Counter-Based Power Modeling Methods: Top-Down vs. Bottom-Up

Article *in* The Computer Journal · February 2013

DOI: 10.1093/comjnl/bxs116

CITATIONS
22

READS
189

5 authors, including:

Ramon Bertran
IBM
37 PUBLICATIONS **525** CITATIONS

SEE PROFILE

Counter-based Power Modeling Methods: Top-down vs Bottom-up

Ramon Bertran*[1], Marc Gonzàlez[2], Xavier Martorell[1], Nacho Navarro[2] and Eduard Ayguadé[1]

[1]*Barcelona Supercomputing Center Cr. Jordi Girona 29, 08034 Barcelona, Spain.*
[2]*Department of Computer Architecture, Universitat Politècnica de Catalunya, Cr. Jordi Girona 1-3, 08034 Barcelona, Spain.*
*** Corresponding author: rbertran@ac.upc.edu*
Email: {rbertran,marc,xavim,nacho,eduard}@ac.upc.edu

Counter-based power models have attracted the interest of researchers because they became a quick approach to know the insights of power consumption. Moreover, they allow to overpass the limitations of measurement devices. In this paper, we compare different *Top-down* and *Bottom-up* Counter-based modeling methods. We present a qualitative and quantitative evaluation of their properties. In addition, we study how to extend them to support the currently ubiquitous Dynamic Voltage and Frequency Scaling (DVFS) mechanism. We propose a simple method to generate DVFS agnostic power models from the DVFS specific models. The proposed method is applicable to models generated using any methodology and allows the reduction of the modeling time without affecting the fundamental properties of the models. The study is performed on an 18 DVFS states Intel® Core™ 2 platform using the SPECcpu2006, NAS and LMBENCH benchmark suites. In our testbed, a 6x reduction on the modeling time only increments 1 percentage point on average the error in the predictions.

Keywords: Power estimation, power modeling, performance counters, DVFS

Received 00 Month 201X; revised 00 Month 201X; accepted 00 Month 201X

1. INTRODUCTION

Power density and power consumption are main design constraints in processor designs [1, 2]. Several factors such as the reliability, the lifetime and the operating frequency of the processors are related to power [3, 4]. Moreover, the limitations related to power can also be found at higher level. Data centers composed by thousands of processors require energy sources to sustain the power consumption levels associated to both the actual computation plus the cooling mechanisms to keep the infrastructure below a certain temperature. Large computing infrastructures generate such a power consumption that power and energy have become as critical as performance is. In this context, performance has to be accompanied by efficiency [5, 6].

In accordance to this new situation many techniques have been proposed for controlling the power consumption. Some of them operate at very low level, close to specific hardware functionalities and they are always activated by the hardware itself. An example of them is clock-gating [7]. Others actuate at higher level and they are managed by the runtime according to some defined power policies. For instance, Dynamic Voltage and Frequency Scaling (DVFS) [8] allows to select the operating frequency of each core, thus, it permits to control the overall power consumption. This has been shown to be a powerful technology, and many proposals use it to implement power-aware policies such as power capping or power envelopes [9, 10, 11, 12].

Power-aware policies require some method to obtain the power consumption level [13]. This is usually supported through physical devices that can be placed at the level of a system node, or that can even give more accurate data on a per core basis. Multi-core architectures incorporate power sensors associated to each core [14]. One limitation of these mechanisms is that they are not able to associate particular levels of power consumption to processes in the system. That means that they do not provide an estimation of the power consumption of a particular running process. When dealing with OS power-aware policies, this is a very necessary feature in order to guide the OS scheduling. One promising alternative is to build power models based on hardware events that describe the activity that a process is causing within a processor [15, 16, 17, 18]. This has even been evaluated in shared virtualized environments in order to understand better the energy efficiency challenges [6] and to perform power and energy accounting [19, 20].

Modeling methods based on performance monitoring counters (PMCs) have been shown to be a good solution to estimate power consumption. Their relevance has been confirmed in several areas such as power management [21] or application profiling [22]. PMC-based models are used on-line to guide power aware policies [15, 23] or to evaluate novel research ideas. Their main advantage is that they allow to profile real systems and perform full executions without the need to carry out time-consuming simulations [24, 25, 26, 27, 28]. These examples corroborate that PMC-based power models have been a very solid approach for addressing power issues.

We can distinguish two types of PMC-based modeling methodologies. On the one hand, there are the *Top-down* approaches [15, 17, 24, 26, 29, 30, 31] which aim to define simple, fast and easy to deploy models. *Top-down* models achieve these objectives by being architecture agnostic, avoiding the requirement of specific knowledge of the modeled architecture and by using a reduced set of PMCs. Consequently, even different platforms can be modeled quickly.

On the other hand, the *Bottom-up* approaches [16, 18, 25] rely on some knowledge of the underlying architecture to produce more informative, responsive and accurate power models. These approaches are able to breakdown the power components of the architecture but they are more complex to deploy than the *Top-down* ones.

1.1. Motivation

In this work, we compare various existing modeling methodologies, the *Top-down* and the *Bottom-up*. Specially, we study how they interact with the currently ubiquitous DVFS mechanism. DVFS allows to change the processor voltage and frequency to dynamically adapt the processor performance to a target power and energy requirements. To do so, this flexible hardware mechanism defines a set of performance states (P-states) enabling the runtime to manage the trade-offs between power, energy and performance.

Solutions that use the DVFS mechanism require a model for all the possible DVFS configurations in order to be able to implement the desired policy. The process of modeling all the states of a given platform was not an issue until recently. Nowadays, complexity has increased considerably due to the growth of the number of cores and the number of DVFS states.

To illustrate this problem, we show in Table 1 some realistic examples about the number of models that are required depending on the number of DVFS states and cores. The $DVFS_{chip}$ column is generated using Equation 1 that defines the number of models needed when the DVFS mechanism is implemented globally, in which all cores in the chip share the same DVFS state, like in the Intel® Core™ 2 processor [32]. In contrast, the $DVFS_{core}$ column, generated using the Equation 2, shows the number of models needed if per core DVFS is implemented, in which each core can run at a different DVFS state[3], like in the POWER7 processor [33].

TABLE 1. Number of models needed to fully model a platform given the number of cores and DVFS states and the DVFS type.

#Cores	#DVFS states	#Models required	
		$DVFS_{chip}$	$DVFS_{core}$
2	3	6	9
2	18	36	189
8	3	24	164
8	18	144	1562274
16	3	48	968
16	18	288	2203961429

$$\#Models = \#Cores \times \#States_{dvfs} \quad (1)$$

$$\#Models = \sum_{i=1}^{i=\#Cores} \left(\frac{(\#States_{dvfs} + (i-1))!}{i! \times (\#States_{dvfs} - 1)!} \right) \quad (2)$$

As we can see, adding extra cores and DVFS states increases notably the number of required models. For instance, for a 8 core machine with 18 DVFS states, we need to generate from 144 to up to 1562274 different models depending on how DVFS is implemented[4]. Obviously, this is an impractical task, even when applying systematic methods. Not only that, this problem is going to increase on upcoming architectures with more cores and DVFS states. For instance, the POWER7 already provides an almost continuous DVFS mechanism with 1% variations in frequency between DVFS states [14].

In conclusion, the time required to fully model a platform hardens the research on power aware solutions based on PMC models. Thus, there is a need to extend the models to support the different DVFS settings without increasing excessively the modeling time and maintaining the fundamental model properties. Moreover with the advent of heterogeneous systems [34] it will be useful to not increase the modeling time in order to enable the application of PMC-based solutions on all the components present in such systems.

The modeling time problem can also affect the real products. For instance, the POWER7 already includes *power proxies* (PMC based power models) to estimate the power consumption of each core of the processor [35]. These firmware encoded power models should be calibrated at production time before commercialize the product. Not only that, due to aging effects these power

[3] Both formulas are derived from directly applying permutation and combination theory. (1) we need to generate $\#States_{dvfs}$ models for 1 core active, for 2 cores active, ..., $\#Cores$ active. (2) we need to generate for 1 to $\#Cores$, all possible combinations of $\#Cores$ length of possible $States_{dvfs}$, without taking into account the order.

[4] Sometimes DVFS is implemented per groups of cores. Consequently, the numbers shown represent the minimum and maximum number of possible combinations.

models eventually would require to be re-calibrated. In these cases, the time required to perform the process of generating/re-calibrating the power models is also crucial.

1.2. Contributions

The contributions of this paper are fourfold:

1. We provide, for the first time, a comprehensive description of the implications of using *top-down* and *bottom-up* modeling methodologies. To the best of our knowledge, this work is the first comparing –qualitatively and quantitatively– models generated using both types of modeling methods.
2. We propose a methodology to derive the DVFS agnostic models from DVFS specific ones, keeping their fundamental properties. The benefits of having a single model for all the DVFS states are notorious. A single model simplifies the existing solutions and allows the detection of operational DVFS sweet-points. Our method ensures that the model properties such as accuracy and responsiveness remain similar to the ones of the DVFS specific models.
3. We study, for the first time, the trade-offs between the model properties and the modeling time. We perform a design space exploration showing that reducing the number of training DVFS states reduces considerably the modeling time and keeps the model properties.
4. The quantitative evaluation is done by applying our method to extend models generated using three *Top-down* and one *Bottom-up* modeling methods. The direct comparison shows that the *Bottom-up* model consistently outperforms *Top-down* approaches, even when it is trained using only 3 DVFS states. This confirms our intuitions about the *Bottom-up* models: they have a higher first time deployment cost but it is worth to pay the extra cost because they show more consistent results among different workloads (SPECcpu2006, NAS and LMBENCH suites were studied). Moreover, when they need to be extended to support DVFS, less training time is needed to achieve optimal results because they capture better the DVFS behavior. In particular, we show that a 6x reduction of the modeling time, only increases the average error in one percentage point.

The rest of the paper is organized as follows. Section 2 presents the characteristics of the existing PMC-based modeling methods. Our proposed DVFS agnostic modeling methodology is detailed in Section 3. Section 4 describes the experimental framework and the models generated. The validation, the evaluation and the comparison of the modeling methods are explained in Section 5. Section 6 overviews the related work and finally, Section 7 summarizes our conclusions.

2. BACKGROUND

From the very beginning of the introduction of PMCs, the researchers proposed methodologies to use them as proxies of power consumption. Hence, the idea of using PMCs to estimate power consumption is not novel.

There have been proposed different methodologies targeting different purposes and requirements. For instance, there are methodologies that produce fast and simple power models just based on few PMCs –e.g. instructions per cycle (IPC) ratio– and other methodologies that produce more informative and accurate power models by using more PMCs.

All proposed approaches require to gather empirical data –training data– and then use statistical regression techniques to derive the power from a set of selected PMCs. Ideally, this process is done systematically assuming linear relationships between power and PMCs, although sometimes the human interaction is used to improve the model accuracy. For example, one can apply transformations to the input data or generate piece-wise models if the power trend changes at a given threshold [15].

The training data set used during the model formation influences its accuracy and generality. In the literature, we can find models trained using specifically designed microbenchmarks [15, 16, 18] or models trained using real applications [36]. The former claim more generality whereas the latter aim to be more accurate for such application domain.

Besides these characteristics, the methodologies to generate models can be classified in two categories depending on how the model is derived: the *Top-down* and *Bottom-up* approaches. The *Top-down* ones derive the power consumption treating the processor as a 'black-box', whereas the *Bottom-up* ones divide the architecture in power components and predict their standalone power consumption to later derive the overall power consumption. The following sections provide more insights about the pros and cons and the implications of using each one.

2.1. Top-down power models

The *Top-down* approaches aim to produce simple, fast and easy to deploy power models, allowing a quick implementation on different platforms. For that purpose, they avoid the need of specific knowledge of the underlying architecture, being the most architecture agnostic as possible.

We differentiate two subtypes depending on how the PMCs are selected. On the one hand, the simpler *Top-down* methodologies define models that use generic activity ratios present in all architectures such as IPC or memory requests per cycle to estimate the power consumption [24, 26].

On the other hand, there are methodologies that propose to evaluate several PMCs and then use statistical parameter selection techniques to select the PMCs that are more correlated with power consumption [29, 30]. Although this process can be automatized, sometimes manual tuning is used to improve the accuracy of the model [15, 31]. The main drawback of using statistical parameter selection techniques is that they bias the models generated towards the characteristics of the input training set, reducing their generality.

Regardless of the method used for selecting the PMCs, the number of PMCs used by this type of models usually does not exceed the maximum number of PMCs that can be tracked simultaneously on the modeled platform. The rationale behind this limitation is that these *Top-down* approaches aim to produce fast and simple power models. Thus, the number of multiplexed PMC sets is minimized, avoiding the multiplexing issue if possible.

Finally, *Top-down* multicore models are generated by stressing all the cores simultaneously. Then, to obtain the per-core power model, the overall power is divided by the number of cores assuming an idealistic resource sharing situation. This simplification fulfills the main aim of these models: being fast and simple. However, this simplifying assumption is only correct when all the cores of the Chip Multi-Processor (CMP) operate at the same DVFS state –in which the model was trained. As a result, these models are unable to breakdown per core power consumption when the frequency of the cores in the CMP is not homogeneous, limiting their applicability in current architectures.

2.2. Bottom-up power models

The *Bottom-up* approaches rely on some knowledge of the underlying architecture to produce more reliable and enlightening power models [16, 18]. For instance, they can provide a microarchitectural component-wise power consumption breakdown [25]. But this extra information and accuracy comes at the expense of requiring to multiplex the PMCs. As a result, they are more error prone for applications with very transient activity.

These approaches have a higher first-time deployment cost because they require to study the microarchitecture of the processor being modeled in order to define the power components and their respective PMCs. Then, it is required to specifically design microbenchmarks that stress each of the defined components.

Like the *Top-down* approaches, the *Bottom-up* ones generate the final model using statistical regression techniques. Again, some manual tuning may be needed to improve the accuracy of the model [25], although the model generation can be completely automatized if enough microbenchmarks are defined [16, 18].

When modeling several DVFS states, the higher first-time deployment cost drawback of the *Bottom-up* approaches is minimized since the overall modeling time is dominated by time needed to gather the experimental data rather than the one required to analyse the architecture and to design the microbenchmarks. Moreover, this is not a big drawback compared to the *Top-down* approaches since some of them also require the usage of specifically designed microbenchmarks to improve their accuracy [15].

The extension of *Bottom-up* models to multicores is straightforward and it does not have the limitations that the *Top-down* models have. The reason is that *Bottom-up* models for CMPs are just the addition of the particular single core models. As a result, modeling all the DVFS states of one core of a CMP is enough to be able to be able to model the entire CMP. Although sometimes, a final tunning is needed [16, 18] to fix the distribution of the static component. Nevertheless, this extra step is not a time consuming procedure.

In summary, the choice of the modeling methodology depends more on the requirements –accuracy, granularity, responsiveness, ...– of the environment in which the model will be used rather than the deployment cost. *Top-down* approaches generate faster models –higher granularity– than the *Bottom-up* ones, but the last ones are more accurate and reliable on current CMP architectures. Table 2 summarizes the characteristics of the different modeling methodologies.

3. DVFS AGNOSTIC POWER MODELS

In order to generate DVFS agnostic power models, we propose to start from the already proven valid power modeling methods presented in the previous section. The idea is to directly generate DVFS agnostic models from DVFS specific ones. To reach this goal, we model the relationships between DVFS specific model coefficients and DVFS state using regression techniques.

The rationale behind this procedure is to keep the model properties untouched. We want to keep similar accuracy and responsiveness results, while at the same time keep other properties such as decomposability.

The methodologies presented in Section 2 generate power models like:

$$P_{total} = \left(\sum_{i=1}^{i=numcomp} AR_i \times W_i \right) + W_{intercept} \quad (3)$$

where the total power (P_{total}) predicted is the sum of the power components defined ($numcomp$) –inputs of the regression– plus the regression intercept ($W_{intercept}$). Each power component is the product of an activity ratio formula (AR_i) based on PMCs and the weight assigned by the regression method (W_i).

If we generate a model for each of the available DVFS states of the modeled platform, we can obtain a vector of pairs of DVFS state (S_i) and weight for each component cmp (the $W_{intercept}$ and W_i in Equation 3)

TABLE 2. Summary of the general properties of the models depending on the modeling methodology used. Column values are with respect to the other modeling methods. (✓: low, ✓✓: average, ✓✓✓: high, N/A: Not applicable.)

Modeling Methodology	Accuracy	Responsiveness	1st time Deployment Cost	Workload Generality	Decomposability	Robustness to CMP	Online Speed
Top-down simple [24, 26]	✓	✓	✓	✓✓	N/A	✓	✓✓✓
Top-down complex [15, 29, 30, 31]	✓✓	✓✓	✓✓	✓	N/A	✓	✓✓✓
Bottom-up [16, 25, 18]	✓✓✓	✓✓✓	✓✓✓	✓✓✓	✓✓✓	✓✓✓	✓

of the form:

$$(S_0, W_{cmp_0}), (S_1, W_{cmp_1}), \ldots, (S_{n-1}, W_{cmp_{n-1}}), (S_n, W_{cmp_n}) \quad (4)$$

where n is the number of DVFS states. Taking into account that with regards to voltage (V) and frequency (F) the power (P) is modeled as:

$$P = C \times V^2 \times F \quad (5)$$

in which the capacitance (C) remains constant, we can model each component weight (W_{cmp_i}) as a function of the DVFS state (S_i):

$$W_{cmp_i} = f_{cmp}(S_i) \quad (6)$$

Then, we can apply regression analysis to obtain a f_{cmp} that models the power weight (W_{cmp_i}) from the DVFS state[5]. So that, replacing the variables from Equation 3, we end up with a DVFS agnostic power model:

$$P_{total} = \left(\sum_{i=1}^{i=numcomp} AR_i \times f_i(F) \right) + f_{intercept}(F) \quad (7)$$

in which the power is derived from the PMC activity ratios (AR_i) and the current DVFS frequency (F).

Notice that the idea of DVFS agnostic PMC-based power models is not novel. Previous works also present DVFS agnostic modeling methods [29, 30]. The work in [30] normalizes the PMC values using a frequency-independent metric, such as the execution time, and the work in [29] uses the different frequencies (e.g. core frequency, memory frequency) as independent model inputs –not related to PMCs– in conjunction with PMC values normalized to execution time. In both cases, they generate data for each of the DVFS states available on the platform in order to produce directly –in one step– a DVFS agnostic power model. Normalizing the activity ratios to execution time implies a loss of information and introduces errors. For instance, an activity ratio of n events per second can have different power consumption depending on the DVFS setting. However, the model would predict the same power consumption. Therefore, by definition, the properties of the models generated using these 'one-step' methods are affected by the DVFS mechanism.

Other options to generate DVFS agnostic power models are to extend the model by adding more inputs, or to scale the model using the well-known power-DVFS relation (Equation 5). The former is discarded because it modifies the model properties. For instance, the decomposability property of the models is affected. The later, the direct scaling, is discarded because it can not be generally applied. Often the detailed information –i.e. the exact voltage– is not available because is processor dependent [37]. Moreover, on current architectures the DVFS is not applied homogeneously among all the microarchitectural components [33]. As a result, directly scaling the models would affect their properties. In the end, the proposed two step method is the only one that does not directly affect the model properties and can be generally applicable.

3.1. Reducing the modeling time

Using the proposed method, as the number of cores and DVFS states grows, the modeling time increases and becomes dominated by the time required to gather the training and validation data. In fact, this problem was already faced in [21]. In that work, Snowdon *et al.* specifically describe the need to reduce the 18 days that would require to perform the entire platform modeling. Their solution was to use incomplete benchmarks suites (benchmark pruning), but still gather data for all the DVFS states available.

Clearly, reducing the number of training benchmarks reduces the modeling time. However, it also reduces the generality of the models generated since they are trained on fewer situations. Moreover, any other models derived from them –like the ones generated using our proposal to generate DVFS agnostic models– will also loose reliability.

Another drawback of performing benchmark pruning to speed up the modeling process is the lack of the generality. It can only be applied on the *Top-down* modeling methodologies, which are the ones that are more dependent to the training set. *Bottom-up* modeling methods require specifically designed training microbenchmark sets, which can not be reduced.

As a result, we propose to reduce the number of DVFS states in which gather training data for generating DVFS specific models. The usage of less DVFS specific models to generate the DVFS agnostic ones might reduce final reliability of them. Specially, this might be true for DVFS states in which we do not gather training data.

However, we will see in Section 5.3 that the number of DVFS states to model can be reduced –up to the

[5] We represent it numerically using its frequency (F). Notice that it would be more accurate to use the $V^2 \times F$ product. However, the DVFS state voltage information is not directly available as explained in footnote 6.

minimum of three– without affecting significantly the accuracy of the models. The rationale behind this observation is that the power consumption has a strong and well known relationship with the DVFS state (Equation 5). Moreover, other properties such as the generality, the responsiveness or the decomposability of the models are neither affected since the former depends on the input benchmark variety –which is not reduced– and the latter ones depend only on the model design, which we do not modify.

4. EXPERIMENTAL FRAMEWORK

4.1. Platform

We have performed the experiments on an Intel® Core™ 2 Duo T9300 [32] with 2GB of RAM. We only present results for one core enabled for brevity. Two core models of the same platform were generated and they lead to the same conclusions. Moreover, the modeling methodologies used in this work can also be applied to newer and different platforms in the same fashion, without affecting the conclusion and the generality of this work.

The modeled processor implements DVFS with frequency and voltage ranges that are [0.8,2.5]GHz and [0.75,1.25][6] volts respectively. We used the mechanism provided by the Linux® kernel to override the Differentiated System Description Table (DSDT). The DSDT table is part of the ACPI [38] specification and it supplies configuration information about a base system such as the DVFS states, also known as P-states. Our modified DSDT enabled several P-states between the platform pre-defined ones, allowing us to model the entire frequency range with higher granularity.

Table 3 summarizes all the P-states enabled. The first two columns show the core and the front-side bus (FSB) frequency. The core frequency is multiple of the external FSB clock speed. Frequencies below 1.2 GHz are achieved by halving the FSB frequency, increasing the latency of memory accesses. We will see later (in Section 4.4) that this architectural property also affects some power modeling methodologies.

The third and fourth columns are the frequency identifier (FID) and the voltage identifier respectively (VID). The fifth column, 1/2 Bus, indicates if the bit for halving the FSB frequency is set. The sixth column, 1/2 Clk, indicates if a half point should be added to the final Bus multiplier, shown in last but one column. The final core frequency is the product of the FID and FSB frequency, plus a half point if the 1/2 Clk bit is set. The last column shows the value that should be written to the `perf_ctl` MSR[7] in order to activate the DVFS state.

TABLE 3. Details of the ACPI *P-States* defined on the experimental platform.

Freq Ghz	FSB Mhz	FID	VID	1/2 Bus	1/2 Clk	Bus mul.	MSR perf_ctl
2.5	200	12	34	No	Yes	12.5	0x00004c22
2.4	200	12	33	No	No	12	0x00000c21
2.3	200	11	32	No	Yes	11.5	0x00004b20
2.2	200	11	31	No	No	11	0x00000b1f
2.1	200	10	30	No	Yes	10.5	0x00004a1e
2.0	200	10	30	No	No	10	0x00000a1e
1.9	200	9	29	No	Yes	9.5	0x0000491d
1.8	200	9	29	No	No	9	0x0000091d
1.7	200	8	28	No	Yes	8.5	0x0000481c
1.6	200	8	27	No	No	8	0x0000081b
1.5	200	7	26	No	Yes	7.5	0x0000471a
1.4	200	7	25	No	No	7	0x00000719
1.2B	200	6	23	No	No	6	0x00000617
1.2A	100	12	23	Yes	No	12	0x00008c17
1.1	100	11	21	Yes	No	11	0x00008b15
1.0	100	10	20	Yes	No	10	0x00008a14
0.9	100	9	19	Yes	No	9	0x00008913
0.8	100	8	19	Yes	No	8	0x00008813

4.2. Power and PMC gathering

The embedded controller firmware of the modeled platform –the BIOS– is used to gather power measurements. It provides information about the power source and fulfills the SMAPI specification [39]. This specification defines an interface to obtain power consumption measurements with a guaranteed granularity and accuracy. As a result, the device provides power measurements in a granularity of milliwatts with a maximum error of 2%. We have validated the readings against the ones from a Watts Up.Net [40] power meter for correctness. The measurements from both devices were consistent. In the end, we use the platform provided measurement device because it provided higher granularity and responsiveness.

We installed a Linux® kernel 2.6.28 with the required patches to allow access to the PMCs [41]. We boot it in level 1 mode[8] to nullify possible interferences. The `tp_smapi` [42] module was loaded to access the measurement device. We modified `pfmon` [43] to gather the PMCs and the power consumption simultaneously every two seconds. We switched off all sources of power consumption –e.g. the display– that we do not want to interfere. For platform components where it is not possible to switch off, we configured them at constant operation mode. Under this conditions, we track the power consumption during one minute before each experiment and we find the baseline power consumption for the idle system to be 15765 mW. The fact that for all the experiments carried out we found a fairly constant idle power consumption demonstrates that we nullify possible interferences. For the purpose of power

[6]Each processor is programmed with a maximum valid voltage identification value (VID), which is set at manufacturing and cannot be altered. Individual maximum VID values are calibrated during manufacturing such that two processors at the same frequency may have different settings within the VID range. [37]

[7]MSR stands for Model Specific Register. The `perf_ctl` and `perf_status` registers are in charge of the DVFS mechanism in our experimental platform.

[8]Also known as standalone mode. Only the minimum required system services and processes are executed.

TABLE 4. Experiments executed for each of the 18 DVFS states available on the experimental platform.

Suite	# Bench.	Bench. time	Suite time	Suite samples
SPECcpu2006	26	5 min.	2h10min.	3900
NAS	5	5 min.	25min.	750
LMBENCH	44	5 min.	3h40min.	6600
microbenchmarks	98	1 min.	1h38min.	2940
TOTAL	173	-	7h53min.	14190

estimation only, we assume that baseline, 15765 mW, to be the power consumption of the platform except for the processor and the memory. Similar set-ups and measurement methodologies were already used and validated in the literature [15, 16, 19, 18, 20].

4.3. Benchmarks

Under the explained conditions, we have collected the data to generate and validate the models. We executed the SPECcpu2006 [44], the NAS [45], and the LMBENCH [46] benchmark suites. They were compiled using `gcc` with `-O2` optimization flag.

The SPECcpu2006 benchmarks are designed to stress integer and floating point processor performance, including the memory hierarchy. The NAS benchmarks are designed to evaluate parallel systems. However, we configured them to use only one thread that solves the largest problem size. This configuration maximizes the stress of the memory subsystem. Finally, the LMBENCH benchmarks evaluate the performance of UNIX systems. They stress various OS related aspects such as process creation latency, signal handling or the TCP stack. Thus, the evaluation of the three different suites allows us to provide a consistent and sound study of the modeling methodologies for a wide range of situations, corroborating the generality of the proposal. Moreover, we developed a bunch of microbenchmarks that stress specific parts of the architecture in order to gather the training data required for the *Bottom-up* methodologies.

Table 4 describes the experiments executed for each DVFS state available in our platform. In total, we ran 3366 experiments to obtain 255420 samples of PMCs and power. The overall process required about 6 days to complete, providing an evidence of how time consuming already is the task of fully modeling a single core platform.

4.4. Data processing and model generation

We used $\mathcal{R}$ [47] to perform all the statistical regression analysis. $\mathcal{R}$ is a software environment for statistical computing which also provides a scripting interface. We developed a set of $\mathcal{R}$ scripts to automatize the generation of the power models. Hence, no human intervention nor expert manual tuning have been performed to form the models. This is an important point to remark, since fine-tuning each DVFS specific model would be impractical.

Table 5 describes the models generated using the $\mathcal{R}$ scripts. `TD_A` and `TD_B` are generated using the most simple *Top-down* approaches explained in Section 2.1 which use common activity ratios present in all platforms. In this case, `TD_A` uses IPC and `TD_B` uses IPC and memory access ratio as input respectively. The `TD_C` model was generated using the counters suggested by the most accurate *Top-down* approaches [15, 29, 30], which use a guided statistical parameter selection method to select the input PMCs. We ended up with a similar subset of counters as the ones selected in [15], which are IPC, FP_EXEC, L2_MISS and STALL ratios. The other model, `BU`, was generated by applying a *Bottom-up* methodology. Concretely, we followed the methodology we presented in [16, 18], in which for each functional unit –e.g. Integer, Floating-point, L1, L2,...– a weight is derived using specifically designed microbenchmarks. We refer the reader to these works [16, 18] for a comprehensive explanation of the modeling method.

Once the input PMCs are defined for each model, we applied the regression analysis on the training data for each DVFS state in order to get the model weights for each state (the DVFS specific models). The *Top-down* models were trained using the SPECcpu2006 data as input, whereas the *Bottom-up* one used the microbenchmark data as training data. We do not show the actual weights obtained for each model and DVFS state to not saturate the explanation. Nevertheless, we note that as expected all weights present a direct positive relation with frequency.

Then, as explained in Section 3, we perform a second regression on the weights using the DVFS state frequency as input. In order to choose a modeling function (f_i) for each component, the script applies both linear regression and exponential regression. Then, it selects the one with better correlation coefficient (R). As a result, the f_i from Equation 6 is defined as:

$$f_i(F) = \begin{cases} (\alpha_i \times F) + \beta_i & if(R_{lin} \geq R_{exp}), \\ (\gamma_i \times \delta_i^F) + \epsilon_i & otherwise \end{cases} \quad (8)$$

where α_i, β_i, γ_i, δ_i and ϵ_i are coefficients generated by the regression applied. In almost all the cases, the regressions showed high –higher than 0.9– correlation coefficients (R_{lin}, R_{exp}), indicating as we will see in Section 5.1 that the generalization of the DVFS specific models to DVFS agnostic ones does not affect noticeably the properties of the models.

A special case is the regression for the memory component of the `BU` model. We did a piece-wise regression at 1.2GHz because as explained in Section 4.1 the frequency of the memory bus changes at that point and consequently its weight trend changes. Notice that this special behavior does not affect the model generation automation since the memory frequency change is an architectural characteristic known a priori. Hence, the modeling $\mathcal{R}$ script is already designed to

TABLE 5. Name, modeling methodology and the input performance monitoring counters of the power models generated.

Model	Method	Input PMCs
TD_A	Top-down	UOPS_RETIRED,CPU_CLK_UNHALTED
TD_B	Top-down	UOPS_RETIRED,CPU_CLK_UNHALTED, BUS_TRANS_BURST
TD_C	Top-down	UOPS_RETIRED,CPU_CLK_UNHALTED, L2_LINES_IN,FP_COMP_OPS_EXE, RESOURCE_STALLS
BU	Bottom-up	UOPS_RETIRED, FP_COMP_OPS_EXE, CPU_CLK_UNHALTED, BUS_CLK_UNHALTED, SIMD_UOPS_EXEC, BR_INST_DECODED, L1D_ALL_REF, L2_RQSTS, BUS_DRDY_CLOCKS

take that effect into account. *Top-down* models do not suffer from these changes on weight trends because they do not model each component separately.

We named each of the generated DVFS agnostic models after their original source model. Concretely, they are named TD_A_DVFS, TD_B_DVFS, TD_C_DVFS and BU_DVFS respectively. The final models generated are shown in Figure 1. We use them to get estimated power consumption traces as shown in Figure 2, in which we see the predictions of the four models in conjunction to the real power trace. We do not include previous proposals of DVFS agnostic counter based power models [29, 30] because the main focus of this work is to compare *Top-down* vs *Bottom-up* models when they are extended –using our generally applicable proposal– to be DVFS agnostic.

5. MODEL VALIDATION

In this section we validate the models generated in terms of *accuracy*, *responsiveness* and *robustness* to DVFS. The accuracy is mandatory because an inaccurate model is useless. The *responsiveness* of a model –its capacity to track dynamic variations– is important for the applicability of the model as shown in previous works [16, 18]. Finally, we define *robustness* as the capacity of the model to maintain the *accuracy* and the *responsiveness* when the number of DVFS states used to generate it is reduced. So, the more *robust* a model is, the more quickly it can be produced.

5.1. Accuracy validation

5.1.1. Average error

We evaluate the accuracy using the percentage average absolute prediction error (PAAE)[27]. The PAAE of the models for each DVFS state and suite are shown in Figure 3. The models are sorted to facilitate the comparison between the DVFS specific models and their respective DVFS agnostic one. The most important point to remark is that in overall, for the three benchmark suites, the DVFS agnostic models do not introduce a significant error with respect to the DVFS specific ones.

Figure 3(a) shows the PAAE for the SPECcpu2006 suite. All the models present low PAAE (below 3% in general) for this suite. In this figure, it is important to notice that the *Bottom-up* model shows competitive results even though *Top-down* models are over-trained for this suite (They use the same training and validation benchmark suite). We also see the TD_A.* models increase their PAAE when the frequency decreases. This is because these models do not take into account the memory activity, whose relative contribution to the total power consumption increases when processor frequency decreases.

Figure 3(b) shows the average error for the NAS suite. In this case, we see that the TD_A and the TD_A_DVFS models are very inaccurate, with up to 22% error in predictions. Again, the reason is that these simple models do not take into account the memory activity, which is important in the NAS suite. Besides, the other models show a slightly higher PAAEs than in Figure 3(a), but they remain in an acceptable values (in general between 2% and 7%). Notice that now, when any of the models is over-trained, the *Bottom-up* model still shows very competitive results.

The LMBENCH suite is the one with more benchmarks (44) and more heterogeneity between them, covering a wider range of situations than the other benchmark suites. The results for this suite, shown in Figure 3(c), corroborate that the *Bottom-up* models provide higher accuracy and generality than *Top-down* ones. In this case, we see that *Bottom-up* models show PAAEs between 2% and 4%, whereas the *Top-down* ones double the prediction error, with values between 7% and 9% in overall.

In summary, these results show that the generalization from DVFS specific models to DVFS agnostic ones does not affect noticeably the accuracy of the models. As a result, the model generated using the *Bottom-up* method exhibits more generality.

5.1.2. Coverage

The coverage of the generated models is another aspect to evaluate the overall accuracy of them. Figure 4 shows the Empirical Cumulative Distribution Function (ECDF) of model errors for the three suites analyzed.

In Figure 4(a), we see a fairly similar ECDF between the models even though *Top-down* ones are over-trained. Only the simplest TD_A_DVFS shows noticeable worse results. Similar trends are obtained for the NAS suite, shown in Figure 4(b).

The LMBENCH results, shown in Figure 4(c), exhibit a more clear difference between the models. In this case the BU_DVFS model performs about 80% of predictions with less than 10% error. In contrast, for the same error threshold, the TD_C_DVFS model, which was the best *Top-down* one in the other suites, covers only about the 45% of the space.

This corroborates the results found in previous

$$
\begin{aligned}
TD\_A\_DVFS &= (2.27 \times (12.5^F) - 263) \times AR_{ipc} + ((3895 \times F) + 1040) \\
TD\_B\_DVFS &= (76 \times (3.28^F)) \times AR_{ipc} + (34549 \times (0.68^F)) \times AR_{mem} + ((3721 \times F) + 268) \\
TD\_C\_DVFS &= ((279 \times F) - 257) \times AR_{ipc} + ((421 \times F) + 273) \times AR_{fp} + ((-1946 \times F) + 694) \times AR_{stalls} \\
&+ ((5266 \times F) - 423) + ((110437 \times F) + 85828) \times AR_{L2} \\
BU\_DVFS &= ((269 \times F) - 117) \times AR_{ipc} + (35.5 \times (2.66^F)) \times AR_{int} + ((250 \times F) - 139) \times AR_{fp} \\
&+ ((794 \times F) - 385) \times AR_{bpu} + (94.5 \times (2.55^F)) \times AR_{L1} + ((10637 \times F) - 4517) \times AR_{L2} + ((3567 \times F) + 173) \\
&+ \begin{cases} ((562 \times F) + 13553) \times AR_{mem} & if\ (F \geq 1.2GHz), \\ ((709 \times F) + 16268) \times AR_{mem} & otherwise \end{cases}
\end{aligned}
$$

FIGURE 1. Detailed formulas of the DVFS agnostic power models generated from the DVFS specific ones after applying the proposed extension methodology.

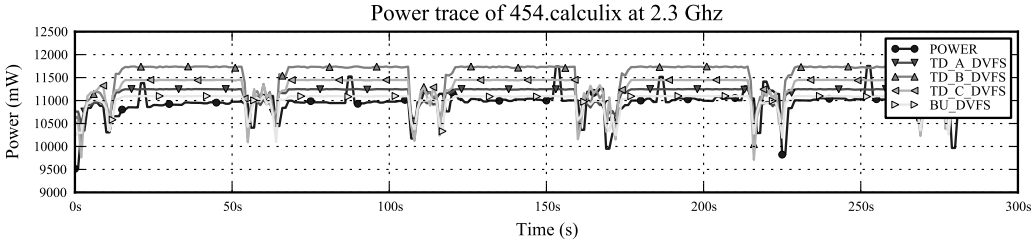
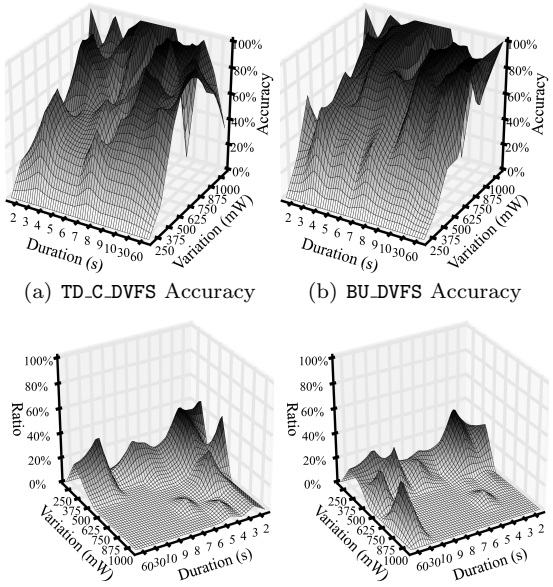


FIGURE 2. Power trace and the prediction of the models of the SPECcpu2006 `454.calculix` benchmark running at 2.3 Ghz.



(a) `TD_C_DVFS` Accuracy (b) `BU_DVFS` Accuracy

(c) `TD_C_DVFS` False Positives (d) `BU_DVFS` False Positives

FIGURE 5. Phase detection accuracy and false positive ratios of the models for the SPECcpu2006 suite. The results are classified by phase variation (x-axis) and phase duration (y-axis).

section: the *Bottom-up* approaches provide higher coverage and generality. Moreover, we corroborate that the extension to be DVFS agnostic does not affect the coverage of the model generated.

TABLE 6. Phase detection accuracy and false positive ratio of the models for the SPECcpu2006 suite.

(a) Phase detection accuracy

Variation	TD_C_DVFS	BU_DVFS
All	50.5%	46.8%
>1000mW	76.4%	84.0%

(b) False positive ratio

Variation	TD_C_DVFS	BU_DVFS
All	147.9%	75.8%
>1000mW	27.8%	0.5%

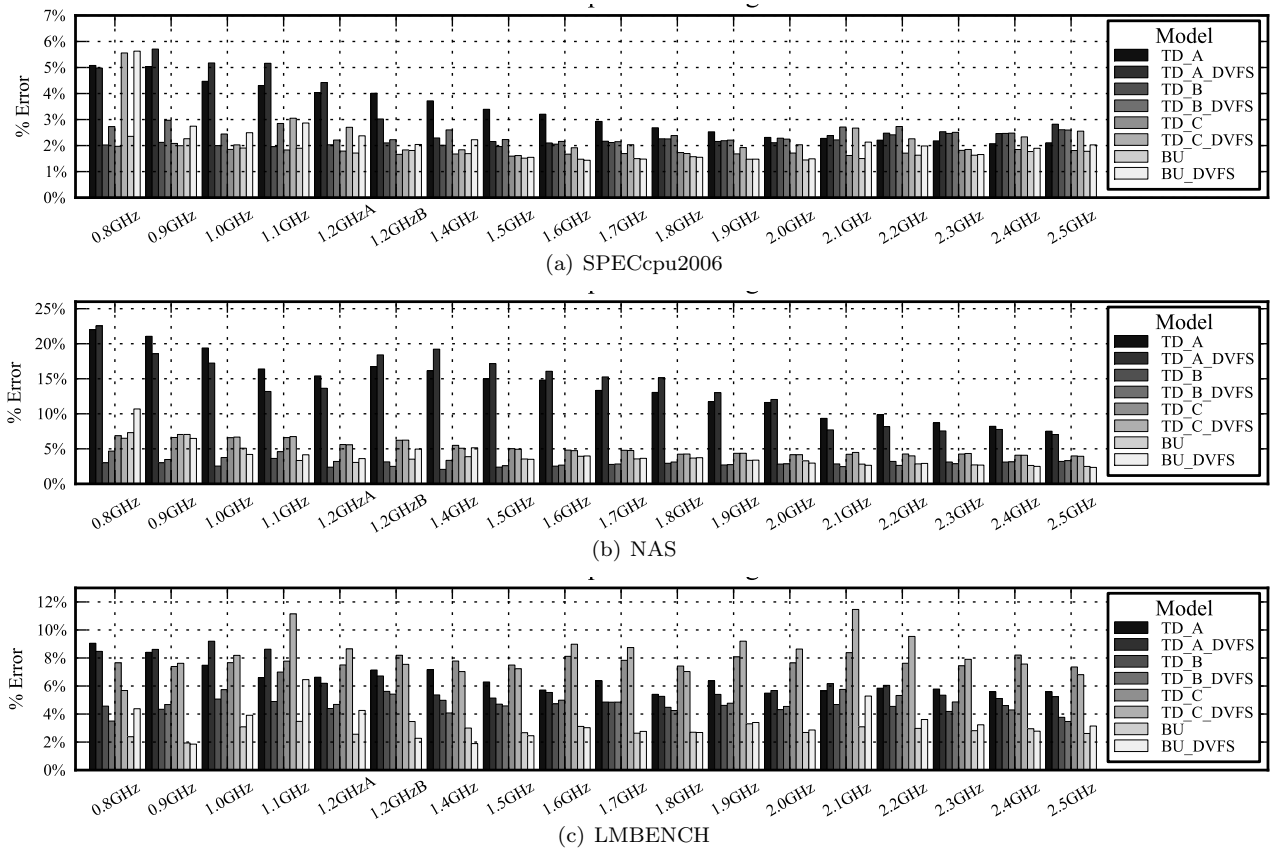
5.2. Responsiveness

We evaluate the responsiveness of the models by applying the same phase detection algorithm to both, the modeled and the actual power traces, and comparing the results. Similar to our previous works [16, 18], we apply a first pivot clustering algorithm with a minimum phase threshold of 250mW.

5.2.1. Phase detection accuracy

The top two charts in Figure 5 show the phase detection accuracy of the `TD_C_DVFS` and `BU_DVFS` models for the entire SPECcpu2006 suite on all the frequencies. The simpler models, `TD_A_DVFS` and `TD_B_DVFS`, are not shown for brevity. Moreover, the results of the DVFS specific models are not shown because the DVFS agnostic and DVFS specific models exhibit a very similar behavior in terms of responsiveness. The results are classified by phase duration and phase variation in order to understand better the responsiveness trends.

In terms of variation, the general trend in both

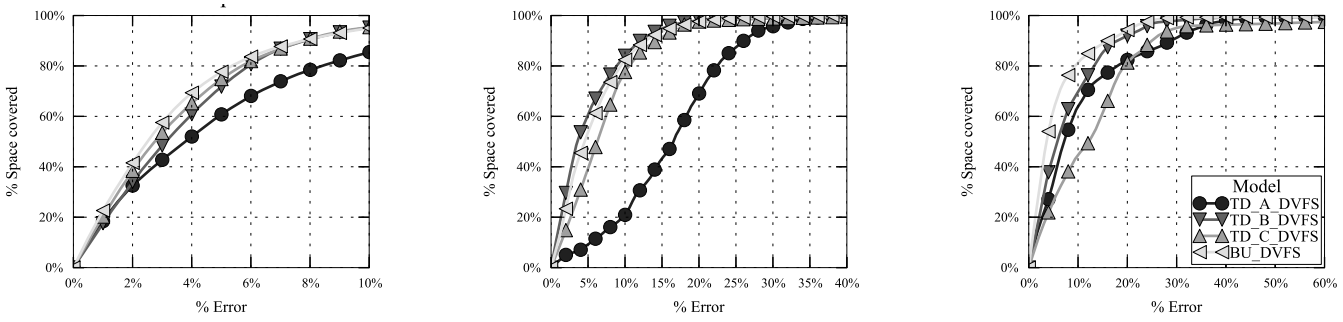


(a) SPECcpu2006

(b) NAS

(c) LMBENCH

FIGURE 3. Average absolute percentage prediction errors of the models for each DVFS state and benchmark suite.



(a) SPECcpu2006 (b) NAS (c) LMBENCH

FIGURE 4. Empirical Cumulative Distributions Functions (ECDF) of model errors for the three suites analyzed. The ECDF shows percentage of space predicted (y-axis) under a given error (x-axis).

models is that the higher the variation the higher the phase detection accuracy. In terms of phase duration there is not a clear trend to comment. These results are similar to the ones shown by previous works [16, 18], corroborating that the DVFS extension method does not modify the model responsiveness property. Moreover, even not shown here, the results of the `TD_A_DVFS` and `TD_B_DVFS` models also behave similarly as their DVFS specific counterparts.

In more detail, the direct comparison of both models show that both models, the `BU_DVFS` model and the `TD_C_DVFS`, present similar results in overall. However, for higher variations the `BU_DVFS` model outperforms the `TD_C_DVFS` one. For instance, the *Bottom-up* model is able to detect 100% of phases with variations above 1 watt and duration more than 60 seconds, whereas the `TD_C_DVFS` detects less than 40% of such phases.

Table 6(a) shows the average accuracy results of the models for all phases and durations and for phases with more than 1000mW variations. We see that on average both models detect a similar amount of phases, around 50% of the 3027 power phases. Nevertheless, for the important phases, the ones with high variations (>1000mW), the `BU_DVFS` model outperforms the other

by detecting 84% of the 187 power phases. The *Top-down* model has around 8 percentage points less accuracy than the *Bottom-up* one, detecting only 76.4% of the phases.

Notice that, by implementing a more complex phase detection algorithm instead of using the simple first pivot clustering algorithm, the accuracy results could be higher[9]. However, the results shown already provide an overview of the overall phase detection accuracy trends of the different modeling approaches.

5.2.2. False positives

The drawback of having a more responsive power model is its false positive ratio, which we define as the percentage of non-existent phases predicted with respect to the total number of phases. The evaluation of the false positives ratio of a model is important to know its applicability because an over-reactive model is also useless [16, 18].

The bottom two charts in Figure 5 show the classification in variation and duration of the false positives. In both models the general trend is that most of the non-existent phases reported are concentrated on low variations due to the same reason as before; in such low variations phases, a slight error in the prediction results in the detection of a non-existent phase.

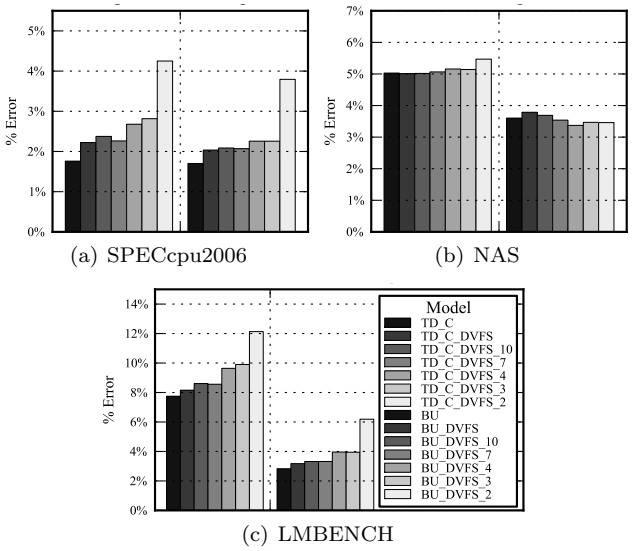
Table 6(b) shows the summary of false positive ratios for all the phases and for the ones with more than 1000mW variation. It is important to notice that in contrast to what one might expect, the `BU_DVFS` model reports much less false positives than the `TD_C` model, 75.2% versus 147.9% respectively. And for high variations, the important ones, the `BU_DVFS` model provides a very low false positive ratio of 0.50%, whereas the *top-down* model has a 27.8% false positive ratio.

In summary, the *Bottom-up* DVFS agnostic model provides the better trade-off between phase detection accuracy and false positive ratio (specially for important variations). Moreover, it provides the lowest PAAEs for the three benchmark suites analysed. These results, which are in accordance with previous observations [16, 18], confirm that the extension of DVFS specific models to DVFS agnostic ones does not affect the responsiveness property of the models.

5.3. Robustness to DVFS

We have already explained that our extension from DVFS specific models to DVFS agnostic ones does not affect noticeably the basic properties of the models: accuracy and responsiveness. In this section, we study the effects on these properties when the number of DVFS states used to train the DVFS agnostic models is reduced. The reduction of the number of DVFS states is important to minimize the time required to fully model a given platform.

[9]The study of different phase detection algorithms is out of the scope of this paper. In any case, one should notice that the algorithms have to be simple enough to be used online without incurring in overheads.



(a) SPECcpu2006 (b) NAS

(c) LMBENCH

FIGURE 6. Average PAAE of the models for the SPECcpu2006, NAS and LMBENCH benchmark suites for all the DVFS states.

To perform the robustness to DVFS evaluation, we re-applied the step explained in Section 4.4 but modifying the number of DVFS states used as input of the second regression. We generated models using 2,3,4,7 and 10 states of the 18 available in our system. The frequencies selected in all the cases were equally distant (e.g. the model `TD_C_DVFS_3` is generated using the 0.8, 1.7 and 2.5 frequencies and the `BU_DVFS_4` is generated using the 0.8, 1.4, 2 and 2.5 frequencies).

5.3.1. Accuracy

Figure 6 shows the PAAEs of the model predictions for the three suites studied and for all the 18 frequencies. As expected, the average error increases when the number of the training DVFS states is reduced. However, it is important to note that this increment is not very significant, demonstrating that a continuous DVFS mechanism can be fully modeled by training only few –three– DVFS states.

In general, the reduction from 18 DVFS states to only 3 (6x reduction) shows increments of less than 1 percentage point in the PAAEs for both models and the three suites. Only the `TD_C_DVFS` model for the LMBENCH suite shows a higher increment, about 2 percentage points increment in PAAE for the same reduction. As a result, for this particular case, a 6x reduction of the data gathering process (modeling time) only reduces about 1 percentage point the accuracy of the models.

The models trained only using two frequencies, the minimal and the maximum one, exhibit noticeable higher PAAEs. The reason is that they are over-simplistic (linear), omitting the exponential relation of

power with regards to the DVFS state. For instance, in Figure 6(a) and 6(c), they have about 2 percentage points more PAAE than the ones trained using 3 DVFS states.

In detail, Figure 6(b) shows the PAAEs for the NAS suite. In this case, we see that the average error of the models is not affected by the number of DVFS states used as much as in the other suites (less than 1% variations). The reason is that the power related to the memory –which is the main component stressed by these benchmarks– can be easily modeled using just two states. This is because such component is less affected by the DVFS state changes[10].

Figure 7 shows the PAAEs of all the `TD_C` related models against the `BU_DVFS_3` one for the LMBENCH suite, the most generic one. The important point to remark is that the `BU_DVFS_3`, which is generated from only 3 DVFS states, is more accurate than the rest for any particular DVFS state. Notice that this is true even when the *Top-down* models are specific for that DVFS state (`TD_C`) or generated from more DVFS states.

From these results, we conclude that the methodology presented in Section 3 is very robust to the reduction of the number of DVFS states in terms of accuracy. As a result, it does not affect the previous assumptions regarding the properties and characteristics of the modeling methodologies. In this case, we have seen that the DVFS agnostic `BU` model outperforms in the same way as the DVFS specific `BU` model did, the DVFS specific `TD_C` models.

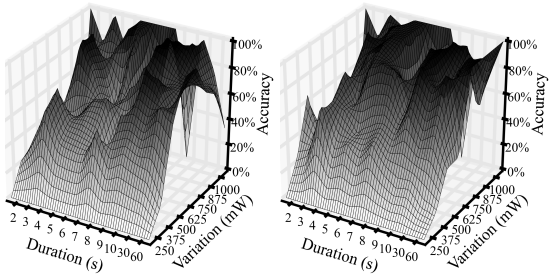
5.3.2. Responsiveness

In order to validate if the responsiveness property is kept when the number of DVFS states is reduced, we re-applied the responsiveness validation to the models generated using less DVFS states as input. Figure 8 shows the phase detection accuracy plots of the models generated using only 3 DVFS states. If we compare the charts in this figure against the ones at the top of Figure 5, we see very similar trends and results.

This observation leads us to conclude that the responsiveness of a DVFS agnostic model is neither affected by the reduction of the DVFS training states. Again, this property, the responsiveness of a model, like the other ones, such as the accuracy, is defined only by the model design: the number of the components, its associated input PMCs and the training set (for a single DVFS state).

From the facts shown through the paper, we conclude that in order to fully model a DVFS enabled platform, it is more important to generate an accurate and responsive minimum set of DVFS specific models and then apply our extension method than to reduce the training set on each DVFS state and explore more states. As demonstrated in this paper, performing the modeling using the proposed 2-step method allows the reduction of the modeling time, keeping at the same time the modeling properties.

[10]In our testbed, the DVFS does not change the memory voltage, and has operating frequencies as shown in Table 3.



(a) `TD_C_DVFS_3` Accuracy (b) `BU_DVFS_3` Accuracy

FIGURE 8. Phase detection accuracy and false positive ratios of the models generated using only 3 DVFS states for the SPECcpu2006 suite. The results are classified by phase variation (x-axis) and phase duration (y-axis).

6. RELATED WORK

In Section 2, we discussed the state of the art on power modeling methods based on performance counters. In this section, we provide wider view of the importance of power modeling methods in order to improve the efficiency of Information and Communication Technologies (ICTs) systems.

Previous works on power modeling also focused on its usage to guide power aware policies, characterize systems or evaluate new proposals. For instance, Tao Li *et al.* [26] study the power consumption at the OS level. For this purpose, a simple *Top-down* model is built based on the broadly accepted observation that there is a positive relation between instructions retired and power consumption. F. Bellosa [24] implements an OS power aware policy based on PMCs collected at runtime, which are used to estimate the power consumption. A similar approach is used by V. Jiménez *et al.* [31], in which a model is generated to characterize the power consumption of a POWER6 system. Moreover, our previous work in local memories [28] used the *Bottom-up* model presented by Isci *et al.* [25] to evaluate the power consumption.

Besides, power models are required in order to perform architectural design space explorations on simulation platforms. For instance, in [48], M.Hsieh *et al.* presents a framework that integrates various power and thermal libraries such as Wattch [49]. Also, M.Powell *et al.* [50] proposes a method based on few PMCs to estimate the activity of several microarchitectural structures and then predict their power consumption.

Counter-based DVFS aware power models can be used to guide energy aware techniques and provide feedback for improving a wide-spectrum of proposed energy-conscious solutions. There exists several approaches that with accurate on-line power estimates, such as

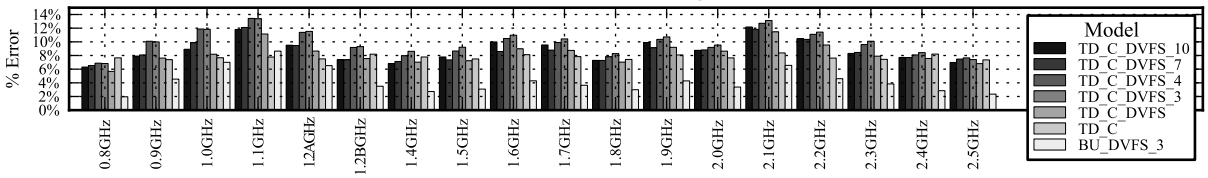


FIGURE 7. Average PAAE of the `TD_C*` and `BU_DVFS_3` models for the LMBENCH suite.

the ones provided by counter-based power models, can be improved or implemented on real platforms. In [51], T.V. Do proposes an energy-aware policy for operating server farms. The presented policy manages the hosts power modes depending on the workload in order to save energy; In [12], a power-aware partitioner and scheduler for real-time systems is presented. The proposed mechanism uses DVFS to save energy while guaranteeing the tasks deadlines; In [34] a benchmarking methodology for measuring performance and energy efficiency for heterogeneous systems is presented; and in [11], the authors present a to solution the inter-task and intra-task DVFS problems simultaneously, taking into account low power (sleep) modes. In all the variety of these works, the authors could have used counter-based power models to quickly evaluate their solutions.

In [6], a review of the approaches currently used for energy-efficient operation of ICTs systems is presented. The paper points out some key research demands that come up when such green-approaches are applied in cloud computing environments. The improvements of the power modeling techniques, like the one presented in this work, which have been applied in cloud computing environments [19, 20], help to improve the insights about ICTs power consumption. As a result, they facilitate the task of developing solutions for solving the demands pointed out in [6].

Finally, the importance of improving the energy efficiency at all levels of ICTs infrastructures has become a topic of interests for researchers. In that sense, all the subsystems of such infrastructures with associated performance counters, like the network subsystem, can be subject to be empirically modeled using a methodology similar to the one presented in this paper. For instance, in [52], an analytical model is presented to grain insights about how to use routing control in a network for reducing energy consumption, maintaining a certain level of QoS; and in [53], Lent presents a simulation model to evaluate the power consumption of computer networks. The simulation model is calibrated using empirical measurements, and then the system is modeled based in terms of data traffic. All these works remark the importance of energy-conscious models to understand the complexity of the power/performance trade-offs of current ICTs systems.

7. CONCLUSION

In this paper, we compared different PMC-based modeling approaches –*Top-down* and *Bottom-up*–. Specially, we studied how they interact with the currently ubiquitous Dynamic Voltage and Frequency Scaling (DVFS) mechanism. We derived 4 different DVFS agnostic power models from existing DVFS specific models and proposed an extension methodology that does not affect the model properties. Moreover, we studied the reduction of the modeling time in order to enable the usage of PMC based models on future architectures with several DVFS states and cores.

The study performed on a 18 DVFS states Intel® Core™ 2 platform using the SPECcpu2006, NAS and LMBENCH benchmark suites showed that DVFS agnostic power models provide the same levels of accuracy and responsiveness as the DVFS specific ones. Moreover, we showed that DVFS agnostic models trained only using 3 DVFS states provide the same levels of accuracy (1% increment on average error) and responsiveness as the ones trained using all the DVFS states. As a result, for this particular case, the modeling time can be reduced by 6 without compromising the reliability of the models generated.

The proof that DVFS agnostic models generated only from few DVFS states are still reliable, enables the generation of PMC based models for new architectures and consequently, it allows to continue the research on power aware policies based on them.

FUNDING

This work was supported by the Ministry of Science and Innovation of Spain (CICYT) [TIN-2007-60625]; the Generalitat de Catalunya [2009-SGR-980]; and the European Commission in the context of the SARC Project #27648 (FP6).

ACKNOWLEDGEMENTS

The authors acknowledge the support of their body funding, the Barcelona Supercomputing Center (BSC). We would like to thank the anonymous reviewers for their comments that helped us significantly improve the presentation of our work. We also want to thank the colleagues of our department and research group for their helpful comments.

REFERENCES

[1] Mudge, T. (2001) Power: A first-class architectural design constraint. *Computer*, **34**, 52–58.

[2] Ranganathan, P., Leech, P., Irwin, D., and Chase, J. (2006) Ensemble-level power management for dense blade servers. *ISCA '06: Proceedings of the 33rd annual international symposium on Computer Architecture*, Washington, DC, USA, June, pp. 66–77. IEEE Computer Society.

[3] Skadron, K., Stan, M. R., Huang, W., Velusamy, S., Sankaranarayanan, K., and Tarjan, D. (2003) Temperature-aware microarchitecture. *ISCA '03: Proceedings of the 30th annual international symposium on Computer architecture*, New York, NY, USA, June, pp. 2–13. ACM.

[4] Borkar, S., Karnik, T., Narendra, S., Tschanz, J., Keshavarzi, A., and De, V. (2003) Parameter variations and impact on circuits and microarchitecture. *DAC '03: Proceedings of the 40th annual Design Automation Conference*, New York, NY, USA, June, pp. 338–342. ACM.

[5] Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., and Doyle, R. P. (2001) Managing energy and server resources in hosting centers. *SOSP '01: Proceedings of the eighteenth ACM symposium on Operating systems principles*, New York, NY, USA, October, pp. 103–116. ACM.

[6] Berl, A., Gelenbe, E., Di Girolamo, M., Giuliani, G., De Meer, H., Dang, M. Q., and Pentikousis, K. (2010) Energy-efficient cloud computing. *The Computer Journal*, **53**, 1045–1051.

[7] Kuo, Y.-M., Weng, S.-H., and Chang, S.-C. (2008) A novel sequential circuit optimization with clock gating logic. *ICCAD '08: Proceedings of the 2008 IEEE/ACM International Conference on Computer-Aided Design*, Piscataway, NJ, USA, November, pp. 230–233. IEEE Press.

[8] Donald, J. and Martonosi, M. (2006) Techniques for multicore thermal management: Classification and new exploration. *ISCA '06: Proceedings of the 33rd annual international symposium on Computer Architecture*, Washington, DC, USA, June, pp. 78–88. IEEE Computer Society.

[9] Flautner, K. and Mudge, T. (2002) Vertigo: automatic performance-setting for linux. *Proceedings of the 5th symposium on Operating systems design and implementation*, New York, NY, USA, December OSDI '02, pp. 105–116. ACM.

[10] Isci, C., Buyuktosunoglu, A., Cher, C.-Y., Bose, P., and Martonosi, M. (2006) An analysis of efficient multi-core global power management policies: Maximizing performance for a given power budget. *Proceedings of the 39th Annual IEEE/ACM International Symposium on Microarchitecture*, Washington, DC, USA, December MICRO 39, pp. 347–358. IEEE Computer Society.

[11] Seo, H., Seo, J., and Kim, T. (2012) Algorithms for combined inter- and intra-task dynamic voltage scaling. *The Computer Journal*, **PrePrints**.

[12] March, J. L., Sahuquillo, J., Hassan, H., Petit, S., and Duato, J. (2011) A new energy-aware dynamic task set partitioning algorithm for soft and hard embedded real-time systems. *The Computer Journal*, **54**, 1282–1294.

[13] Esmaeilzadeh, H., Cao, T., Xi, Y., Blackburn, S. M., and McKinley, K. S. (2011) Looking back on the language and hardware revolutions: measured power, performance, and scaling. *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*, New York, NY, USA, March ASPLOS '11, pp. 319–332. ACM.

[14] Rajamani, K., Rawson, F., Ware, M., Hanson, H., Carter, J., Rosedahl, T., Geissler, A., Silva, G., and Hua, H. (2010) Power-performance management on an IBM POWER7 server. *Proceedings of the 16th ACM/IEEE international symposium on Low power electronics and design*, New York, NY, USA, August ISLPED '10, pp. 201–206. ACM.

[15] Singh, K., Bhadauria, M., and McKee, S. A. (2008) Real time power estimation and thread scheduling via performance counters. *SIGARCH Comput. Archit. News*, **37**, 46–55.

[16] Bertran, R., Gonzalez, M., Martorell, X., Navarro, N., and Ayguade, E. (2010) Decomposable and responsive power models for multicore processors using performance counters. *ICS '10: Proceedings of the 24th ACM International Conference on Supercomputing*, Tsukuba, Ibaraki, Japan, June, pp. 147–158. ACM.

[17] Bircher, W. L. and John, L. K. (2012) Complete system power estimation using processor performance events. *IEEE Transactions on Computers*, **61**.

[18] Bertran, R., Tallada, M. G., Martorell, X., Navarro, N., and Ayguade, E. (2012) A systematic methodology to generate decomposable and responsive power models for cmps. *IEEE Transactions on Computers*, **Preprints**.

[19] Bertran, R., Becerra, Y., Carrera, D., Beltran, V., Gonzalez, M., Martorell, X., Torres, J., and Ayguade, E. (2010) Accurate energy accounting for shared virtualized environments using pmc-based power modeling techniques. *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, Piscataway, NJ, USA, oct., pp. 1 –8. IEEE.

[20] Bertran, R., Becerra, Y., Carrera, D., Beltran, V., Gonzlez, M., Martorell, X., Navarro, N., Torres, J., and Ayguad, E. (2012) Energy accounting for shared virtualized environments under dvfs using pmc-based power models. *Future Generation Computer Systems*, **28**, 457 – 468.

[21] Snowdon, D. C., Le Sueur, E., Petters, S. M., and Heiser, G. (2009) Koala: a platform for os-level power management. *Proceedings of the 4th ACM European conference on Computer systems*, New York, NY, USA, April EuroSys '09, pp. 289–302. ACM.

[22] Isci, C. and Martonosi, M. (2006) Phase characterization for power: Evaluating control-flow-based and event-counter-based techniques. *HPCA-12*, Austin, TX, USA, February, pp. 121–132. Princeton University IEEE Press.

[23] Bhattacharjee, A. and Martonosi, M. (2009) Thread criticality predictors for dynamic performance, power, and resource management in chip multiprocessors. *ISCA '09: Proceedings of the 36th annual international symposium on Computer architecture*, New York, NY, USA, June, pp. 290–301. ACM.

[24] Bellosa, F. (2000) The benefits of event: driven energy accounting in power-sensitive systems. *Proceedings of the 9th workshop on ACM SIGOPS European workshop: beyond the PC: new challenges for the operating system*, New York, NY, USA, September EW 9, pp. 37–42. ACM.

[25] Isci, C. and Martonosi, M. (2003) Runtime power monitoring in high-end processors: Methodology and empirical data. *MICRO 36: Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*, Washington, DC, USA, December 93. IEEE Computer Society.

[26] Li, T. and John, L. K. (2003) Run-time modeling and estimation of operating system power consumption. *SIGMETRICS Perform. Eval. Rev.*, **31**, 160–171.

[27] Bircher, W. and John, L. (2007) Complete system power estimation: A trickle-down approach based on performance events. *Performance Analysis of Systems and Software, IEEE International Symmposium on*, **0**, 158–168.

[28] Bertran, R., Gonzàlez, M., Martorell, X., Navarro, N., and Ayguadé, E. (2011) Local memory design space exploration for high-performance computing. *The Computer Journal*, **54**, 786–799.

[29] Snowdon, D. C., Petters, S. M., and Heiser, G. (2007) Accurate on-line prediction of processor and memoryenergy usage under voltage scaling. *Proceedings of the 7th ACM & IEEE international conference on Embedded software*, New York, NY, USA, Sept./Oct. EMSOFT '07, pp. 84–93. ACM.

[30] Pusukuri, K. K., Vengerov, D., and Fedorova, A. (2009) A Methodology for Developing Simple and Robust Power Models Using Performance Monitoring Events. *Workshop on the Interaction between Operating Systems and Computer Architecture*, Austin, TX, USA, June, pp. 1–10. -.

[31] Jiménez, V., Cazorla, F. J., Gioiosa, R., Valero, M., Boneti, C., Kursun, E., Cher, C.-Y., Isci, C., Buyuktosunoglu, A., and Bose, P. (2010) Power and thermal characterization of power6 system. *Proceedings of the 19th international conference on Parallel architectures and compilation techniques*, New York, NY, USA, September PACT '10, pp. 7–18. ACM.

[32] George, V., Jahagirdar, S., Tong, C., Smits, K., Damaraju, S., Siers, S., Naydenov, V., Khondker, T., Sarkar, S., and Singh, P. (2007) Penryn: 45-nm next generation Intel® Core™ 2 processor. *ASSCC'07 IEEE Asian Solid-State Circuits Conference*.

[33] Sinharoy, B., Kalla, R., Starke, W. J., Le, H. Q., Cargnoni, R., Van Norstrand, J. A., Ronchetti, B. J., Stuecheli, J., Leenstra, J., Guthrie, G. L., Nguyen, D. Q., Blaner, B., Marino, C. F., Retter, E., and Williams, P. (2011) IBM POWER7 multicore server processor. *IBM Journal of Research and Development*, **55**, 1:1 –1:29.

[34] McIntosh-Smith, S., Wilson, T., Ibarra, A. ., Crisp, J., and Sessions, R. B. (2012) Benchmarking energy efficiency, power costs and carbon emissions on heterogeneous systems. *The Computer Journal*, **55**, 192–205.

[35] Floyd, M., Ware, M., Rajamani, K., Gloekler, T., Brock, B., Bose, P., Buyuktosunoglu, A., Rubio, J. C., Schubert, B., Spruth, B., Tierno, J. A., and Pesantez, L. (2011) Adaptive energy-management features of the IBM POWER7 chip. *IBM Journal of Research and Development*, **55**, 8:1 –8:18.

[36] Contreras, G. and Martonosi, M. (2005) Power prediction for Intel XScale® processors using performance monitoring unit events. *ISLPED '05: Proceedings of the 2005 international symposium on Low power electronics and design*, New York, NY, USA, August, pp. 221–226. ACM.

[37] Intel (2009). Intel Core2 Duo Mobile Processor, Intel Core2 Solo Mobile Processor and Intel Core2 Extreme Mobile Processor on 45-nm Process. For platforms based on Mobile Intel 4 Series Express Chipset Family. Datasheet.

[38] ACPI Specification. [Online] Available: `http://www.acpi.info/`.

[39] SBSIF. SMART specifiaction rev.1.1 Dec 11, 1998. [Online] Available: `http://sbs-forum.org`.

[40] Watts Up? [Online] Available: `https://www.wattsupmeters.com`.

[41] PertCtr, the Linux/x86 performance monitoring counters driver. [Online] Available: `http://user.it.uu.se/~mikpe/linux/perfctr/2.6/`.

[42] Thinkpad SMAPI kernel module version 0.40 . [Online] Available: `http://tpctl.sourceforge.net/`.

[43] Perfmon2. [Online] Available: `http://perfmon2.sourceforge.net/`.

[44] Henning, J. L. (2006) Spec cpu2006 benchmark descriptions. *SIGARCH Comput. Archit. News*, **34**, 1–17.

[45] NAS Parallel Benchmarks v2.3. [Online] `http://www.hpcs.cs.tsukuba.ac.jp/`.

[46] Staelin, C. and packard Laboratories, H. (1996) lmbench: Portable tools for performance analysis. *In USENIX Annual Technical Conference*, pp. 279–294.

[47] R Development Core Team (2005) *R: A language and environment for statistical computing*. R Foundation for Statistical Computing. Vienna, Austria. ISBN 3-900051-07-0.

[48] Hsieh, M.-y., Riesen, R., Thompson, K., Song, W., and Rodrigues, A. (2012) Sst: A scalable parallel framework for architecture-level performance, power, area and thermal simulation. *The Computer Journal*, **55**, 181–191.

[49] Brooks, D., Tiwari, V., and Martonosi, M. (2000) Wattch: a framework for architectural-level power analysis and optimizations. *SIGARCH Comput. Archit. News*, **28**, 83–94.

[50] Powell, M., Biswas, A., Emer, J., Mukherjee, S., Sheikh, B., and Yardi, S. (2009) Camp: A technique to estimate per-structure power at run-time using a few simple parameters. *HPCA '09*, Feb., pp. 289–300.

[51] Do, T. V. (2011) Comparison of allocation schemes for virtual machines in energy-aware server farms. *The Computer Journal*, **54**, 1790–1797.

[52] Gelenbe, E. and Morfopoulou, C. (2011) A framework for energy-aware routing in packet networks. *The Computer Journal*, **54**, 850–859.

[53] Lent, R. (2010) Simulating the power consumption of computer networks. *Computer Aided Modeling, Analysis and Design of Communication Links and*

Networks (CAMAD), 2010 15th IEEE International Workshop on, dec., pp. 96 –100.