánchez
e-mail: dsanchez@ditec.um.es

J. L. Aragón
e-mail: jlaragon@ditec.um.es

S. Kaxiras
University of Uppsala, Uppsala, Sweden
e-mail: stefanos.kaxiras@it.uu.se

times more precise, in terms of total energy consumed over the power budget, than only using DVFS to match a predefined power budget.

Keywords Hardware · Power management · Power budget · DVFS · Energy efficiency · Power estimation

1 Introduction

In the past few years, computer architects have faced an important design change. Individual core performance is saturating and processor designs are moving to multi-core approaches looking for throughput instead of individual core performance. In order to reduce packaging costs, processors are generally designed for average power dissipation and thermal constraints (it is improbable that a processor uses all of its resources at once [9]) and face special cases with both power saving and thermal management mechanisms.

Dynamic thermal management (DTM) is a mechanism that reduces the processor power dissipation (and performance) during critical time intervals so it can cool down. One way to achieve this goal is to set a power budget to the processor. This processor's power budget is not only useful to control the processor power and temperature (and reduce packing and cooling costs) but also to adapt to external (off-chip) power constraints (i.e., shared power input). It also reduces the need for large decoupling on-chip capacitors and the need of power delivery circuitry to be over-designed.

There are situations in which device power requirements are much lower than the power needs of the processor at full speed. In most of the cases, we cannot afford to design a new processor to meet whatever power constraint because it is too expensive. Therefore, being able to set a power budget to the processor could help designers to reuse existent hardware in devices that are heavily constrained. The problem grows when, as usual, power constraints are temporal and after some period of low performance to save power we want all the processor's performance back. For example, imagine we have a computation cluster connected to one or more UPS units to protect it from power failures. If there is a power cut, all processors will continue working at full speed consuming all of the UPS batteries quickly. After some time it will switch off the computers when the batteries are close to run out, losing all the work on fly. During the power failure, it might be more interesting to extend the UPS battery duration at the expense of degrading some performance. On the embedded market, being able to set a power budget depending on the battery level could be useful, as most of the people willing to trade some performance when they are running out of battery.

Another usage for a power budget is the case of a computing center that shares a power supply among all kind of electric devices (i.e., computers, lights, air conditioning, etc.). In a worst case scenario (e.g., in summer at mid-day with all the computers working at full speed), if we integrate some kind of power budget management into the processors, we could set a power budget to lower the ambient temperature also having more power for the air conditioning. In this way, we could design the power capacity of the computing center for the average case, reducing its cost.

In addition, the increasing number of on-chip components is leading to the "Dark Silicon" effect, as the resulting power density does not permit them all to be in simulta-

neous use. This *utilization wall* [9] creates a design space of chips which are collections of heterogeneous components, each carefully tuned to execute particular tasks efficiently. These heterogeneous processors will probably have at least one complex core to deal with single-threaded or latency-oriented applications, and there is where we want to focus our work.

One way to make the processor's power go under a power budget is dynamic voltage and frequency scaling (DVFS) [15,21,22,25]. DVFS relies on modifications in both voltage and frequency to reduce the processor's dynamic power, as dynamic power depends on both voltage (quadratically) and frequency (linearly). DVFS has been implemented in many commercial processors and is commonly used by DTM techniques [8]. The major advantage of DVFS is the low-performance degradation associated with the voltage and frequency reduction. However, DVFS has important drawbacks:

- Long transition times between power modes (partially solved in [12]).
- Long exploration and use windows (in order to amortize DVFS overheads), making it difficult to adapt precisely to the program behavior.
- When activated, DVFS affects all instructions regardless of their usefulness in the program. Therefore, it cannot exploit situations such as instruction slack, instruction criticality, or low confidence on the predicted path.
- As process technology shrinks, reducing voltage (V_{DD}) while maintaining the same switching speed ($\delta \approx 1/(V_{DD} - V_T)^{\alpha}$, with $\alpha > 1$) will force a reduction on V_T. However, this reduction is impractical for low technology scales since leakage power exponentially depends on V_T. Alternatively we can use dynamic frequency scaling (DFS), but it is not as energy-efficient as DVFS, since it seriously hurts performance.

DVFS is a reactive mechanism that works at a coarse-grain level, being useful to control temperature, as it varies slowly over time. However, this kind of mechanism does not provide any peak power guarantees (only allows to control peak temperature), hence it is still susceptible to a power virus [17]. Moreover, it cannot be used for the power supply design or the reliability qualification of the processor because, unlike temperature, power demand can go to its maximum value in any given cycle [5].

This paper studies the use of fine-grain microarchitectural power-saving techniques to deal with power variations when matching a power budget. The studied mechanisms will capture and store information about the processor power dissipation, either at a cycle level (power-token throttling, PTT) or at a basic block level (basic block level manager, BBLM). We can decide whether the next instruction/basic block can be executed by checking both the past energy consumption and how far the processor is from the power budget. In this paper, we also propose an efficient two-level approach that firstly applies DVFS as a coarse-grain approach to lower the average power and, secondly, uses microarchitectural techniques to remove the remaining numerous power spikes. One of the benefits of using microarchitectural techniques is that they can be applied at a fine-grain level (tens of cycles), only when the processor exceeds the power budget. The peculiarities of each microarchitectural technique will be detailed in Sect. 2 but, in general terms, these fine-grain techniques:

1. Locate non-critical instructions in cycles over the power budget and delay them to cycles under the budget. Note that non-critical instructions can become critical if they are delayed for a long time.
2. Previous studies have shown that 30 % of the overall processor power comes from instructions from the wrong path of execution [2,16]. Therefore, we can reduce the number of low-confident speculative instructions in the pipeline when the processor is exceeding the power budget by using a confidence estimator.
3. The pipeline can be throttled at different stages when the two previous policies are not enough to put the processor under the required power budget.

In [7], we presented a preliminary version of both mechanisms: PTT and BBLM. This paper extends our previous work by thoroughly analyzing our ideas and including a more detailed evaluation. In addition, this work also proposes and evaluates the use of two novel mechanisms that help microarchitectural techniques to match the predefined power budget by analyzing the power curve trend: *preventive switch-off* and *switch-on*. *Preventive switch-off* stores differences in power between cycles and predicts if the next cycle will be over the power budget; whereas *preventive switch-on* does the opposite: calculates differences to disable techniques before getting under the power budget, hoping that the decreasing power trend will be enough to lead the processor under the budget. Experimental results show that the use of power-saving microarchitectural techniques is more energy efficient as well as more accurate than DVFS for driving the processor under the required power budget.

The main contributions of the current manuscript are:

– Timeline analysis of power dissipation is performed in Sect. 2.4.
– Section 3.2.5 introduces the two novel mechanisms for managing energy consumption: preventive switch-off and switch-on.
– Section 4.2 illustrates the importance of accuracy when matching a power budget.
– Section 4.4 is extended with more information about the distribution of cycles and area over the power budget for the evaluated benchmarks.
– Analysis on the accuracy and energy-efficiency of the new proposed mechanisms (preventive switch-off/switch-on) is done in Sect. 4.6.
– Finally, Sect. 4.7 performs a temperature analysis of the base case vs the proposed two-level mechanism when matching a predefined power budget.

The rest of the paper is organized as follows. Section 2 provides some background on microarchitectural power-saving techniques. Section 3 shows a first analysis of the individual techniques and motivates the need for a hybrid approach to match the power budget. Section 4 describes our simulation methodology and reports the main experimental results. Finally, Sect. 5 shows our concluding remarks.

2 Background and related work

2.1 Pipeline throttling

Pipeline throttling (PT) is a technique that reduces the amount of in-flight instructions in the pipeline to reduce power dissipation [2,4,16]. PT can be applied at different

stages, producing different effects on both power and performance. In [7], we presented a detailed analysis of the capabilities of PT. This analysis shows that, under low-restrictive power constraints (70–80 %), PT mechanisms are able to match the target power budget with minimal performance degradation, saving power from instructions that belong to a mispredicted path. However, under high-power restrictions, the performance degradation of PT mechanisms makes them worse than DVFS. Sartori et al. [19] proposes a technique to minimize power disparity between pipeline stages, this technique added to our own could further improve the energy efficiency of pipeline throttling.

2.2 Critical path-based instruction reordering

Data dependencies are one of the main bottlenecks in high-performance processors: dependency chains limit the performance of the machine leaving most of the processor structures and logic idle [23]. These chains of dependent instructions are known as the critical path of the code. A processor's performance is determined by how fast it can execute the critical path, not by how fast it can execute all of the code. If we were able to distinguish between instructions that belong to the critical path of the program, we could either accelerate their execution to improve the machine's performance, or do a more power-efficient execution of non-critical instructions to reduce power and temperature. Authors in [23] proposed a critical path predictor that is able to predict critical instructions. However, the amount of cycles a non-critical instruction can be delayed without becoming critical is really small [6]. In [7], we presented a detailed analysis of the capabilities of critical path-based instruction reordering. Results showed that reordering is possible under low-restrictive power budgets. However, for restrictive power budgets, chances to find cycles under the power budget to place delayed instructions becomes a key limiting factor for this mechanism.

2.3 Hybrid approaches

There are several proposals that try to merge both DVFS and microarchitectural techniques in a two-level mechanism to benefit from both coarse and fine-grain mechanisms. Sasanka et al. [20] propose the use of DVS and some microarchitectural techniques to specifically reduce the energy consumption in real time video applications. Their selected microarchitectural techniques try to reduce the power of functional units and the instruction window. Moreover, they do not use clock gating in their proposal, and the studied benchmarks have special properties that their selected microarchitectural techniques can take advantage of. Winter and Albonesi [24] propose the use of a two-level approach that merges DVFS and thread migration to reduce temperature in SMT processors. Other approaches focus on controlling peak power consumption by dynamically configuring the processor's component size [13] or by proposing different DVFS schemes to optimize throughput [18]. Homayoun et al. [10] proposes an alternative way to provide resource sharing among cores that increases energy efficiency. Bacha and Teodorescu [3] pushes DVFS to the next level by relying on hardware cor-

rection mechanisms to reduce voltage and frequency below safe limits. This approach can be very interesting in time-critical power constraint scenarios.

2.4 Timeline analysis of power dissipation

In the previous sections, we have introduced some of the most common dynamic power saving mechanisms. Our primary goal in this paper is to accurately match a predefined power budget while minimizing the energy penalty. A priori, it is not known whether the power constraint is close to the peak power dissipation or not. Therefore, we need to estimate how much these techniques are able to reduce the power towards the budget and how they can be combined to obtain further reductions. As in previous sections we will use an aggressive power budget of 50 % of the peak power dissipation as a challenging scenario for the power saving mechanisms.

This section shows a timeline analysis of the studied techniques for a given power budget of 50 % (Fig. 1) (we only show three benchmarks for the sake of visibility).

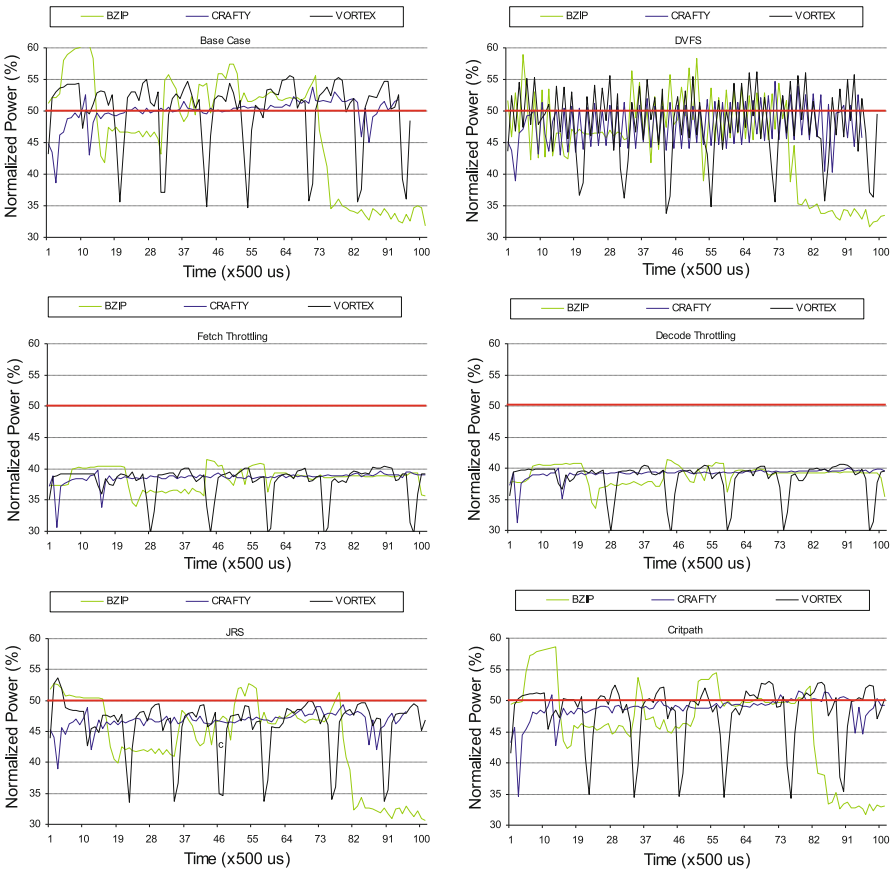


Fig. 1 Per-cycle power dissipation for a power budget of 50 %

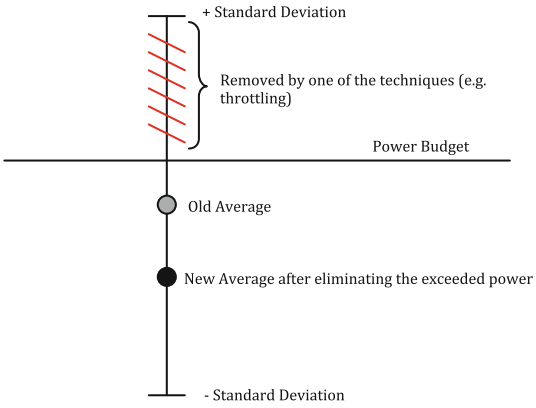


Fig. 2 Standard deviation effect for one window (500 μs)

The power dissipation of each technique is represented relative to the maximum power dissipation, so 50 means 50 % of the peak power of the processor using clock gating. As not all benchmarks last the same, we only show the first 100 windows of 500-μs each. The *critpath* configuration shown is for a delay of 8 cycles. All fetch, decode and JRS configurations shown are enabled when the processor is over the power budget (horizontal line in the figure) and produce a full stop of the front-end. In addition, we show a non-blocking DVFS mechanism (can continue execution during mode changes) that can work in two modes as in [8]: the first one produces a 15 % power reduction with 5 % performance degradation and the second mode reduces power by 35 % and performance by 10 %.

With the information provided by Fig. 1, we can classify the different mechanisms depending on how close they are to the power budget. Critical path is the power-saving mechanism that reduces power the least of all the studied mechanisms, followed by JRS. These two mechanisms are the ones with the least power reduction but also the ones with minimal performance impact on the processor. Meanwhile, DVFS constantly changes between power modes trying to get as close as possible to the power budget, showing low accuracy because of the long exploration and activation windows. Note that both Fetch and Decode throttling (the most restrictive approaches) also have a predefined power budget of 50 % but are stuck at 40 %. This is not a simulation mistake. Note that the graphs plot the average power dissipation on intervals of 500 μs, that is, for our processor configuration (3 GHz), 1500000 instructions. Both mechanisms are applied at cycle level, and only during cycles that are over the PB. That makes the standard deviation from the average power quite high in these windows. When we reduce the power dissipation in the cycles that are over the power budget, the new average for the next window is also reduced. This puts the new average power further below the requested power budget. Figure 2 illustrates this effect.

3 Power-saving microarchitectural techniques

When reducing power dissipation under a power budget there is something that should be kept in mind: there is going to be performance degradation. This is justified by the

fact that there is a strong need to match the imposed power constraint. Our main goal is to reduce power dissipation to match an imposed power budget in an energy-efficient way. To achieve that goal, we first need to study how the different microarchitectural techniques behave independently, in order to design an adaptive mechanism that takes advantage of each of their peculiarities. As evaluation metrics we will measure both the fraction of cycles exceeding the power budget over the total execution cycles as well as the induced performance degradation for the whole SPECint2000 (see Sect. 4.1 for details about the processor configuration).

3.1 Reactive techniques

Reactive techniques check the processor's power at a cycle-level. If the current power exceeds the required budget, the processor applies a certain microarchitectural technique to reduce its power dissipation. The major concern about reactive techniques is that they must be applied for some time in order to achieve its low-power effect. If in a given cycle the processor goes over the power budget, we do not know a priori how long this situation will last. Furthermore, reactive techniques are not able to remove all the cycles the processor spends over the power budget since they are activated once the processor power dissipation is over the budget, unless we use a predictive approach. Examples of dynamic power reduction reactive techniques are: DVFS, pipeline throttling and instruction reordering introduced in Sect. 2.

3.2 Predictive techniques

As cited previously, reactive techniques are not good enough to accurately match a predefined power budget, basically because they rely on local information about power dissipation, and have no knowledge about past or future trends in power dissipation. Predictive techniques, on the other hand, will capture and store information about the processor power dissipation, either at a cycle level (PTT) or at a basic block level (BBLM), in order to decide whether the next instruction/basic block can be executed based on its previous power. For predictive techniques to work, we need a way to measure power dissipation at a cycle level.

3.2.1 Power tokens

Up until recently (Intel Sandy Bridge processors [1] MSRs), there was no way to accurately measure power dissipation in real time. Power measurements are just approximations based on performance counters done over periods of thousands of cycles. When using microarchitectural techniques that work at a very fine granularity (from several to a few tens of cycles) we need some way to estimate power at this fine granularity. That is why we propose the power-token approach, which calculates the power dissipated by an instruction at commit stage by adding the base power dissipation of the instruction (i.e., all regular accesses to structures done by that instruction) plus a dynamic component that depends on the time it spends on the pipeline. The dynamic

component corresponds to the combined RUU[1] wakeup-matching logic power that we divide between all instructions in the RUU. Therefore, in order to work with power-token units in a simple way, we define a power-token unit as the joules dissipated by one instruction staying in the RUU for one cycle.

We calculated the base power-tokens of every instruction type using the SPECint2000 benchmark suite. Once we had the base power for all the possible instructions, we used a K-mean algorithm to group instructions with similar base power dissipation. We analyzed independent instructions and several grouping, that ranged from 32 to 4 groups. Simulated results showed that having just 8 groups of instructions is accurate enough for the power-token approach to properly work with a deviation lower than 1 % (compared to accounting for the actual energy consumption as provided by the evaluated simulator). In this way, the number of power-tokens dissipated by an instruction will be calculated as the addition of its base power-tokens plus the amount of cycles it spends in the RUU. The implementation of the power-token approach is done by means of an 8K-entry history table[2] (power-token history table, PTHT), accessed by PC, which stores the power cost (in power-token units) of each instruction's previous execution. This table introduces a measured power overhead of around 0.5 % which is accounted for in our results. We also need five 16-bit adders for accounting the power tokens in our modeled processor (four wide processor plus one for adding the total power) and a register to store the current power dissipated in the processor (0.025 % power overhead). When an instruction commits, the PTHT is updated with the number of power-tokens dissipated.

We can perform a per-instruction-cycle power estimation dividing the power dissipated in the previous execution of the instruction (stored at the PTHT table) by the cycles it stayed in the pipeline. This estimation is not perfectly accurate, as the instruction power dissipation is not equal across the cycles. However, for a per-cycle power estimation, the fact that we are working with 4–8 width out of order cores minimizes this prediction error, as instructions in different stages share the pipeline. Finally, the overall processor power dissipation (in power-token units) can be easily estimated in a given cycle based on the instructions that are traversing the pipeline without using performance counters by simply accumulating the power-tokens (as provided by the PTHT) of each instruction being fetched. Our evaluated cores implement a three-stage fetch unit so there is enough time to estimate this power. We believe power tokens are a good approximation to power and energy information that could be available in future microprocessors.

3.2.2 Power-token throttling (PTT)

This technique follows a simple premise: if there is power left to burn, let the instruction enter the pipeline, otherwise, stall fetch. Basically, it estimates the dynamic power of the instructions inside the pipeline in a given cycle by means of accounting for the power-tokens they consume. When the total power of all the instructions inside the

[1] Register Update Unit.

[2] 8 bits per entry.

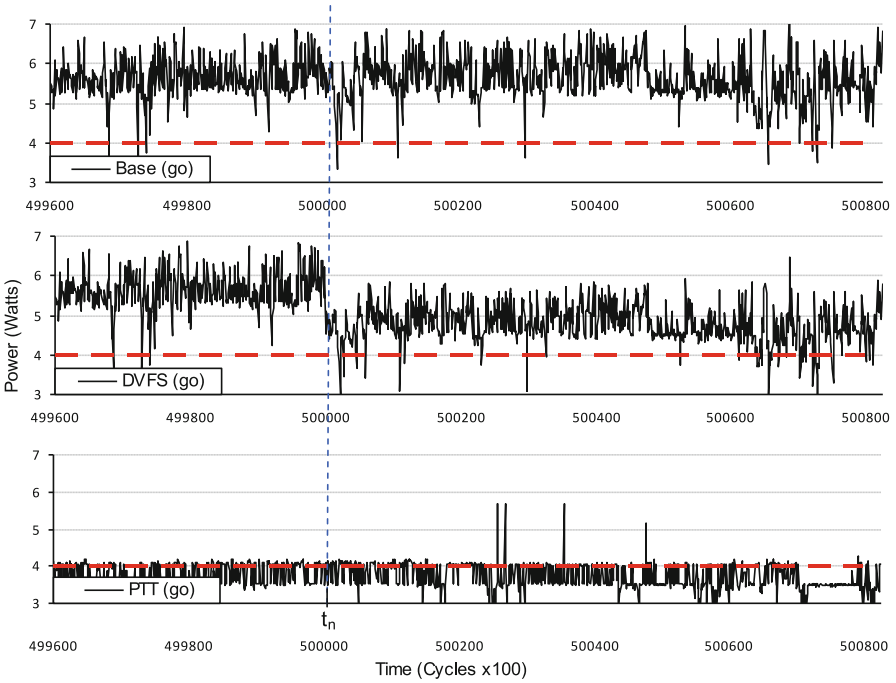


Fig. 3 Base versus DVFS exploration window power dissipation for the "go" benchmark

processor exceeds the power budget, the fetch stage is stalled until instructions commit and leave the pipeline, releasing their power-tokens. PTT allows the introduction of new instructions into the pipeline when the current total power goes under the power budget. This is the most accurate technique but at the same time the least energy-efficient since it incurs in serious performance loss. We also evaluate a modified version of the PTT approach that uses a critical path predictor (labeled as PTT-CP in figures in Sect. 4) to detect critical instructions. In this way, even if we are over the power budget, we allow these instructions to continue their execution. This version is less aggressive than the original PTT.

First of all, in order to gain some insight on the power dissipation behavior of DVFS, Fig. 3 shows the average power dissipation for the "go" benchmark of the base processor compared to the same processor using DVFS. The dashed horizontal line represents the required power budget (set to 4 W, which corresponds to a 50 % power budget over the peak power of 8 W, see Sect. 4.3 for simulation and configuration details). The used DVFS implementation (as in [11]) calculates the average power dissipation over an exploration window of 500K cycles, and modifies the voltage and frequency accordingly (from a set of pairs voltage/frequency modes) to match the desired power budget. As we can see in Fig. 3, for the exploration window W_{n-1} DVFS calculates its average power dissipation and changes to the power mode that is closest to the power budget. This power mode will be used in the next exploration window, W_n. In this example, during W_n the average power dissipation of the base

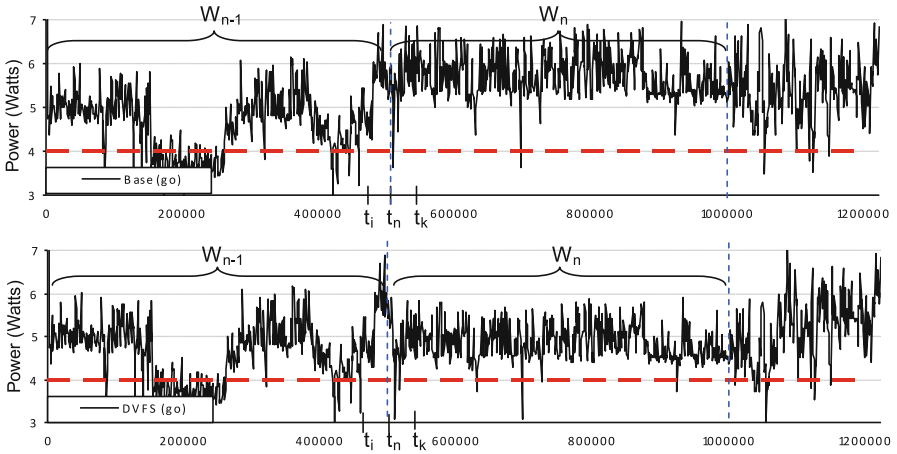


Fig. 4 Detailed per-cycle power dissipation for the "go" benchmark

case increases (Fig. 3, top), which leaves the average power dissipation of the DVFS processor still over the power budget (even though a lower power mode was selected; Fig. 3, bottom). This entails that the amount of power over the budget is indeed lower when using DVFS but its accuracy for matching the power budget is questionable.

Figure 4 is a more detailed representation of the power behavior for the "go" benchmark (this figure shows a zoom-in from time t_i to t_k in Fig. 3). For the sake of visibility, we only plot now the power information for a reduced time interval of 120K cycles. As said before, for the DVFS case (Fig. 4, middle), after t_n the selected power mode is the one closest to the average power dissipation of the previous window W_{n-1} (as seen in Fig. 3, bottom). However, because during window W_{n-1} the average power was higher than the actual power after point t_n, the DVFS selected power mode is still not able to drive the processor under the power budget.

On the other hand, the PTT approach (Fig. 4, bottom), accurately follows the power budget at a cycle level, as it knows in advance how much power (in tokens) the next instruction consumes. If we cannot afford to execute it, we wait until a committed instruction leaves the pipeline and there is enough power left to burn in the new instruction. Spikes in the PTT plot are due to branches that are left to enter the pipeline in order to discover eventual mispredictions as soon as possible (otherwise, performance is seriously hurt). However, the performance penalty of PTT is quite high, as we will see later in this paper.

3.2.3 Basic block level manager (BBLM)

This second technique keeps track of historical information about power-tokens consumed by a basic block[3] as well as the power left to exceed the power budget in order to decide what technique should be applied to reduce power dissipation. The available techniques are (from less to more aggressive): (a) none, (b) critical path, (c)

[3] Assumed in this paper as a stream of instructions between two branches.

confidence estimation throttling, or (d) decode-commit ratio throttling. Every time the processor decodes a branch, it estimates the energy consumed by the previous basic block, measured as the addition of the power-tokens of every instruction that belongs to that basic block. This power measure is stored in the branch predictor entry of the branch that points to the start of that basic block (not the one that we have just decoded). The overhead of this mechanism consists of 9 additional bits per branch predictor entry plus a register and a 16-bit adder to store the current basic block power which corresponds to a negligible 0.3 % power increase in the total processor power, accounted for in our results. Therefore, every time a branch is predicted we also obtain information about the subsequent basic block power estimation, length, etc., and we estimate how far from the power budget we will get if we completely execute that basic block. Depending on how far the estimated power is from the budget, we select a different power saving technique. In addition, when updating the branch predictor's saturating counter, we also update the power-tokens consumed by the subsequent basic block (that corresponds to the next predicted path). Please note that, although it is very difficult that a basic block will always consume the same exact amount of energy, this is not really relevant for our mechanisms, as long as the selected technique is the same for a specific basic block (which is actually the common case). We define two threshold values, X and Y for BBLM to decide which technique should be applied. Therefore, for a fraction of power over the budget lower than X, BBLM will select the first technique (i.e., critical path, which is the least aggressive); from X to Y, BBLM will select the second technique (i.e., confidence estimation throttling); and for a power over the budget greater than Y, BBLM will select the third technique (i.e., decode-commit ratio throttling, which is the most aggressive). Similarly, techniques are disabled progressively, in reverse order, once we get *under* the power budget. We performed an experimental study to determine the best values for these thresholds (best power budget matching with minimal performance degradation), and discovered that for the current processor configuration and used benchmarks the best thresholds are $X = 15$ and $Y = 65$.

3.2.4 Two-level approach

Cycle-level microarchitectural techniques have some disadvantages compared to a circuit-level mechanism like DVFS. When using DVFS only a third part of the power reduction comes from performance degradation ($P \approx V_{DD}^2 * f$) whereas for microarchitectural techniques every change has a direct impact in performance. All instructions from the correct path of execution consume a constant amount of power, so only removing instructions from wrong path of execution or reordering instructions will allow the processor to go towards a power budget with minimal performance impact.

When the current power dissipation is far from the budget, microarchitectural techniques do not work efficiently, as we will see in Sect. 4, since there are few chances to reduce power dissipation without performance degradation. On the other hand, DVFS is extremely inaccurate when there are power spikes, because the influence of power spikes on the average power dissipation from the sampling window is negligible. If we only use microarchitectural techniques we may not get close to the power budget, or the performance degradation will be high. Moreover, it has been proved that bench-

marks exhibit different program phases with different average power dissipation. If we are able to detect these phases we could use only microarchitectural techniques in phases close to the power budget and a coarse-grain approach (DVFS) in phases far from the power budget. Therefore, our proposal consists of a two-level approach: first we apply DVFS to make coarse-grain decisions about power dissipation, and secondly we apply microarchitectural techniques for fine-grain decisions (mainly for removing the numerous power spikes). This two-level approach increases DVFS accuracy for matching a power budget while at the same time does not require all the power modes to reach a power budget. By using this two-level approach only the least aggressive DVFS power modes are enough to accurately match the predefined power budget. It is important to note that when the process technology goes far below 65 nm the reduction on V_T will be limited by both reliability and leakage power. At this point, it may not be a matter of not using the extreme power saving modes, but implementing those power modes will be infeasible.

3.2.5 Preventive switch-off and switch-on

Reactive techniques rely on current information to decide whether or not to apply some microarchitectural techniques to reduce power dissipation. The base decision mechanism looks at the current cycle power dissipation to detect if the processor is over the power budget, and then applies whatever mechanism the processor implements to reduce dynamic power. One possible way to improve this could be adding some intelligence to the process. If the power is continuously increasing from cycle to cycle, we may predict that the next cycle will continue rising. If that is true, we can enable the power saving mechanisms preventing the processor from exceeding the power budget. The preventive switch-off approach calculates and stores the differences between per-cycle power dissipation and predicts if power will be over the budget the next cycle by adding the current cycle power plus the difference in power between the last two cycles (Fig. 5 points A and B).

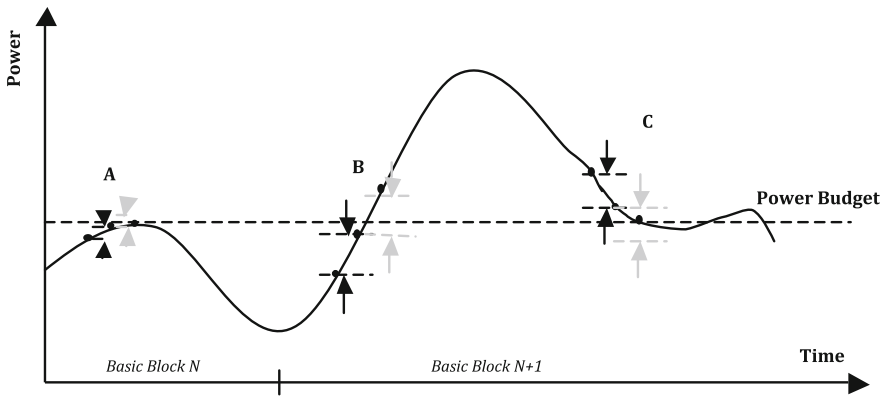


Fig. 5 Overview of the preventive switch-off and switch-on mechanisms

Moreover, this mechanism is also useful when we work with predictive techniques (as PTT and BBLM). In BBLM, we have an idea of how far from the power budget the current basic block is, but have no idea on how power is distributed among cycles. With preventive switch-off we can add some additional information about per-cycle power trends and help the selected microarchitectural mechanism in its task. However, preventive switch-off may lead to false positives if we prematurely enable microarchitectural techniques. For example, in Fig. 5A, the current cycle power plus the difference in power between the last two cycles will lead the next cycle over the power budget and thus we should enable power-saving techniques to prevent that. In this specific case this decision is wrong, as power will start to descend in the next few cycles. This may not happen with predictive techniques because they have additional information about future power behavior of the basic block. If BBLM was enabled in Fig. 5A, the mechanism will choose not to do anything prematurely as the average power of the basic block is under the power budget.

Analogously, we have the preventive switch-on approach. When microarchitectural mechanisms are enabled power begins to descend until the processor gets under the power budget, and then power-saving mechanisms are disabled. Usually, microarchitectural mechanisms have some hysteresis in which power continues going down. If we disable the power-saving technique in advance, before getting under the power budget, the hysteresis period should be enough to lead the processor under the power budget. The preventive switch-on mechanism works like preventive switch-off but when power is decreasing. The mechanism calculates the power differences between cycles and estimates if the current power minus the delta power is under the budget in order to disable microarchitectural techniques (as seen in Fig. 5C). Note that false positives in this mechanism are less harmful than for preventive switch-off, because only "accuracy" is reduced for matching the power budget, but not performance.

4 Experimental results

4.1 Simulation methodology

To evaluate the energy-efficiency of both DVFS and microarchitectural techniques, we have used the SPECint2000 benchmark suite. All benchmarks were compiled with maximum optimizations (-O4-fast) by the Compaq Alpha compiler and they were run using a modified version of Simplescalar/Wattch simulator using power models provided by McPAT [14] that includes the dynamic and static power models for the evaluated microarchitectural approaches as well as their associated power overhead. All benchmarks were run to completion using the reduced input data set (test). Table 1 shows the configuration of the simulated processor.

4.2 Importance of accuracy

This paper is focused on adapting the processor's power dissipation to a target power constraint, with minimal deviation from the target power budget while minimizing energy increase and performance slowdown.

Table 1 Core configuration

Processor core	
Process technology	32 nm
Frequency	3000 MHz
V_{DD}	1.2 V
Instruction window	128 RUU + 64 IW
Load store queue	64 entries
Decode width	4 inst/cycle
Issue width	4 inst/cycle
Functional units	6 Int Alu; 2 Int Mult
	4 FP Alu; 2 FP Mult
Branch predictor	16 bit Gshare
Memory hierarchy	
Memory latency	300
L1 I-cache	64 KB, 2-way, 1 cycle lat.
L1 D-cache	64 KB, 2-way, 1 cycle lat.
L2 cache	1 MB/core, 4-way, unified
	12 cycle latency
TLB	256 entries

If we want to ensure minimal deviation from the target power budget we need to introduce a metric that enables us to measure how close we are from the power constraint. We define the metric Area over the Power Budget (AoPB) as the amount of energy (Joules) between the power budget and each core dynamic power curve, represented by shadowed areas in Fig. 6. The lower the area (energy) the more accurate the analyzed technique is (note that the ideal AoPB is zero). We can compare two scenarios, one where the processor is going over the PB by 100 % for 0.1 of the time and another where it is going over PB by 10 % for 1.0 of the time. These two will give the same area over PB value, but the first case could be much more harmful. In order to ensure that our power saving mechanisms do not behave like the first case we will also provide two additional metrics, the total number of cycles that the processor spends over the power budget and, for our final results (Sect. 4.5), a population chart where we show the number of cycles and the areas in different power intervals, that go from 50 % of the power budget to our processor peak power consumption.

Lets us provide an example that tries to illustrate the importance of the accuracy on matching a predefined power budget. Imagine that we want to increase the number of cores in a CMP while maintaining the same TDP.[4] (this is a common practice done by microprocessor manufacturers in the past years). For a 16-core CMP with a 100 W TDP, each core would use 6.25 W (for simplicity let us ignore the interconnection network). If we set a power budget of 50 % we could ideally duplicate the number of cores in that CMP with the same TDP (up to 32 cores, each one consuming an average

[4] Thermal design package.

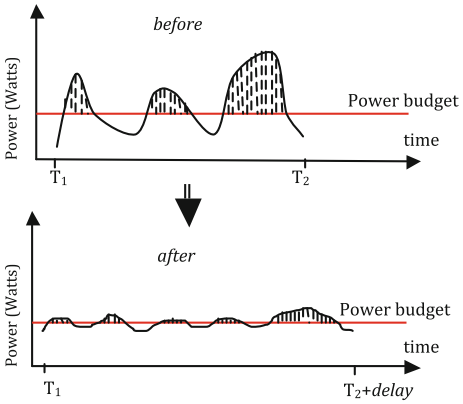


Fig. 6 Example of the Area over Power Budget (AoPB) metric. *Shadowed areas* represent the energy consumed over the target power budget

of 3.125 W). But for this ideal case a perfect accuracy on matching the power budget is needed.

As we will see later on this paper, standard DVFS incurs in a energy deviation of 75 % over the power budget (the AoPB metric). Therefore, with a 75 % deviation each core dissipation raises to $3.125 * 1.75 = 5.46$ W, and, for a 100 W TDP, we can put a maximum of $100/5.46 = 18$ cores inside the CMP. Using a two-level approach the deviation is reduced below 10 %, that gives us a potential average power dissipation of $3.125 * 1.1 = 3.4375$ W per core, so we can put $100/3.4375 = 29$ cores inside our CMP. Therefore, thanks to the extra cores (we could go from 16 cores in the original CMP design to 29 cores within the same 100 W TDP) we can perfectly overcome the performance degradation that results from using our proposed two-level mechanism.

4.3 A power budget of what? (100 % Usage $\neq$ 100 % Power)

When we think about the maximum power dissipation of a processor we think about the processor using all of its resources at once (i.e., the peak dissipation of a processor), and that barely happens in common applications or even scientific applications. Our base processor has an a priori peak power dissipation of 9.6 W as provided by the McPAT power simulator. When a circuit-level technique such as clock gating is enabled, the average power dissipation for the SPECint2000 drops to 4 W with a dynamic peak power dissipation of 8 W for our simulated processor and benchmark suite. We will use 8 W of peak power[5] as our reference power (i.e., 100 % power budget), which corresponds to the power dissipation of the processor before applying any power-saving technique.

Next sections show the simulation results for the SPECint2000 benchmark suite for: (a) fraction of cycles over the power budget (Cycles over PB); (b) total power exceeded over the budget (AoPB); and (c) normalized energy. We use the AoPB

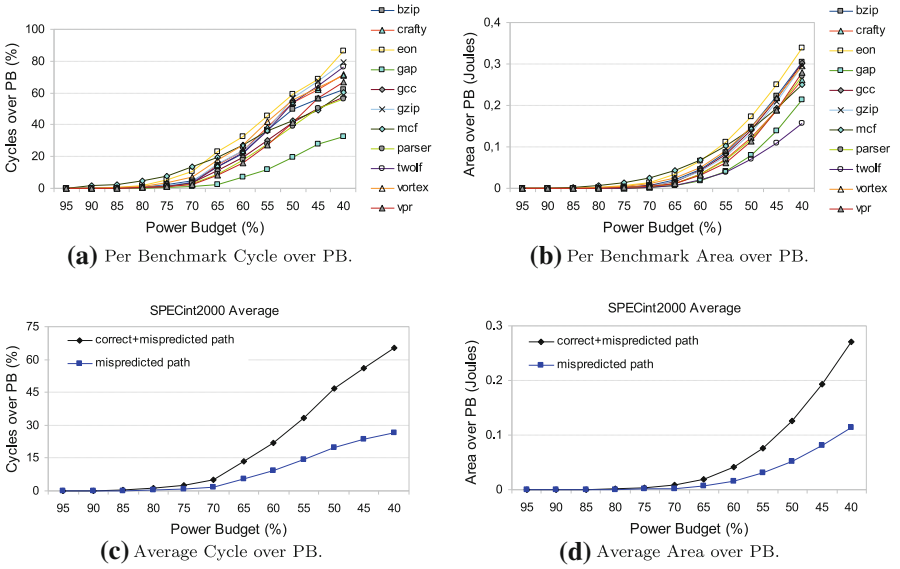
[5] Using the Wattch's clock-gating style cc3, which scales power linearly with unit usage. Inactive units still dissipate 10 % of its maximum power.

metric instead of the average power since the standard deviation of the per-cycle power dissipation is quite high. The processor has periods of high power dissipation hidden by periods of low power dissipation (branch mispredictions, cache misses, etc.), therefore, the average power is not a good metric for what is really happening inside the processor.

4.4 Cycles over PB and area distribution

Figure 7 shows both the fraction of cycles over power budget and the area over the power budget for different budgets and benchmarks before applying any power saving technique. Evaluated PBs range from 95 % of the original peak power to 40 %.

As we can see in Fig. 7 top (a, b), both the amount of cycles and the area the processor spends over the power budget is almost negligible for high budgets (95–70 %), due to the effects of clock gating on power dissipation, but being highly noticeable for power budgets under 65 %. As explained before, DVFS will be unable to find power spikes for low power budgets, as it works with the average power dissipation of its long exploration windows. On the other hand, microarchitectural techniques will detect these spikes and remove them whenever possible. Figure 7 bottom (c, d) shows the cycles and area over the power budget distinguishing between power/cycles from correct path of execution as well as mispredicted paths. We can see how almost half of the power/cycles over the power budget come from instructions from the wrong path of execution, those are the ones we are most interested in, as microarchitectural techniques heavily rely on them to reduce power towards the budget with low performance degradation.



(a) Per Benchmark Cycle over PB. **(b)** Per Benchmark Area over PB.

(c) Average Cycle over PB. **(d)** Average Area over PB.

Fig. 7 Area and cycles over PB distributions

4.5 Efficiency on matching a power budget

As cited before, the main objective of this paper is the proposal of accurate energy-efficient techniques for matching a power budget, and thus this subsection evaluates the energy-efficiency of the proposed microarchitectural techniques as well as their accuracy for matching an imposed power budget. We report results for the following techniques: DVFS, PTT, BBLM (with and without using critical path information), and the two-level approach labeled as (DVFS + BBLM). Our DVFS approach initially calculates the average power dissipated during an exploration window. If the mechanism finds out that the average power dissipation is over the budget, DVFS changes to a pair of voltage and frequency values from a set of predefined working modes in order to reduce the average power dissipation. DVFS uses a exploration window of 500K cycles with a very fast transition time between modes set to 50 mV/ns (as in [12]), so it takes only 4 cycles to switch from one mode to another.[6] This implementation is similar to the one proposed in [11]. The evaluated working modes are the following for each technique:

- DVFS: Uses five power modes (100 % V_{DD}, 100 % f), (95 % V_{DD}, 95 % f), (90 % V_{DD}, 90 % f), (90 % V_{DD}, 75 % f), and (90 % V_{DD}, 65 % f).
- DVFS+BBLM: The power modes are reduced to (100 % V_{DD}, 100 % f), (95 % V_{DD}, 95 % f), and (90 % V_{DD}, 90 % f).
- DFS: Only scales frequency as needed (V_{DD} remains unchanged).

The studied BBLM uses the configuration parameters and techniques proposed in Sect. 3.2.3. As explained before, the selected thresholds are $X = 15$ and $Y = 65$. Critical path (CP) is used as the first technique, JRS-throttling as the second technique, and DCR-throttling as the third one.

Figure 8 shows the normalized energy consumption for the different techniques and power budgets. As we can see, the two-level approach (DVFS + BBLM) is the most energy-efficient technique for all the studied power budgets, and especially for very restrictive power budgets. The rest of microarchitectural techniques (except PTT) as well as DFS show a low energy degradation (between 4 and 10 %). BBLM shows similar energy-efficiency than DFS up to a power budget of 50 % while reducing four times more the AoPB. This energy reduction comes from the additional power reduction from microarchitectural techniques when matching the power budget. In addition, as we are working under an imposed power budget, Figs. 9 and 10 show how accurate each technique is when trying to match the power constraint, as we plot both the relative cycles and area over the power budget. It can be observed that all microarchitectural techniques are far more accurate than DVFS when adapting to the imposed power budget. This is because the coarse-grain nature of DVFS that makes it unable to detect and remove the exceeding power over the budget. For power budgets between 90 and 70 %, BBLM+PTT (CP) is the best approach: 25 % of area over PB is left after applying this technique with a total energy increase of only 6 % (Fig. 8).

[6] For McPAT V_{DD} at 32 nm is 1.2 V, so each 5 % reduction in voltage translates into 60 mV. That means it will take 1.2 ns to switch between modes. As the processor runs at 3 GHz, it will take 3.6 (4) cycles to change between power modes.

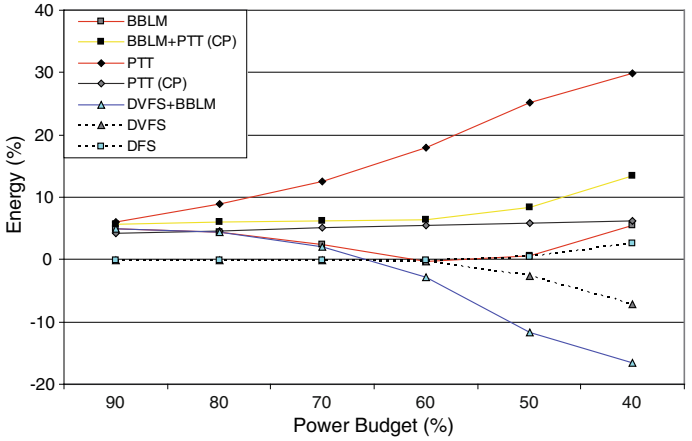


Fig. 8 Normalized energy consumption

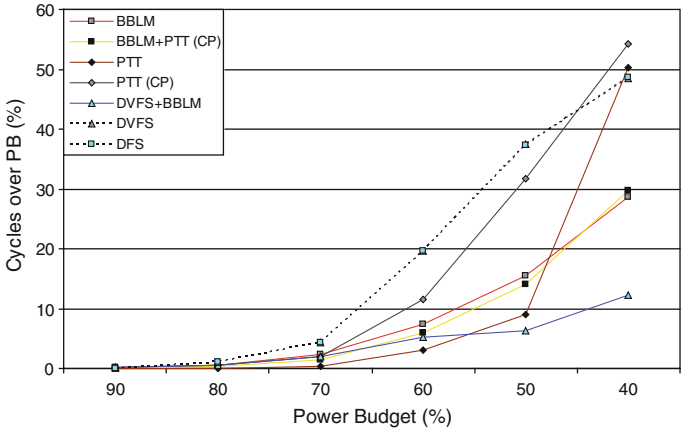


Fig. 9 Relative cycles over PB for different power budgets

PTT is more accurate but also shows an increasing energy trend. This is because it throttles the pipeline very frequently, making the overall energy increase to offset the savings provided by the throttling mechanisms.

When we move to more restrictive power budgets (<60 %), our two-level (DVFS + BBLM) approach is 4 % more energy-efficient than DVFS while the accuracy of the former is six times better (11 % of relative AoPB is left by the two-level approach whereas DVFS barely reduces the area to a 90 % of the original). For an extreme power budget of 40 %, our two-level approach gets even better: only 10 % of relative AoPB is left, in contrast with the 60 % reported by DVFS. In addition, the two-level approach for this extreme power budget is 11 % more energy-efficient than DVFS and 20 % than DFS. In general terms, DVFS and DFS are coarse-grain approaches unable to remove the numerous power spikes making them far less accurate than microarchitectural techniques.

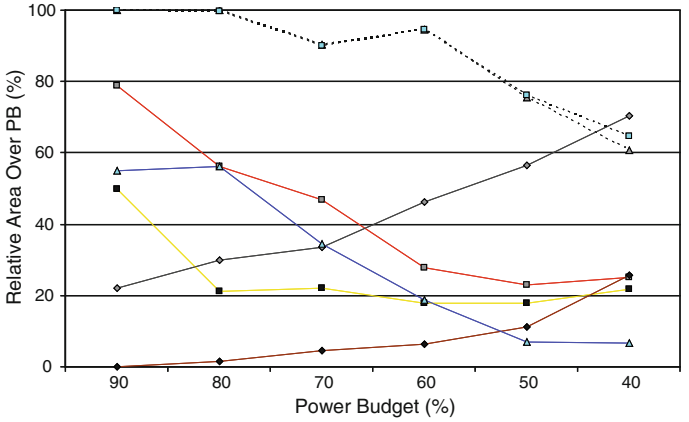


Fig. 10 Relative area over PB for different power budgets

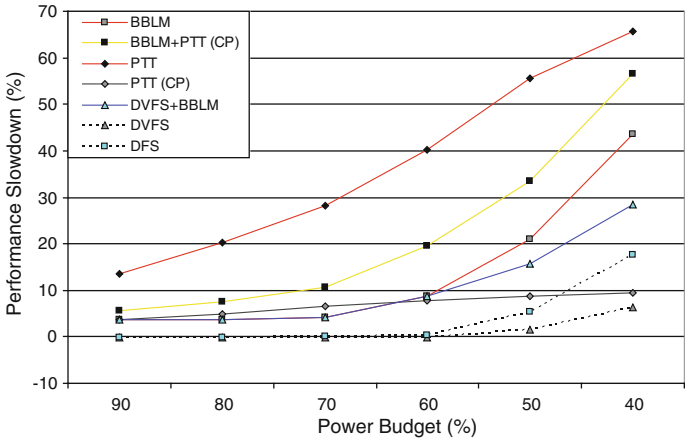


Fig. 11 Normalized performance slowdown

Figure 11 shows the performance degradation of the different approaches. We can clearly see that PTT is the less performance efficient because it completely stalls the pipeline when the processor gets over the power budget. With BBLM+PTT being less aggressive than PTT and BBLM even less that the two previous approaches (that is also reflected in the energy consumption, Fig. 8). Finally, our proposed two-level mechanism (DVFS + BBLM) follows the performance degradation curve of DFS, being always 10 % slower than DFS (or 21 % slower than DVFS). However, as mentioned in Sect. 4.2, this slowdown can be overcome by the addition of new cores in a CMP scenario, using the saved power from setting the power budget.

Finally, Figs. 12 and 13 show the AoPB and CoPB distributions (population chart) of different power intervals when we request to match a power budget of 50 %. We can clearly see how our hybrid mechanism is able to almost completely eliminate the exceeding energy (area) and cycles over the power budget, ensuring that almost no

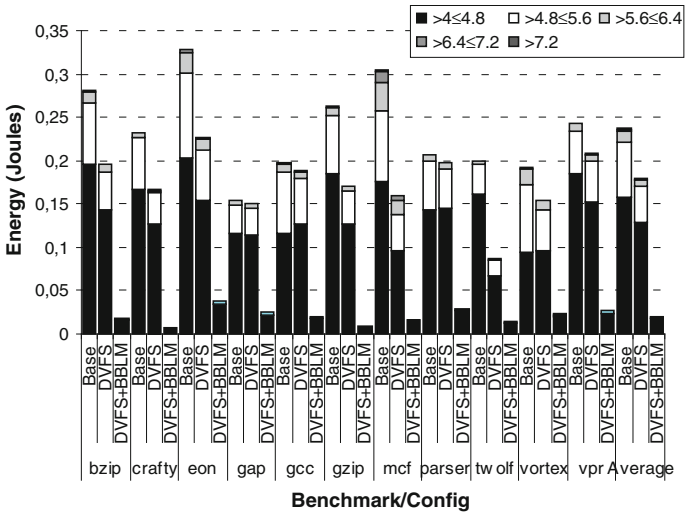


Fig. 12 AoPB distribution over the power budget

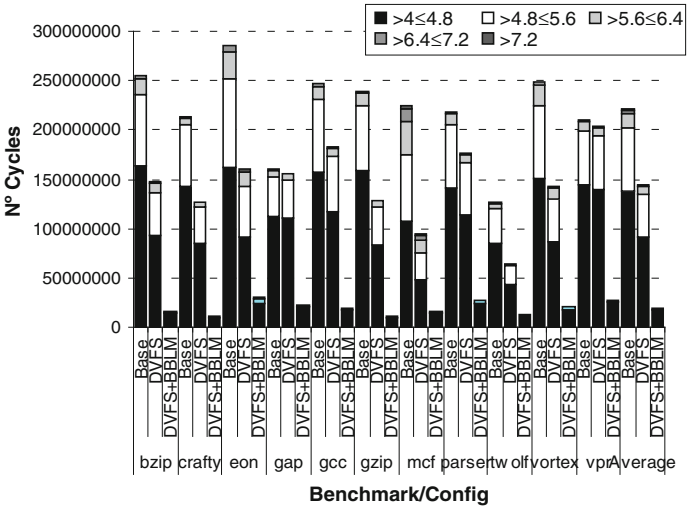


Fig. 13 Cycle distribution over the power budget

cycle exceeds 5.6 W when a 4 W power budget is set. Meanwhile DVFS, due to its coarse grain nature, is unable to properly adapt to the power variations, and still shows some cycles with power peaks close to 7.2 W.

4.6 Efficiency of preventive switch-off and switch-on

In this section, we show some experimental results for both the preventive switch-off and switch-on mechanisms when simultaneously applied with both BBLM and the proposed two-level approach (DVFS + BBLM).

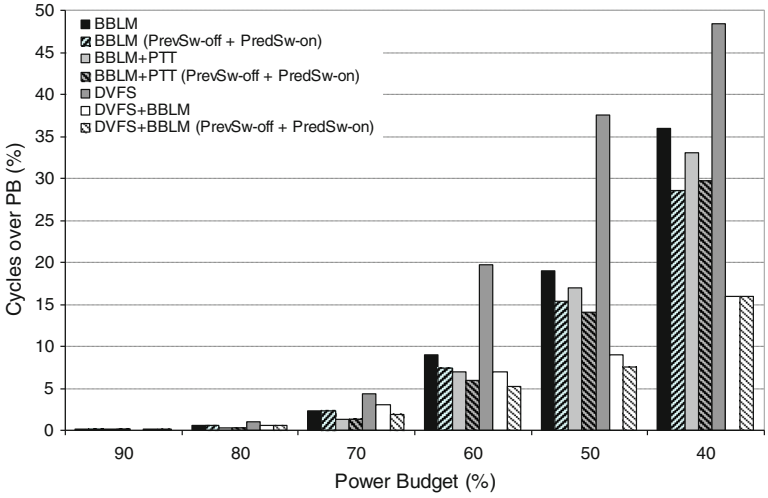


Fig. 14 Relative cycles over PB for different power budgets

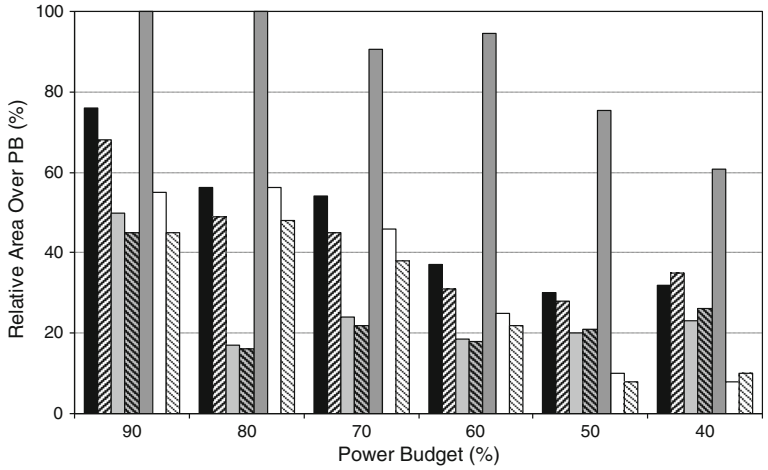


Fig. 15 Relative area over PB for different power budgets

Figures 14 and 15 show a comparison between the original techniques and the ones that include the preventive switch-off and switch-on mechanisms in terms of cycles and area over the power budget. For power budgets between 90 and 60 % there is a reduction in both area and cycles over the power budget of 3–4 % for all the studied mechanisms. For power budgets of 50 and 40 % there is something curious happening in the processor. Although there are less cycles over the power budget when preventive switch-off is applied, the total AoPB increases. This is due to the fact that preventive switch-off and switch-on approaches give more false positives with restrictive power budgets. In addition, as most of the cycles are over the power budget, preventive switch-off has few chances to discover rising power trends.

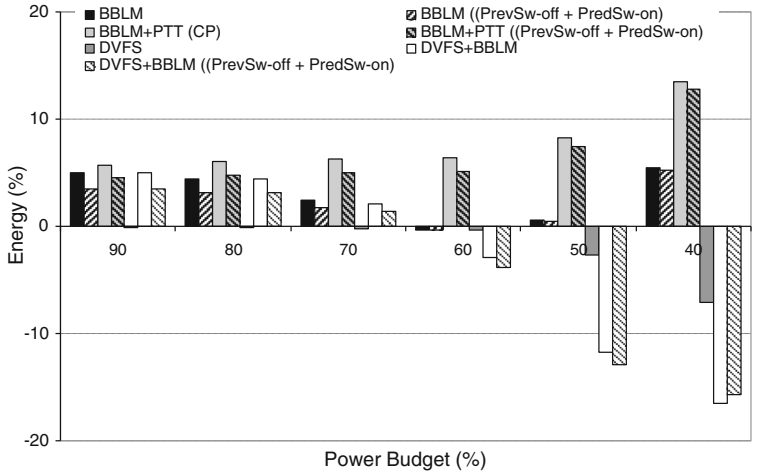


Fig. 16 Normalized energy for all the evaluated approaches along with preventive switch-off and switch-on

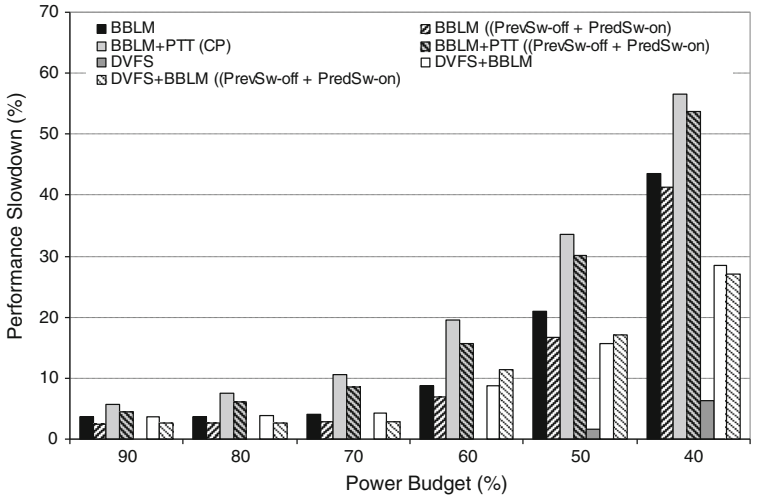


Fig. 17 Normalized performance slowdown for all the evaluated approaches along with preventive switch-off and switch-on

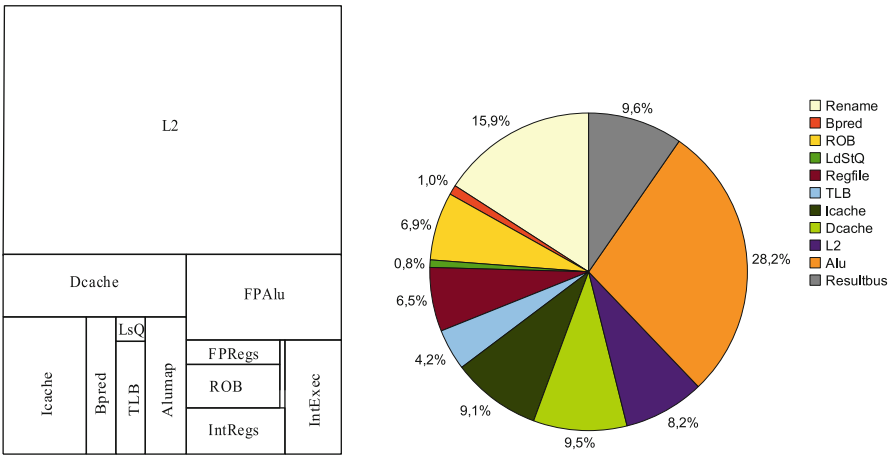
When preventive switch-off and switch-on work together with BBLM and the two-level approach, energy numbers get better (1–2 %) as shown in Fig. 16, especially for power budgets between 90 and 60 %. For a power budget of 50 %, the difference is minimal, and for 40 % techniques are less energy efficient, again, due to false positives. Summarizing, preventive switch-off and switch-on techniques usually help with both accuracy and energy for power budgets between 90 and 50 %, but become less effective with restrictive power budgets (Fig. 17).

4.7 Temperature analysis

Thermal hotspots increase cooling costs and have a negative impact on reliability and performance. The significant increase in cooling costs requires designs for temperature margins lower than the worst-case. Moreover, leakage power is exponentially dependent on temperature and an incremental feedback loop exists between temperature and leakage, which may turn small structures into hotspots and potentially damage the circuit. High temperatures also adversely affect performance, as the effective operating speed of transistors decreases as they heat up. In this section we will analyze the per-structure and per-benchmark temperature for the base case, and our proposed two-level mechanism.

In this section we analyze the thermal profiles for all the benchmark suites used in this paper running on a single core. In addition, we will also show power usage of the different structures normalized to the total power usage of each benchmark, as well as a comparison with the processor peak power dissipation as measured by McPAT.

We simulate a 14-stage, out-of-order core. Figure 18a shows the floorplan configuration used to obtain temperature and power numbers in this section. To build a N-core CMP, the core floorplan can be replicated N times. Power and area numbers for this core configuration were obtained through the McPAT framework. Figure 18b depicts the power distribution of the different core structures provided by McPAT and normalized to the total (peak) power of the core (9.6 W). We then input these power numbers into Simplescalar/Watch to obtain preliminary average power numbers of each structure. Using these average power numbers we build the input files for HotSpot—Hotfloorplanner that will generate the core floorplan. In addition to the power numbers, HotSpot also needs per-structure area information (provided by



(a) Core Floorplan **(b)** Power breakdown (McPAT)

Fig. 18 Core configuration

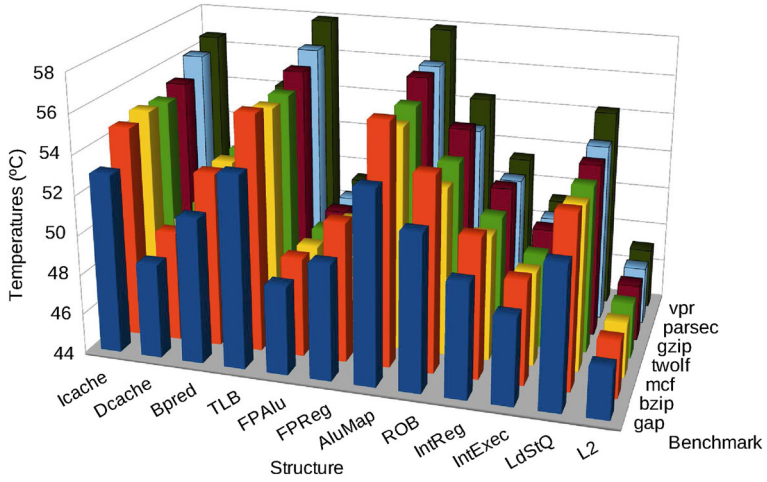


Fig. 19 Thermal profiles for the SPECint2000 benchmark suite

McPAT) and structure communication needs (we used Alpha 21264 structure dependencies as our input for Hotfloorplanner). To estimate temperatures we integrated HotSpot, that uses as inputs the generated floorplan and the dynamic and leakage power provided by McPAT, into Simplescalar/Wattch. We also implemented a temperature/leakage power loop inside the Simplescalar/Wattch simulator, as there is a direct relationship between both terms.

Figure 19 shows the per-structure temperatures for the SPECint2000 suite. We can clearly identify the major core hotspots: TLB (Translation Lookaside Buffer), Icache (Instruction Cache), Alumap (Rename logic), ALUs (Arithmetic Logic Unit), ROB (Reorder buffer) and LdStQ (Load Store Queue). Power utilization (and thus temperature) has a high variability between benchmarks. For example, we can see temperature variations of 6 °C between vpr (58 °C) and gap (52 °C).

Figure 20 shows the per-structure and per-benchmark temperatures of the analyzed core using our proposed two-level mechanism with a power budget of 50 % of the original peak power consumption using clock gating. We can see a temperature reduction of 6–7 °C for the hottest structures of the floorplan (AluMap, TLB and Icache). This inter-structure temperature reduction reduces the temperature gradient of the chip, making it more reliable.

If we want to gain some insight of how each benchmark uses the core resources we need to take a look to the power numbers. Table 2 shows the per-structure runtime power usage for the benchmark suite. Structure power is normalized to the total power dissipated by each benchmark. According to these numbers, most of the power dissipated by the core is done by the Icache, ROB, Rename Logic, ALUs and TLB, that match the temperature hotspots of the chip seen in Fig. 19. There is also another hotspot, the TLB, that does not consume a big part of the total power. The TLB has a really small area (as it can be seen in Fig. 18, left), so it is difficult for this structure to transfer heat to the heatsink.

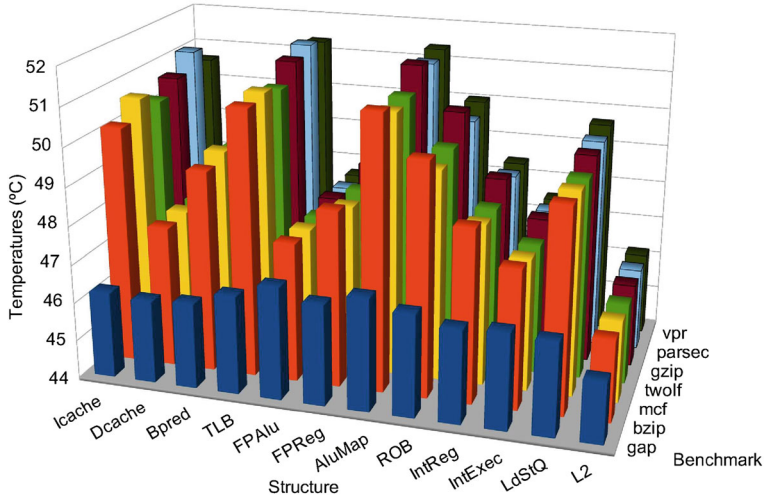


Fig. 20 Thermal profiles for the SPECint2000 benchmark suite using a two-level mechanism with a PB of 50 %

Table 2 Normalized power distribution for SPECint2000

Structure	bzip (%)	gap (%)	gzip (%)	mcf (%)	parsec (%)	twolf (%)	vpr (%)
Rename	20.2	19	20.4	19.1	19.6	19.7	20.3
Bpred	0.7	0.6	0.7	0.7	0.7	0.7	0.6
ROB	13.4	12.3	13.3	11.1	11.2	12.1	12
LdStQ	0.4	0.6	0.4	0.5	0.4	0.5	0.6
Regfile	6.3	6	6.1	5.6	5.8	6	6.1
TLB	13.2	12.9	13.3	13.9	14.4	13.4	13.9
Icache	28.1	29	29	31.2	32	29.8	29.6
Dcache	5.1	5.4	4.5	5.4	4.4	5.5	6.5
L2	1.4	1.8	1.4	1.5	1.4	1.5	1.2
Alu	10.5	11.7	10.2	10.3	9.5	10.1	8.4
Resultbus	0.2	0.3	0.2	0.2	0.2	0.2	0.2

5 Conclusions and future work

Current general purpose microprocessors can be used in several kinds of gadgets that usually have different power requirements. Some scenarios being able to define a maximum power budget for the processor can help the reuse of previous designs in new devices. This can be especially useful if available options are reduced to either switch-off the device or to reduce the power dissipation to match the power constraint. Note that, in many cases, the shut-off option is not even viable (e.g., for critical systems). We cannot design the thermal envelopment of a processor for the worst case scenario, because production price highly increases and the processor barely ever reaches its

peak temperature. However, we can set a power budget to the processor, limiting its power and temperature.

When microarchitectural techniques are used we obtain a more precise power budget matching while maximizing the processor's energy efficiency. We propose two adaptive techniques: PTT and BBLM. PTT is a token-based approach that keeps track of the current power being dissipated by the processor, measured as tokens, and stalls fetch if the next instruction to be executed will make the processor go over the power budget. This is a very aggressive mechanism that obtains an accurate power budget matching but has a huge performance degradation. On the other hand, BBLM uses basic block energy consumption history (translated into power-tokens) in order to determine the best power saving technique for the current and near-future processor power dissipation. BBLM optimizes the use of microarchitectural techniques to minimize performance impact while removing most of the power spikes. However, these mechanisms by themselves are not enough to match restrictive power budgets because of their high performance degradation.

We have then proposed a two-level approach that combines both microarchitectural techniques and DVFS to take advantage of their best qualities. DVFS acts as a coarse-grain technique to lower the average power dissipation while microarchitectural techniques remove all the power spikes efficiently. The two-level approach (BBLM + DVFS) is able to outperform DVFS in both energy efficiency (up to 11 % more energy efficient) and area exceeded over the power budget (up to 6 times less area). As a side effect of this accurate power budget matching, our mechanism provides another interesting benefit: a more stable temperature over execution time. For the studied benchmarks we can obtain a 12 % peak temperature reduction. They are also able to balance temperature between structures, reducing the peak temperature of the hottest component making it closer to the temperature of the coldest structure (temperature gradient). This temperature reduction not only reduces leakage power but also increases reliability and can result in reduced packing costs.

Current trends in microprocessor design include heterogeneous architectures. These architectures are highly dominated by the "Dark Silicon" effect, that is, the amount of transistors that can be enabled simultaneously is limited. As part of our future work we want to cover such architectures, starting by extending our work to regular CMPs and then including heterogeneous cores, hardware components and accelerators.

Acknowledgments This work was supported by the Spanish MEC, MICINN and EU Commission FEDER funds under Grants CSD2006-00046 and TIN2009-14475-C04. Also by the EU-FP7 ICT Project "Embedded Reconfigurable Architecture (ERA)", contract No. 249059. Finally, the EU-FP7 HiPEAC funded an internship of J.M. Cebrián at U. Uppsala.

References

1. Intel 64 and IA-32 Architectures Software Developer's Manual, Volume 3C. Intel Corporation
2. Aragon JL, Gonzalez J, Gonzalez A (2003) Power-aware control speculation through selective throttling. In: Proceedings of the 9th international symposium on high-performance computer, architecture, pp 103–112. doi:10.1109/HPCA.2003.1183528

3. Bacha A, Teodorescu R (2013) Dynamic reduction of voltage margins by leveraging on-chip ecc in itanium ii processors. In: Proceedings of the 40th annual international symposium on computer architecture, ISCA '13. ACM, New York, pp 297–307. doi:10.1145/2485922.2485948
4. Baniasadi A, Moshovos A (2001) Instruction flow-based front-end throttling for power-aware high-performance processors. In: Proceedings of the international symposium on low power electronics and design, pp 16–21. doi:10.1109/LPE.2001.945365
5. Brooks DM, Bose P, Schuster SE, Jacobson H, Kudva PN, Buyuktosunoglu A, Wellman JD, Zyuban V, Gupta M, Cook PW (2000) Power-aware microarchitecture: design and modeling challenges for next-generation microprocessors. IEEE Micro 20:26–44. doi:10.1109/40.888701. http://portal.acm.org/citation.cfm?id=623296.624401
6. Casmira J, Grunwald D (2000) Dynamic instruction scheduling slack. In: Proceedings of the KoolChips workshop
7. Cebrian JM, Aragon JL, Garcia JM, Petoumenos P, Kaxiras S (2009) Efficient microarchitecture policies for accurately adapting to power constraints. In: Proceedings of the IEEE international parallel & distributed processing, symposium, pp 1–12. doi:10.1109/IPDPS.2009.5161022
8. Donald J, Martonosi M (2006) Techniques for multicore thermal management: classification and new exploration. In: Proceedings of the 33rd international symposium on computer, architecture, pp 78–88. doi:10.1109/ISCA.2006.39
9. Esmaeilzadeh H, Blem E, St Amant R, Sankaralingam K, Burger D (2011) Dark silicon and the end of multicore scaling. In: Proceedings of the 38th annual international symposium on computer architecture, ISCA '11. ACM, New York, pp 365–376. doi:10.1145/2000064.2000108
10. Homayoun H, Kontorinis V, Shayan A, Lin TW, Tullsen D (2012) Dynamically heterogeneous cores through 3d resource pooling. In: 2012 IEEE 18th international symposium on high performance computer architecture (HPCA), pp 1–12. doi:10.1109/HPCA.2012.6169037
11. Isci C, Buyuktosunoglu A, Cher CY, Bose P, Martonosi M (2006) An analysis of efficient multi-core global power management policies: maximizing performance for a given power budget. In: Proceedings of the 39th annual IEEE/ACM international symposium on microarchitecture, pp 347–358. doi:10.1109/MICRO.2006.8
12. Kim W, Gupta MS, Wei GY, Brooks D (2008) System level analysis of fast, per-core dvfs using on-chip switching regulators. In: Proceedings of the IEEE 14th international symposium on high performance computer, architecture, pp 123–134. doi:10.1109/HPCA.2008.4658633
13. Kontorinis V, Shayan A, Tullsen DM, Kumar R (2009) Reducing peak power with a table-driven adaptive processor core. In: Proceedings of the 42nd annual IEEE/ACM international symposium on microarchitecture, MICRO 42. ACM, New York, pp 189–200. doi:10.1145/1669112.1669137
14. Li S, Ahn JH, Strong RD, Brockman JB, Tullsen DM, Jouppi NP (2009) Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In: Proceedings of the 42th international symposium on microarchitecture, pp 469–480
15. Macken P, Degrauwe M, Van Paemel M, Oguey H (1990) A voltage reduction technique for digital systems. In: Proceedings of the 37th IEEE international solid-state circuits conference. Digest of Technical Papers, pp 238–239. doi:10.1109/ISSCC.1990.110213
16. Manne S, Klauser A, Grunwald D (1998) Pipeline gating: speculation control for energy reduction. In: Proceedings of the 25th, annual international symposium on computer architecture, pp 132–141. doi:10.1109/ISCA.1998.694769
17. Najeeb K, Konda VVR, Hari SKS, Kamakoti V, Vedula VM (2007) Power virus generation using behavioral models of circuits. In: Proceedings of the 25th IEEE VLSI test symmposium, VTS '07. IEEE Computer Society, Washington, pp 35–42. doi:10.1109/VTS.2007.49
18. Sartori J, Kumar R (2009) Three scalable approaches to improving many-core throughput for a given peak power budget. In: HiPC'09, pp 89–98
19. Sartori J, Ahrens B, Kumar R (2012) Power balanced pipelines. In: 2012 IEEE 18th international symposium on high performance computer architecture (HPCA), pp 1–12. doi:10.1109/HPCA.2012.6169032
20. Sasanka R, Hughes CJ, Adve SV (2002) Joint local and global hardware adaptations for energy. In: Proceedings of the 10th international conference on architectural support for programming languages and operating systems, ASPLOS-X. ACM, New York, pp 144–155. doi:10.1145/605397.605413
21. Semeraro G, Magklis G, Balasubramonian R, Albonesi DH, Dwarkadas S, Scott ML (2002) Energy-efficient processor design using multiple clock domains with dynamic voltage and frequency scaling.

In: Proceedings of the 8th international high-performance computer architecture, symposium, pp 29–40. doi:10.1109/HPCA.2002.995696
22. Simunic T, Benini L, Acquaviva A, Glynn P, de Micheli G (2001) Dynamic voltage scaling and power management for portable systems. In: Proceedings on design automation conference, pp 524–529. doi:10.1109/DAC.2001.156195
23. Tune E, Liang D, Tullsen DM, Calder B (2001) Dynamic prediction of critical path instructions. In: Proceedings of the 7th international symposium on high-performance computer, architecture, pp 185–195. doi:10.1109/HPCA.2001.903262
24. Winter JA, Albonesi DH (2008) Addressing thermal nonuniformity in smt workloads. ACM Trans Archit Code Optim 5:4.1-4.28. doi:10.1145/1369396.1369400
25. Wu Q, Juang P, Martonosi M, Clark DW (2005) Voltage and frequency control with adaptive reaction time in multiple-clock-domain processors. In: Proceedings of the 11th international symposium on high-performance computer, architecture, pp 178–189. doi:10.1109/HPCA.2005.43