Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs

Matthew J. Walker, Stephan Diestelhorst, Andreas Hansson, Anup K. Das, Sheng Yang, Bashir M. Al-Hashimi, and Geoff V. Merrett

***Abstract*—Modern mobile and embedded devices are required to be increasingly energy-efficient while running more sophisticated tasks, causing the CPU design to become more complex and employ more energy-saving techniques. This has created a greater need for fast and accurate power estimation frameworks for both run-time CPU energy management and design-space exploration. We present a statistically rigorous and novel methodology for building accurate run-time power models using Performance Monitoring Counters (PMCs) for mobile and embedded devices, and demonstrate how our models make more efficient use of limited training data and better adapt to unseen scenarios by uniquely considering stability. Our robust model formulation reduces multicollinearity, allows separation of static and dynamic power, and allows a 100× reduction in experiment time while sacrificing only 0.6% accuracy. We present a statistically detailed evaluation of our model, highlighting and addressing the problem of heteroscedasticity in power modeling. We present software implementing our methodology and build power models for ARM Cortex-A7 and Cortex-A15 CPUs, with 3.8% and 2.8% average error, respectively. We model the behavior of the non-ideal CPU voltage regulator under dynamic CPU activity to improve modeling accuracy by up to 5.5% in situations where the voltage cannot be measured. To address the lack of research utilizing PMC data from real mobile devices, we also present our data acquisition method and experimental platform software. We support this work with online resources including software tools, documentation, raw data and further results.**

***Index Terms*—power modeling and estimation, embedded systems, performance monitoring counters, PMC event selection**

I. Introduction

THE last 10 years has seen a significant shift in emphasis from desktop and laptop computers towards mobile devices such as smartphones, tablets, thinner fan-less laptops, and, more recently, smart watches [1]. This has promoted energy-efficiency ahead of raw performance as the main design goal in modern CPUs [2]. Increasing the energy efficiency of mobile CPUs not only allows the battery to last longer, but allows these smaller, compact devices to perform more complex tasks while remaining within their thermal design power (TDP), enabling new and innovate applications [3].

This work was supported by ARM Ltd., EPSRC and the PRiME Project

M.J. Walker, A.K. Das, S. Yang, B.M Al-Hashimi and G.V. Merrett are with the Department of Electronics and Computer Science, University of Southampton, UK (e-mail: mw9g09@ecs.soton.ac.uk).

S. Diestelhorst and A. Hansson are with ARM Research, ARM Ltd.

Experimental data and presented software tools can be downloaded at http://www.powmon.ecs.soton.ac.uk/powermodeling/.

Experimental data can also be found at DOI:10.5258/SOTON/393673 (http://dx.doi.org/10.5258/SOTON/393673)

Fundamental to improving energy efficiency is managing the operation of the CPU in an intelligent way. Modern CPUs employ power-saving techniques such as dynamic voltage frequency scaling (DVFS), power gating and multiple asymmetric cores (e.g. ARM big.LITTLE™ technology [4]). It has been shown that run-time management software (or the *run-time manager*) can make significant energy improvements by controlling these power-saving techniques to smartly manage the energy-performance trade-off while taking external factors into account [5], [6]. However, run-time knowledge of the power consumption of each CPU core in the system is paramount in finding the optimum power/performance trade-off.

We present a novel methodology and corresponding software tools for both characterizing the power consumption of real mobile CPUs and producing accurate and stable run-time power models. These models can be inserted into the operating system to provide accurate, per-core, run-time CPU power estimations to the run-time management software. They can also be used as accurate and trusted reference models in design-space exploration in conjunction with a performance simulator, such as *gem5* [7]. We focus on mobile devices because their energy efficiency is of particular importance and modeling their power consumption is challenging due to the dynamic nature of their workloads. However, our proposed modeling methodology is generic, and can be used with other CPUs of different ISAs (instruction set architectures) or in desktop or server systems. The resulting models themselves are specific to a particular CPU implementation.

Our methodology employs statistical rigor throughout each stage of the modeling process, in which we introduce novel techniques and insights and demonstrate how they improve the model quality. We illustrate our approach on a device containing an ARM mobile CPU that is also found in the Samsung Galaxy S5 smartphone (released in 2014). It utilizes ARM's big.LITTLE™ technology, having two CPU clusters of significantly differing microarchitectures: one optimized for greater energy-efficiency (quad-core Cortex-A7), and one optimized for higher performance (quad-core Cortex-A15). We present models of both clusters, focusing on the latter to illustrate our methodology throughout this paper.

In Section II we present a background on CPU power modeling and review related works. In Section III we give an overview of the three main steps of our modeling methodology with a brief description of our key contributions in each of them. Sections IV, V and VI describe these three steps in detail: data acquisition, Performance Monitoring Counter

(PMC) event selection, and model formulation and validation, respectively. We then directly compare our model to related work in Section VII. Section VIII highlights how the voltage supplied to the CPU is dependent on the dynamic CPU activity and introduces a voltage model to significantly improve the power model accuracy in situations where the voltage cannot be directly measured. The key contributions of this paper are:

- An automated PMC event selection methodology that uniquely considers model stability and a demonstration of its importance (Section V);
- A robust model formulation technique that breaks down static and dynamic power and reduces both multi-collinearity and experiment time (Section VI);
- A voltage model that compensates for CPU voltage deviations induced by CPU activity and a demonstration of how it improves the power model accuracy (Section VIII);
- Two accurate and extensively validated run-time PMC power models for an ARM big.LITTLE mobile platform;
- Software tools for both collecting PMC, power and voltage data (platform specific), and implementing our novel modeling methodology (platform independent), available from [8].

II. Related Work

Power models for CPUs can be split into two key groups: *top-down* approaches, where existing devices are characterized experimentally; and *bottom-up* approaches which use theoretical knowledge of each component.

Bottom-up approaches are commonly employed in design-space exploration as they can be adapted to different design specifications. An example of such a tool is *McPAT* [9], a power, area and timing modeling framework. While this flexibility is required for some research in system design-space exploration, it comes at the cost of accuracy. In [10] significant sources of errors are found in McPAT which are largely caused by abstraction error. Rethinagiri *et al.* [11] show McPAT to have average power errors of over 20% for most of the tested workloads when comparing McPAT to a physical ARM Cortex-A9 device. Bottom-up power models are generally unsuitable for run-time management applications due to their relatively poor accuracy and large computational complexity.

Top-down CPU power models, which are built by characterizing existing hardware, are more accurate but inflexible. This lack of flexibility renders them unsuitable for some investigations, but their trusted accuracy makes them valuable for many others. Their low computational complexity brings speed benefits to simulations; Isci and Martonosi [12] report that simple run-time top-down models are vital for studies requiring long executing times, such as thermal analysis. There have been some attempts to combine both techniques together, for example, Lee *et al.* [13] calibrate McPAT using measured values from real hardware in order to improve its accuracy, however, only the overall power is calibrated and the errors of individual components are unknown.

PMC based models have long been used for top-down CPU power estimation due to their high level of accuracy and low overhead [12], [14]–[23]. PMCs count hardware events related to the CPU, such as instructions architecturally executed or L2 cache misses, to give fast run-time information for analyzing the CPU's performance. This detailed insight into the instantaneous operation of the processor allows power predictions to be made. However, despite energy-efficiency being crucial to mobile devices, there are relatively few works that provide accurate run-time PMC-models for them using real data.

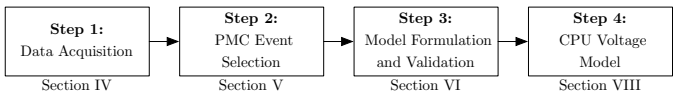


Fig. 1. Steps of our proposed power modeling methodology

To the best of our knowledge, [24] and [25] are the only works that use both PMC data and measured power data from a *real device* to build a run-time power model for an ARM-based mobile CPU. This is due to the lack of a known method of extracting PMCs on mobile platforms, where there are more technical challenges in doing so. Rethinagiri *et al.* [11] also highlight the scarcity of fast power modeling work for mobile systems and present a tool for system-level power estimation created by using a performance simulator, to provide simulated PMC data, with real measured power data from a development board. To overcome the difficulties in obtaining PMC data, simple CPU utilization has been used to make energy-aware run-time scheduling decisions in an ARM big.LITTLE architecture [26]. However, using utilization results in poor accuracy as it provides no information on the *type* of workload [24]. Our method of obtaining PMCs on a mobile platform and corresponding software tools address this problem, aiding future research.

While we have highlighted many works that utilize PMC data to build run-time power models, we introduce several important steps and considerations into our methodology and illustrate their importance on the resulting model e.g. model stability, homoscedasticity and robust model formulation. We also show how the reported model statistics, e.g. average error, can be misleading.

III. Proposed Methodology

This section gives an overview of the steps in our modeling methodology (Fig. 1) and highlights our contributions in each. Step 1 (described in Section IV) concerns the experimental setup and method for acquiring the data used to build and validate the models. We collect PMCs, CPU power and CPU voltage from an ARM big.LITTLE™ based development platform while simultaneously exercising the CPU with a large number of workloads, including ones that utilize parallel programming and the NEON SIMD (single instruction, multiple data) processing unit. To address the lack of existing works using PMC data from real mobile platforms, we have made our software tools available to aid and encourage future research. The quality of the resulting models inherently depends on the quality of data used to build them and we also demonstrate the low overhead and precision of our experimental platform.

Only a limited number of the many selectable PMC events (e.g. L2 cache misses, instructions executed) can be simultaneously recorded; in Step 2 (described in Section V) we present our novel PMC selection methodology that uniquely considers the *stability* of the model when choosing model inputs. Furthermore, we demonstrate how it allows a model to better predict a wider range of workloads, even if they are not well covered in the training data. This is a crucial quality for real-world power models and, although it is not considered in previous work, we show that demonstrating stability of a model is perhaps more important than giving an average error value.

Once the PMC events have been selected, the experiment is run with this selection and the results used to build a linear regression model to predict the CPU power. In Step 3 (described in Section VI) we describe our robust model formulation where we use our knowledge of how power is consumed in CPUs, as opposed to adding regression coefficients directly to PMC data as is typical in existing works [11], [20], [21], [24]. Our formulation reduces multicollinearity, separates dynamic and static power consumption, works with any given voltage and frequency, and, when combined with the added model stability, allows the model building experiment duration to be reduced by 100× while trading off only 0.6% error. In this stage we also thoroughly validate our models and provide a large set of statistical results, allowing the quality of the model to be properly assessed. We identify the problem of heteroscedasticity in run-time power modeling and describe its effects and how to alleviate it.

Furthermore, we uniquely demonstrate how the dynamic CPU activity affects the voltage being supplied to the CPU by the non-ideal voltage regulator, which in turn affects CPU power consumption (described in Section VIII). In run-time management scenarios, the voltage supplied to the processor cannot be measured and we therefore present a novel voltage model, which takes the current frequency and modeled dynamic power consumption as inputs (Step 4). We demonstrate how this improves the run-time power estimation accuracy by as much as 5.5%.

We provide software tools to aid research related to PMCs, power and temperature on mobile devices using real data (platform dependent) and to implement our novel methodologies described in Steps 2 and 3 (platform independent) at [8]; raw data, additional graphs and results, model coefficients, and software documentation are also provided.

IV. Data Acquisition

As highlighted in Section II, there is very little reported research that uses real PMC data from a mobile ARM-based platform due to the lack of an established method of doing so. In this section we present our experimental setup, method and accompanying software tools for collecting PMC data, running workloads and measuring CPU voltage and power on a mobile platform (Step 1 of our power modeling methodology). We also demonstrate its low overhead and high precision, which is essential for producing accurate models.

We use an ODROID-XU3 development board by Hardkernel to illustrate our approach. It utilizes a Samsung Exynos 5422 SoC (System on Chip) which has an ARM big.LITTLE™ design containing two types of CPU core: four ARM Cortex-A15s optimized for high performance and four Cortex-A7s optimized for energy-efficiency. Each core contains a NEON SIMD processing unit, which we account for in our model.

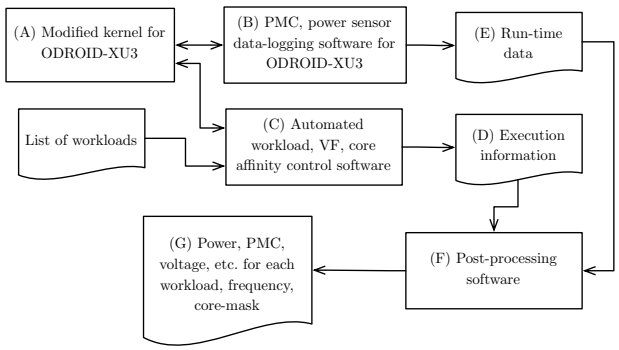
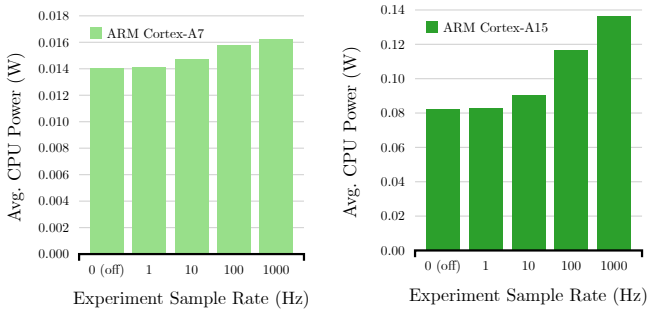


Fig. 2. Simplified overview of our experimental platform software (corresponding to Step 1 in Fig. 1).

We develop, and make available, software tools to streamline future research on mobile development boards in the form of a customized Linux operating system image (Ubuntu 14.01.1) and programs for running experiments and capturing data from PMCs, operating system statistics and the built-in energy sensors on the ODROID-XU3 (Fig. 2).

We modify the kernel of the operating system (Item A in Fig. 2) to include the *userspace* frequency governor so that the frequency and voltage can be changed from userspace. We develop a loadable kernel module (LKM) to allow userspace access to the PMCs by setting the *PMUSERENR* register on each CPU core. We write data-logging software in C and inline assembly to access and record the PMCs, power sensor data and operating system statistics (Item B) with a low overhead (Fig. 3). There is no perceivable power overhead when running the experiment with a sample frequency of 1 Hz and a very small overhead when running the experiment at 10 Hz. The overhead does not contribute to errors because the PMC data measures all of the CPU activity. However, for high-quality experimental data, the effect of the experimental software should be minimized so it does not interfere with the workloads under test. We use a sample rate of 5 Hz and run the workloads for an extended period of time to gather a large number of data points to properly account for workload phases. To test the consistency of the experimental platform, we run our experiment 11 times and observe the deviation. The average standard deviation of the measured power (which deviates more than the PMC events) over all 60 workloads, whose average power ranges from 0.23 W to 1.8 W, is 0.0049 W. This indicates a high level of precision and repeatability in our experimental setup which is necessary for building high-quality models.

As well as recording data, the experimental software runs a specified set of workloads (Item C, Fig. 2) over: 1) any specified list of voltage-frequency points (Cortex-A7 frequencies and Cortex-A15 frequencies can be independently specified,



(a) Cortex-A7 (b) Cortex-A15

Fig. 3. Power overhead of experimental platform (@200 MHz, worst-case)

allowing many combinations); 2) any specified set of core masks (control of CPU affinity on both 'big' and 'little' cores); and, 3) any specified number of simultaneous executions of the workload (for exercising multiple cores simultaneously). Having the ability to automatically collect data for many combinations allows the model to be built with a wide range of scenarios.

We use 60 workloads from a variety of sources, including: 14 from the MiBench [27] suite; 20 from LMbench [28]; 11 from Roy Longbottom [29]; 1 from MediaBench [30]; 5 handwritten workloads; and other workloads that make use of programs such as *MPlayer*, *tar* and *gcc*. This set covers a large range of both realistic and synthetic workloads, including ones that are CPU intensive (e.g. *bitcount*, *neon_mul*, *jpeg_dec*, *dhrystone*), memory intensive (*mp_lp_neon*, *lat_mem_rd_1_256*, *bw_mem_rd*, *cache*) and I/O intensive (*bw_mem_cp_700m*, *par_mem*, *openmp_mem_spd*). We tune some synthetic workloads to trigger certain CPU behavior (e.g. accessing particular levels of cache) and use PMC data to ensure the workloads achieve the desired effects. For example, we use *lat_pagefault* and *mp_lp_neon* to cause a large number of TLB (translation lookaside buffer) misses, *cstm_int* to cause a large number of mispredicted branch instructions, and *neon_add*, *cstm_fp* and *mp_neon_mflops* to cause a large number of unaligned accesses. In Section V we show how using a diverse subset of these workloads results in a more robust model than when using a more typical set (i.e. MiBench and MediaBench). In Section VI we show how the components of the dynamic power prediction varies significantly between different workloads (Fig 11).

To keep the overhead of the experiment to a minimum, the data from both Items B and C (Fig. 2) are captured in a raw format and then combined and processed (Item F) after the data has been recorded. The result is a table of workloads (run on different numbers of cores and at different DVFS levels) against the corresponding CPU power, voltage and PMC data, all averaged over the duration of each workload (Item G).

V. PMC Event Selection

While there are many PMC events, only a few of them can be monitored simultaneously on mobile CPUs. On the ARM Cortex-A7 and Cortex-A15 only four and six PMC events (in addition to the Cycle Count) can be monitored simultaneously, respectively. The decision of which PMC events to use as inputs to our power model is key, as we will demonstrate. This section presents our novel methodology for choosing PMC events (Step 2, Fig. 1) which results in an accurate and *stable* model. Furthermore, we experimentally demonstrate the importance of stability, give insight into how many PMC events should be used, and show how the common practice of using a limited number of workloads for training and validation results in a poor power model with an optimistic reported error.

In order to analyze and compare all of the available PMC events, we set our experimental framework to keep the frequency and core-mask constant while running all of the workloads multiple times and change the recorded PMC events on each iteration to cover almost all of them; we collected data for 39 and 66 events on the Cortex-A7 and Cortex-A15, respectively. The large set of PMC events are then cross-compared and analyzed in order to find an optimum selection to use as model inputs.

An important consideration when choosing the events is multicollinearity, which occurs when two or more independent variables in the linear regression model have inter-correlation (a relationship) between them. While this does not necessarily have a *direct* impact on the reported accuracy or overall fit, it causes errors in the model coefficients and makes the model overly sensitive to changes in the inputs, resulting in an unstable model. If a model with high multicollinearity comes across a scenario that is not well covered in the training data, then it likely produces inaccurate predictions, as we demonstrate experimentally later in this section. It is impractical to cover all of the possible behaviors that real-world workloads may exhibit in the training workloads used to build the model. Therefore, a stable model that is better able to predict scenarios outside of the training set is a vital property of a practical, real-world power model.

To quantify multicollinearity (and therefore give an indication of the stability) we calculate the VIF (variance inflation factor) for each PMC event. To find the VIF for a particular independent variable, we build an ordinary least squares (OLS) linear regression model which predicts that variable using the others. We use the resulting R^2 value (indicating *goodness-of-fit*) from the model to calculate the VIF:

$$\text{VIF} = \frac{1}{1 - R^2} \tag{1}$$

A VIF of one, for example, indicates that there is no correlation between that independent variable and the others. If an independent variable has a VIF of 10, for example, it would indicate that the variance of that predictor coefficient is $10\times$ larger (and the standard error of that predictor coefficient is therefore $3.2\times$ [square root of 10] larger) than if there was no multicollinearity present. It is widely considered that, as a general rule of thumb, a VIF over five or ten [31], [32] indicates that there are strong multicollinearity problems.

A set of PMC events should therefore be carefully chosen to provide the model with the largest possible amount of *unique* information useful for predicting the power, without providing

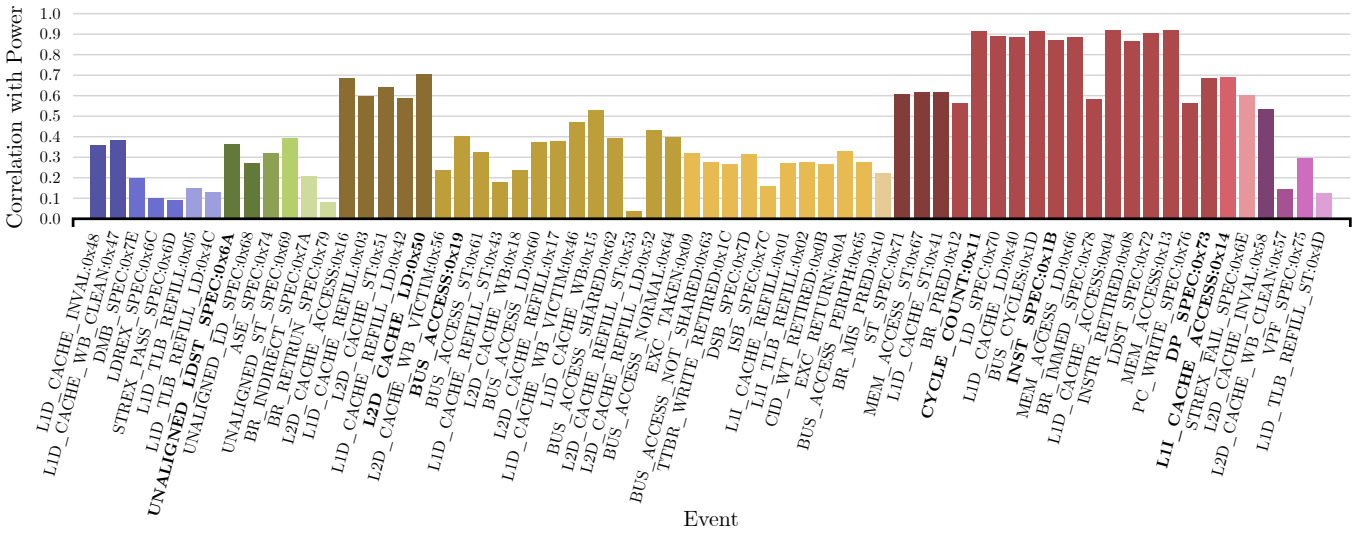


Fig. 4. Correlation of each Cortex-A15 PMC event with power, grouped and colored by cluster, with PMC events chosen by our automated algorithm labeled in bold font

duplicated information which results in multicollinearity; this is key to building an accurate and stable model. Choosing PMC events solely on their correlation with the overall CPU power, for example, results in a poor model because PMC events that correlate well with power also correlate well with each other, starving the model from valuable information contained in other events and causing multicollinearity problems.

We employ several statistical analysis techniques to develop a deep understanding of how to select the optimum events in order to devise a simple automated method of doing so. We use hierarchical cluster analysis (HCA) and inspection of the corresponding dendrograms to group similar PMC events together into clusters (based on how they correlate with each other). Choosing events in different clusters therefore results in diverse information being provided to the model about the current operation of the CPU. However, the information does not necessarily help to predict the power consumption. We used the Pearson product-moment correlation coefficient to calculate the linear correlation of each PMC event with the CPU power consumption and combined the results with the HCA results (Fig. 4). PMC events with a very high correlation with power (>0.75) are all in the same cluster, highlighting how choosing PMC events on correlation alone results in similar events, which means the model will have intercorrelated inputs that only provide it with a narrow range of information. A good PMC selection is made by choosing PMC events that have a high correlation with power but avoiding repeatedly selecting from the same cluster. However, a decision of how many clusters to group the PMC events into and how much to prioritize clustering or correlation has to be made, requiring intuition and further experimentation.

We build on the knowledge from the HCA and the VIF itself to develop a simple, two-stage, automated method for selecting optimum PMC events that provide the model with the largest possible amount of information for predicting power with minimum multicollinearity. The first stage uses regression analysis to select PMC events, one-by-one, that add the largest

```
1:  procedure SELECT_EVENTS(allEvents, no.Events)
2:      selectedList ← cycleCountEvent
3:      while length(selectedList) < no.Events do
4:          for pmcEvent in allEvents do
5:              build_model(selectedList + pmcEvent)
6:              if newR^2 > bestR^2 then
7:                  bestEvent ← pmcEvent
8:                  bestR^2 ← newR^2
9:              end if
10:         end for
11:         append bestEvent to selectedList
12:     end while
13:     return selectedList
14: end procedure
```

Fig. 5. Algorithm of the first stage of our PMC event selection method

insight into power consumption *given* the previously selected PMC events (Fig. 5). Our analysis shows that the Cycle Count (*0x11*) should always be included in the selected set of PMCs as it contains unique information that is useful for predicting the power. Therefore, by adding the Cycle Count to the list of selected PMCs first (line 2, Fig. 5) we find a superior set of PMC events because the remainder of the algorithm finds the optimum events *given* the information in the Cycle Count. Our algorithm then finds the next best PMC event to add to the set of model inputs by building a regression model to predict power for each additional PMC event with the existing selected PMC events also as inputs. The PMC event that results in the most improved R^2 value is then added to the selection. For the Cortex-A15 example, our algorithm chooses the seven PMC events (highlighted in bold in Fig. 4) from five different clusters and the chosen events do not necessarily have a high correlation with power, with one chosen event having a correlation as low as 0.36 (UNALIGNED_LDST_SPEC:0X6A).

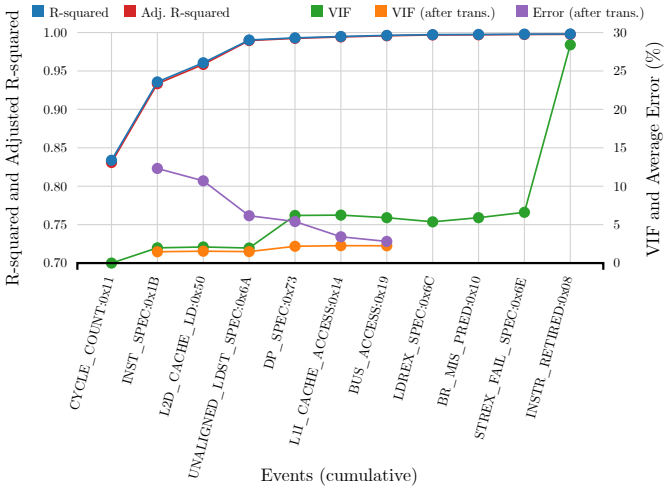


Fig. 6. R^2, Adj. R^2, VIF of selected PMC events (cumulative) from Stage 1. Also shows error and VIF after transformation in Stage 2

TABLE I
FIRST SEVEN SELECTED EVENTS FOR THE CORTEX-A15 WITH THEIR CORRESPONDING CLUSTER NUMBER AND VIF FROM STAGE 1

	Event Hex	Event Name	Cluster	VIF
1	0x11	CYCLE_COUNT	13	2.12
2	0x1B	INST_SPEC	13	17.5
3	0x50	L2D_CACHE_LD	8	1.87
4	0x6A	UNALIGNED_LDST_SPEC	4	1.88
5	0x73	DP_SPEC	13	13.3
6	0x14	L1I_CACHE_ACCESS	14	2.23
7	0x19	BUS_ACCESS	9	1.50

Fig. 6 shows how the R^2 and the Adjusted R^2 (adjusted for the number of predictors in the model) increase as each selected PMC event is added to the model (one-by-one), while the VIF (indicating the presence of multicollinearity) also increases. After choosing the fifth event (*0x73*) the VIF suddenly increases to a value over five (indicating multicollinearity problems) while the R^2 and Adjusted R^2 only marginally increase, suggesting that four PMC events (including the Cycle Count) is an optimum number to choose. However, by not utilizing all seven available counters the model is not making use of all of the available information. Furthermore, from building the model (Section VI) with different numbers of PMC events, we found that, despite only a marginal increase in R^2, there was a significant decrease in average error between using four and seven counters (see the *Error after trans.* line in Fig. 6). To utilize the full seven available PMC events, the multicollinearity must be further reduced.

The second stage of our PMC event selection method takes the result of the first stage and further reduces multicollinearity allowing the model to use as many events as possible (therefore obtaining as much information as possible, improving accuracy) while maintaining stability. Table I shows the chosen PMC events from stage 1 for the cortex-A15 model presented in this paper, along with the VIF and cluster number for each one, which was developed using the full set of 60 workloads.

Our method first identifies relationships between specific PMC events that contribute significantly to the multicollinearity within our chosen set of PMC events using the VIF. As each PMC event is added to the model, the collinearity between that event and the existing events can be understood by monitoring the change in the individual VIFs for each PMC event. In the example of the Cortex-A15 model, Fig. 6 shows that the VIF rises significantly when event *0x73* (DP_SPEC) is included in the model. By looking at Table I (which shows the VIF of each individual PMC event when all seven events are included in the model together) it can be seen that event *0x1B* and *0x73* both have particularly high VIFs. This indicates that the increase of the average VIF among all of the selected events is caused by collinearity between these two events.

Once we identify strong collinearity between specific events, we make a transformation to remove the repeated information. In the case of our Cortex-A15 example, PMC event *0x1B* counts *all* instructions speculatively executed and PMC event *0x73* counts the *integer* instructions speculatively executed, which means that event *0x73* is also counted within event *0x1B*. Both events are required; one of the events cannot be derived without the other, but the repeated information leads to multicollinearity. We therefore transform event *0x1B* into *0x1B-0x73*; no information has been lost but the duplicated information has been removed. The *VIF (after trans.)* line shows how the VIF remains low (well below five) after making this transformation, indicating that the multicollinearity has been significantly reduced.

Reducing the multicollinearity in this second stage has made it possible to utilize information from all six PMCs (and the cycle counter), almost halving the average error from over 5% to less than 3%, without sacrificing stability. Being able to monitor only six PMC events simultaneously is not a significant limitation; as the number of events added to the model increases, the improvement in accuracy decreases (Fig. 6). Our generic methodology allows an appropriate number of PMC events for a particular CPU to be found by measuring the trade-off between VIF and average error.

To demonstrate the importance of stability and carefully selecting PMC events, we built a model using a different set of PMC events that do not consider variance inflation (CYCLE_COUNT:0x11, L1I_TLB_REFILL:0x02, MEM_ACCESS:0x13, L1D_CACHE_ACCESS:0x04, INSTR_RETIRED:0x08, ASE_SPEC:0x74, VPF_SPEC:0x75). These choices reflect similar counters to those proposed in works on desktop and server systems and appear to be reasonable, intuitive choices [11], [21]. However, the average VIF of these events is 1.68×10^7 and the coefficients change significantly when building the model with different sets of workloads because they are unstable (note that the last two events do not contribute to the multicollinearity problems). Fig. 7 compares a model built with these unstable PMC events and an otherwise identical model built using the results from our proposed PMC event selection method. The errors of these two models have been calculated over a variety of training and validation workload sets, all of which use k-fold cross-validation with $k = 10$ (the building and validation

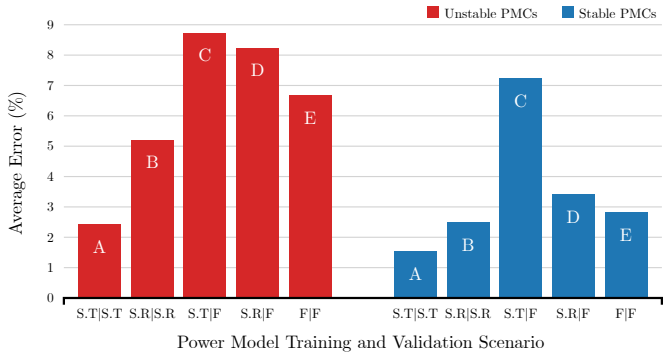


Fig. 7. Comparison of a model with unstable PMC events and one with stable PMC events selected with our proposed methodology. Tested with different training and validation data sets. (S.T = small typical, S.R = small random, F = full set of 60 workloads)

process is described in detail in Section VI).

With a limited set of 20 'typical' (MiBench and MediaBench) training and testing workloads (a typical scenario in related work [11], [20], [33]) both models obtain a small error (*A* bars). However, when 40 unseen training workloads are added to the validation workloads, the error of both models increases significantly, from 1.5% to 7.2% in the case of our 'stable' selection (blue C bar). The small set of relatively similar workloads does not provide either model with enough information for predicting the full (and diverse) set of 60 workloads. This demonstrates how using a small, limited set of workloads for both training and validation results in a poor model which actually *appears* to be very accurate.

With a limited set of 20 'random' (a random selection from our diverse set of 60 workloads) training and testing workloads, the model with an unstable (red model) PMC event selection performs poorly compared to the model with our stable PMC event selection (blue model) due to the diversity in testing workloads (*B* bars). Furthermore, when 40 unseen workloads are added to the training set, the red model has a large error of 8.7% while our blue model has a small error of just 3.4% (*D* bars), which is only 0.6% less than when it is validated on the full set of 60 workloads (blue *E* bar).

To summarize, there are three key points from Fig. 7: 1) training and validating with a small set of workloads results in a poor model but with a very optimistic average error value; 2) training with a limited set of typical and similar workloads results in the lowest error when validated on the same set of workloads, but the highest error when validated on a large number of workloads (e.g. compared to a limited set of diverse workloads); 3) our proposed stable PMC selection is far superior at predicting a large number of diverse workloads outside of the training dataset and it makes efficient use of a limited, but diverse, training set. This benefit of stability comes from the fact that the model coefficients have lower errors and so the model better captures how each PMC event affects the overall power *individually*. A stable model is therefore able to make sensible estimation in unfamiliar situations and is less prone to wild predictions. For example, when both models are trained with 20 random (diverse) workloads and validated on 60, our blue model has a maximum error of less than 15%, while the red model has a maximum error of over 45%. Stability is a crucial quality of a run-time power model as it is impractical to represent the vast number of real-world applications in a training dataset.

We make available software tools for the PMC event selection process. Our platform dependent data acquisition software can be set to automatically collect the experimental data required for this process; it automatically repeats the experiment while switching between PMC events and combines the data to allow the different PMC events to be directly compared. Our platform independent analysis software implements our automated PMC event selection method and also many of our analysis techniques (including correlation analysis and hierarchical cluster analysis) which is useful for other areas of research regarding PMC events on mobile devices. It has a simple interface, easily allows data from other experimental setups to be used, and clearly presents the results with interactive graphs.

VI. Model Formulation and Validation

This section describes the model building and validating stage which is the third step of our power modeling methodology (Fig. 1). Once we have obtained optimal events from Step 2 (Section V), we use our experimental setup (Section IV) to run our full set of 60 workloads at many DVFS levels, with different numbers of cores being utilized, to extensively investigate as many operating conditions as possible, on both the ARM Cortex-A7 and Cortex-A15.

Rather than simply putting the PMC data directly into a linear regression tool, as is the case in previous works [11], [20], [21], [24], we combine regression analysis with knowledge of how power is consumed within CPUs to form an intelligent and more physically-meaningful model that calculates static and dynamic power separately (2). We make the CPU cluster power our dependent variable with functions of our chosen PMC events after preprocessing (E_n), clock frequency (f_{clk}) and CPU voltage (V_{DD}) as our independent variables. We then perform multiple linear regression, using an ordinary least squares (OLS) estimator [31], to calculate the coefficients (β_n) of our model. This not only allows the model to estimate power at any given voltage or clock frequency, but also further reduces multicollinearity and therefore improves the model's stability. CMOS dynamic power is proportional to $V_{DD}^2 f_{clk}$ and CMOS static power is a product of V_{DD} and the leakage current, which is predominantly formed of the sub-threshold leakage and gate oxide leakage [34]. There is a constant dynamic power component independent of the PMC events (BG dynamic). The static power component also absorbs the effect of varying temperature depending on the DVFS level, a more detailed analysis of which is the topic of future work. We experimentally found terms that were able to accurately estimate the static power consumption across all DVFS levels on both the Cortex-A7 and Cortex-A15 (without overfitting and with all inputs statistically highly significant, as described later in this section). We collect the PMC events as counts-per-second, and their values are therefore related to the operating frequency of the cluster (made up of four Cortex-A7 or Cortex-A15 CPUs). To reduce multicollinearity, we then

TABLE II
MODEL FORMULATION EXPERIMENT SPEEDUP WHEN EXPLOITING SMART MODEL FORMULATION AND ENHANCED STABILITY

	Avg. Error (%)	Experiment Time (hours)	Workloads
Slow	2.8	40	60
Fast	3.4	0.42 (25 min.)	30

divide each event by f_{clk}, thus separating the clock frequency from the event values, and add it as a separate term; the events now give information solely on the *type* of workload.

$$P_{cluster} = \underbrace{\left(\sum_{n=0}^{N-1} \beta_n E_n V_{DD}^2 f_{clk}\right)}_{\text{dynamic activity}} + \underbrace{f(V_{DD}, f_{clk})}_{\text{static and BG dynamic}} \quad (2)$$

In (2), N is the total number of PMC events in the model; n is the index of each event; E is the cluster-wide PMC event rate (events-per-second) after being divided by the operating frequency in MHz, f_{clk}, and averaged across all cores; and V_{DD} is the cluster operating voltage. $P_{cluster}$ is the power for the overall quad-core cluster (Cortex-A7 or Cortex-A15).

Our experimental setup measures the total power of the cluster, constituting of the four cores, their respective L1 cache and the shared L2 cache. The model is therefore formulated to predict the power of the overall cluster as this can be directly validated. However, in our training set we utilize different numbers of cores and our results show that the overall cluster power can be estimated accurately without knowing how much each individual core is individually utilized. The power contribution from each core and its L1 cache (including the effect it has on the shared L2 cache) can therefore be calculated by substituting the cluster-wide events with the fraction for a particular core.

Our model formulation allows us to run the workloads at one single DVFS level and then just collect the idle power characteristics at all of the DVFS levels; as opposed to running the workloads at every frequency. Our enhanced model stability also allows us to build an accurate model using fewer workloads. We demonstrate how combining these two qualities allows the experiment time to be reduced by $96\times$ while only adding 0.6% error (Table II). Not only does this demonstrate the robustness of our model formulation and the benefit of stability, but the reduced experiment time (and reduced amount of resulting data) has an important practical advantage. The remainder of this section discusses our method of evaluating our models.

A fundamental and necessary part of building linear regression models is the inspection of the residuals, which must be done in order to determine whether the model is valid and the assumptions for the linear regression model have been met [35]. Yet, despite this, very few related works (the only exception known to us is [18]) present or discuss the residuals, which need to be checked *before* other model statistics can be trusted and interpreted. Importantly, our residual plots (Fig. 8, more plots shown at [8]) show no pattern that can be predicted from another variable or each other, proving that the residuals (observed errors) only represent the stochastic response of the

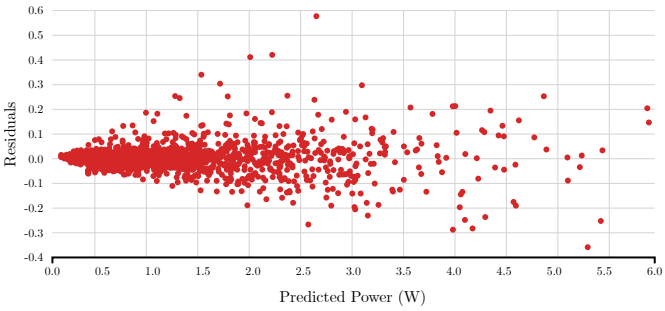


Fig. 8. Plot of the model residuals against the predicted power (other residuals plots found online [8]) before applying HC3

TABLE III
RESULTS FOR THE CORTEX-A7 AND CORTEX-A15 CPU MODELS (FINAL MODEL, TRAINED ON ALL WORKLOADS)

Parameter	A7 Value	A15 Value
R^2	0.993	0.997
Adjusted R^2	0.993	0.997
No. Observations	1680	2160
Std Err. of Regression (SER) [W]	0.0133	0.0517
F-Statistic	28057.2	40168
p-Value for F-Statistic	$p < 0.00001$	$p < 0.00001$
Avg. VIF (PMC events only)	2.13	2.25
Avg. VIF (inc. V and f)	4.94	3.04

model and not the deterministic part, meaning that our model captures all the necessary information from the input variables. However, the 'cone-shape' of the residuals plot shows the presence of heteroscedasticity (meaning that a key assumption of homoscedasticity has been violated), which reduces the accuracy of the coefficients themselves, the standard error of the coefficients, the standard deviation of the forecast errors and the confidence intervals [36], [37]. The problem of heteroscedasticity is inherent in PMC-based power modeling (there is a larger variance in power consumption between high power-consuming workloads than workloads that consume less power), yet, to the extent of our knowledge, this has not been highlighted or addressed in related work.

We address this problem by using a heteroscedasticity-consistent standard error (HCSE) estimator of OLS parameter estimates. While the regression model itself is still estimated using OLS (the accuracy of which is demonstrated later), the standard error calculations no longer assume homoscedasticity. While an explanation of heteroscedasticity and HCSE estimators is well beyond the scope of this paper, Hayes and Cai [38] give a good introduction and highlight the problem of HCSE estimators being largely unknown outside of the fields of statistics and econometrics and how they can eliminate the need for researchers to worry about heteroscedasticity when using OLS regression. The option to use HCSE estimators is available in many statistical computing programs and we use a HC3 estimator [39], [40] in our methodology and available software (the statistical modeling in our software uses the *Statsmodels* Python module [41]).

We publish a large number of statistical results allowing

others to accurately assess the quality of our models and we encourage others to do so to enable comparisons between different models presented in different works. Both our Cortex-A7 and Cortex-A15 models achieve an R^2 value of greater than 0.99 (Table III), showing that the fitted regression explains over 99% of the variation in the power consumption, despite being tested on a large number of diverse workloads. The adjusted R^2 (which compensates for the number of predictors) is less than 0.00006 lower than the R^2 value, showing that the R^2 value is not inflated by the number of predictors (independent variables) and that each predictor adds valuable information to the model. The average VIF of the input PMC events of both models is under 2.3, which is thanks to the PMC event selection method (Step 2, described in Section V), and the average VIF of all of the model inputs (including the voltage and frequency) is less than five, thanks to the model formulation (Step 3, described earlier in this section). These low VIF values indicate that our model is stable, which we also demonstrated in Section V. We also calculate and report statistics for determining the statistical significance in Table III. The p-Value for F-Statistic row shows the p-value when the model is compared to a model with no predictors. We have very small p-values of under 0.00001 for both models; meaning that if the null hypothesis is true (there is no effect or relationship between the model inputs and power), the observed effect would be found in less than 0.001% of the experiments run due to random sampling error. In many scientific studies, $p < 0.05$ is considered borderline statistically significant and $p < 0.001$ is considered borderline statistically highly significant.

Our model equations are used to give predictions for the power consumption given the values of the model inputs. However, despite it not being discussed in most related works, the uncertainty of the model must also be considered when making predictions. Our software allows the prediction intervals to be calculated, which is a range where a new observation is likely to fall given the predictors. While prediction intervals have a confidence level, they should not be confused with confidence intervals, which predict the spread of the mean rather than individual observations. The prediction interval is much larger than the confidence interval and takes into account the variability and uncertainty. The prediction interval depends on the predictors, but an approximation of the 95% prediction interval is given by:

$$P.I.(95\%) = \pm(2 \times \text{SER}) \tag{3}$$

The SER (standard error of regression) gives the average distance between the observed values and the regression line (Table III). Therefore, the approximated 95% prediction interval for the Cortex-A15 model is ± 0.10 W; we are 95% confident that the actual power is within the predicted power ± 0.10 W for the next observation.

We report the model coefficients for the Cortex-A15 model that we have been developing throughout this paper to allow the model to be directly implemented in other work (available at [8]). The coefficients for the transfomed PMC events are all positive and the voltage and frequency both have a positive influence on the power, as expected. We also report the 95% confidence intervals (C.Is) of the coefficients which take into account the sample size and the variance in the population. A narrow confidence interval indicates a low sampling error. Our confidence intervals are very small, showing very low standard error and very high statistical significance for each coefficient. The p-values for every coefficient are very low, far lower than 0.05, confirming the statistical significance of every model coefficient.

TABLE IV
MODEL RESULTS FROM K-FOLD CROSS-VALIDATION

Parameter	A7 Value	A15 Value
No. Folds (k)	10	10
Fold Group Size	168	216
Avg. Err. (MAPE) [%]	3.79	2.81
Mean Sq. Err. (MSE) [W^2]	0.000186	0.00276
Root Mean Sq. Err. (RMSE) [W]	0.00975	0.0613

The results in Table III are derived from all of the observations used to build the model. We also employ k-fold cross validation, which involves randomizing the order of the observations, splitting the observations into k groups, then using $k-1$ of the groups to build the model (training dataset) and the one remaining group to validate the model (testing dataset). We repeat this process so that the model is built k times, with each group of observations being used to validate the model. The reported cross-validated errors (Table IV) are the average of the testing datasets, so the model is always predicting the power for scenarios it has not seen before. The validated average error (mean absolute percentage error [MAPE]) is 3.8% and 2.8% for the Cortex-A7 and Cortex-A15 model, respectively. By looking at the cross-validated average errors for each of the 60 workloads (Fig. 9), it can be seen that *cstm_bmp* (a custom synthetic workload) is the only workload with an average error of over 6.5%. The fact that there is one significant outlier shows the need to use a large number of workloads in power modeling. The error of our model is very low, particularly considering the large variance in power consumption between workloads (Fig. 10, gray bars), i.e. the CPU is not simply being fully utilized by every workload.

Our model formulation allows us to see how the static power (which also includes the background dynamic power) and each PMC event contributes to the overall power (Fig. 10) and it can be seen that the information provided by the PMC events on the *type* of workload is essential to producing an accurate model. Each PMC event makes a significant contribution to the dynamic power prediction, working independently to identify different workload types (Fig. 11), showing the merit of our PMC event selection method (Step 2) and its importance.

In PMC-based models, events are sampled at intervals. However, if the sample period is too low, the CPU frequency is too low, and there are very low levels of activity on the CPU core, then the PMCs may not increment fast enough between samples to give accurate values to the power equation. To observe the point at which this phenomenon occurs, we implement our Cortex-A15 model and run it at various sampling frequencies while the cluster clock frequency is set

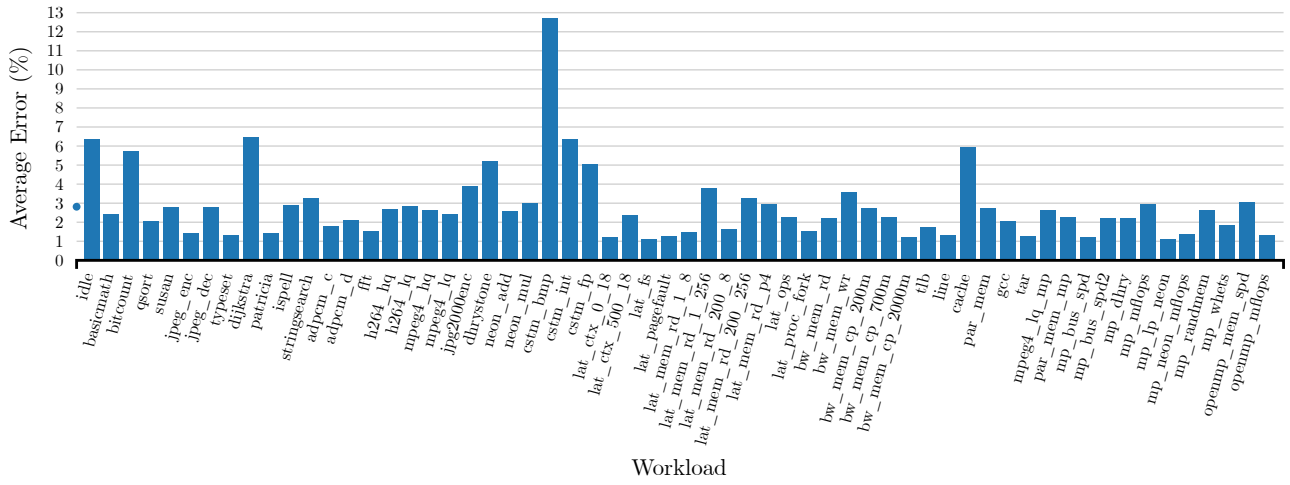


Fig. 9. Average error (mean absolute percentage error [MAPE]) across each DVFS level for all 60 workloads (Cortex-A15 CPU model)

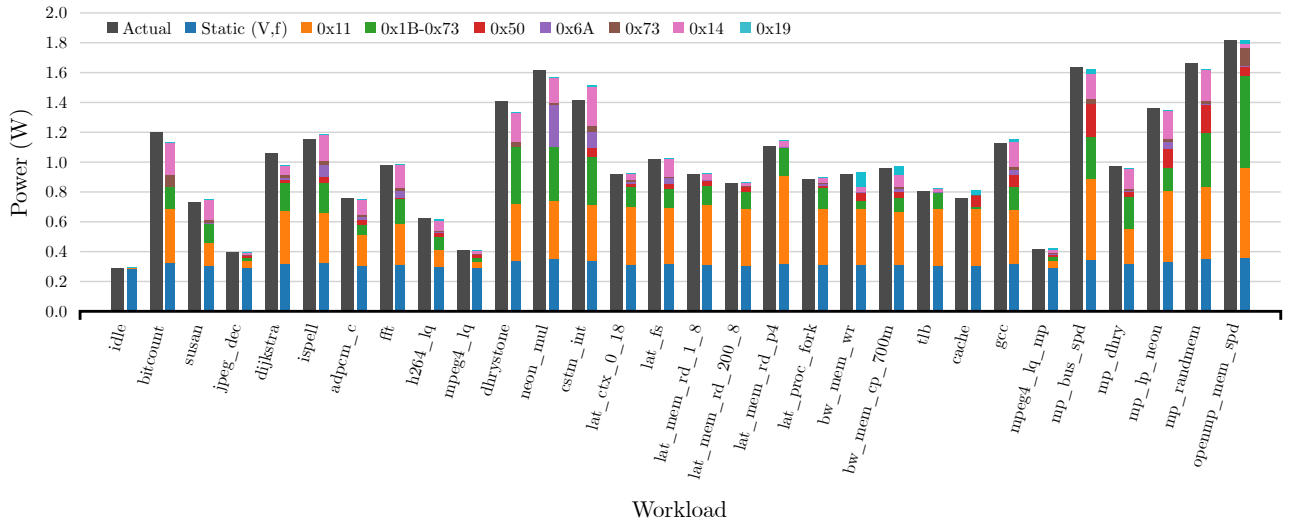


Fig. 10. Actual (measured) power vs. the predicted power for half of the considered workloads, with the predicted power broken down into its constituting parts (Cortex-A15 CPU model)

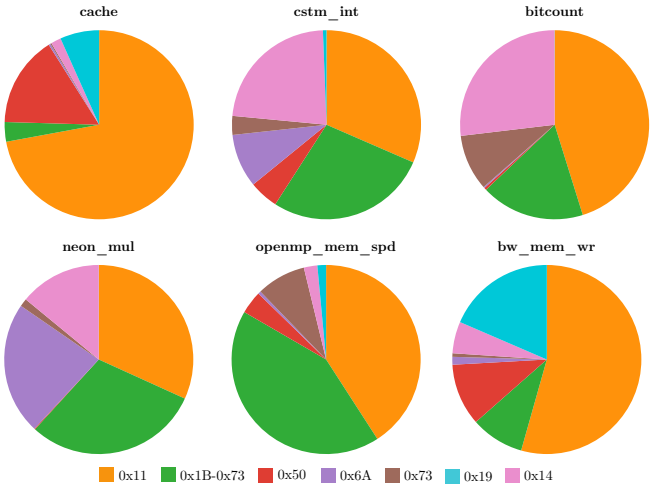


Fig. 11. Contribution of each PMC event to the dynamic power prediction for six different workloads

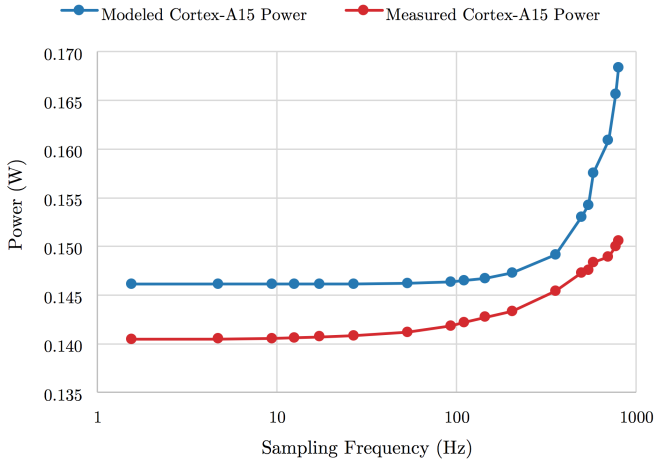


Fig. 12. Modeled power and measured power for the ARM Cortex-A15 cluster running at 200 MHz with various sampling frequencies and no other workloads

to the minimum value with no workloads running, except for the model itself and experiment monitoring software. Even in this worst-case scenario, the model error does not increase until the sampling frequency is beyond 500 Hz (Fig. 12). Many techniques can, however, be employed to reduce this effect, such as adjusting the sampling frequency with the clock frequency, only relying on fast counters when there are low levels of activity on the core, detecting when a particular counter is too slow and sampling that particular counter at a lower rate. Our proposed model formulation and stability allows the contribution of each individual event to be accurately known and if, for example, one PMC event is not occurring regularly enough (event 0x19 in this case), it can be dealt with individually with limited knock-on effects to the other coefficients and therefore the overall power consumption. Another source of error variance in this example is the overhead of writing the results to a file (only required for this experiment); as the sampling frequency increases, the more time the CPU spends running this workload, changing the *type* of workload running. The model accuracy naturally changes with the workload (Fig. 10).

Fig. 12 also shows the worst-case power overhead, which includes extracting and recording data from both the power sensors and the model for evaluation purposes; this is considerable compared to the overhead of the model itself. We present a modeling *methodology* that can be used with a variety of platforms and in many scenarios (both offline and online). When implementing our model, many optimizations and trade-offs can be made, depending on the number of available counters, required accuracy, sample frequency, etc. For example, with a fixed sampling frequency and known set of DVFS levels, then many of the variables in our formula can be pre-computed and the software using out model can simply switch equation when changing DVFS level.

VII. Comparison with Existing Works

In this section we compare our Cortex-A15 model developed using the proposed approach to five models from four recent works in mobile PMC-based run-time power modeling. We implement existing works on our platform and compare them directly. To be consistent, we re-train and implement each model (including our proposed model), using data from the same experiment, and calculate the coefficients using OLS regression. Although our model formula works for any given voltage and frequency, existing works typically consider a single DVFS level or each one separately, and so this comparison considers a single clock frequency of 1 GHz. This simplification of our model causes slight discrepancies between the statistics reported in this section and those reported in previous sections. We model the power for the quad-core Cortex-A15 cluster and the model inputs have been calculated considering the activity of the overall cluster. The process for building and evaluating all of the models is identical, with the only difference being the choice of model inputs and the model equation.

Pricopi *et al.* [25] build a PMC-based power model for an ARM Cortex-A15, considering just a single core of their multi-core cluster (*Model a*, Table V). Walker *et al.* [24] present equations for modeling an ARM Cortex-A8 single-core CPU using four PMC events (*Model b*). We also compare against the dual-core Cortex-A9 model presented by Rethinagiri *et al.* [11], removing the frequency term as we are only considering a single clock frequency (*Model c*). It is not clear in the original work whether the cache miss rate term in their model should take the L1 instruction cache into account, however, doing so would require more than six PMC events in total to implement their model, which is not possible on our platform or the ARM Cortex-A9 (the original work used a simulator to obtain PMC data instead of recording it from a real board directly). Rodrigues *et al.* [21] present several models with varying numbers of PMC events, concluding that the same three PMC events (number of fetched instructions, L1 cache hits and dispatch stalls) can be used to yield an acceptable error ($< 5\%$) across multiple architecture types, including both high performance and low power CPUs. They simulate two cores representative of an Intel Nehalem and an Intel Atom processor using SESC and Wattch. Unfortunately, on ARM-based platforms, there is no PMC event that represents dispatch stalls. We therefore implement their model named *Exp 2* (*Model d*), which does not use dispatch stalls and uses just two model inputs (but requires three PMC events to calculate on our platform), and *Exp 6* (*Model e*) which uses five model inputs events, but we omit the unavailable dispatch stalls event, meaning our implementation of this model has just four model inputs. Two inputs to *Model e* each require two PMC event counters to derive on our platform. One term in this model counts the L1 cache hits, which we implement as just the L1 data cache hits as all of the limited number of performance counters were in use. Our model uses all six PMCs and the separate cycle counter. The number of required PMC events (*No. Evts.*) and the number of model independent variables (n) used in each model is shown in Table V.

TABLE V
Parameters of models included in this comparison

	Source	No. Evts.	n	Adj. R^2	Err. [%]
a	Pricopi *et al.* [25]	6	6	0.747	12.5
b	Walker *et al.* [24]	4	4	0.785	12.3
c	Rethinagiri *et al.* [11]	5	3	0.672	12.7
d	Rodrigues *et al.* [21]	3	2	0.760	15.2
e	Rodrigues *et al.* [21]	6	4	0.897	9.7
P	Proposed	6 + C	7	0.999	2.9

We build these five models (a, b, c, d, e) and our proposed (P) model using our small set of diverse workloads (discussed in Section V) and report the adjusted R^2 to measure how well the models fit their *training* data (Table V). The high adjusted R^2 value achieved by our proposed model (*Model P*) demonstrates how the chosen inputs and model formula captures the largest amount of useful data for predicting power consumption, allowing our model to closely fit the training data. Out of the implemented existing models, *Model e* captures the largest amount of useful information and therefore most closely fits the dataset. We also analyzed the VIF of each independent variable to give an indication of the model stability. *Model e*

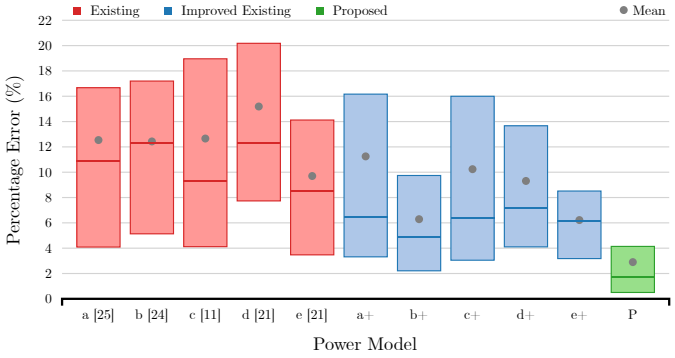


Fig. 13. Box plot of error distribution for each model, trained with 20 diverse workloads and validated with 60 workloads, showing the median, lower quartile, upper quartile and arithmetic mean

has four independent variables and two of the coefficients have a VIF of 15.7 and 9.7, meaning that these coefficients have a standard error $4.0\times$ $[\sqrt{15.7}]$ and $3.1\times$ larger than if no multicollinearity was present. Despite capturing more useful data from the model inputs, our proposed model has less variance inflation, signaling a low amount of repeated data in our seven inputs; the highest VIF of any coefficient in our proposed model is 4.8. The effect of errors in the coefficients can clearly be seen in Table I of [11], where two power models (*Exp 4* and *Exp 5*) for the same system feature the same input (dispatch stalls, D) but with a coefficient of $+1.25$ in one model and -0.47 in the other. This shows how the modeling methodology is not capturing how each variable individually affects the power consumption, forming an unstable model.

We then validate the models on our full set of 60 workloads (equivalent to the D bars in Fig. 7) and show the resulting error distribution (Fig. 13, red and green boxes). The proposed model best utilizes the limited number of training workloads to achieve an average error of 2.9%; $3\times$ smaller than the next best model. Furthermore, the box plot in Fig. 13 shows that our proposed model has a significantly narrower range of errors across the 60 workloads. This means that greater confidence can be placed in our model to contain the error within a smaller interval. For example, the error of *Model e*, which has the smallest average error out of the existing works, has a maximum error of 49%, whereas our proposed model has a maximum error of 13%. The approximate 95% prediction interval (explained in Section VI) of our proposed model is ± 0.014 W whereas the prediction interval for *Model e* is ± 0.13 W.

Pricopi *et al.* [25] (*Model a*) report a low error of 2.6%. However, they build and validate with a small number of 15 workloads and report a minimum and maximum power consumption of 4.54 W and 5.16 W across their training workloads, respectively; a very narrow range when compared to many of our testing workloads (Fig. 10). We find that two independent variables of the model have VIFs larger than 130, contributing to the high average error across our diverse set of 60 workloads (Fig. 13). This highlights the importance of considering stability and validating with a large number of diverse workloads when building reliable power models.

In Section V we highlighted how the cycle count provides

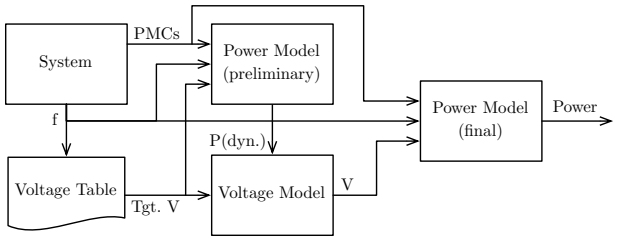


Fig. 14. Run-time power estimation setup with the voltage model

unique data to the model (it is used to determine how much time the cores spend in a low-power [inactive cycle] state). However, this event is not included in any of the existing models (they either use simulated data that may not take this feature into account or an old development board that does not have this feature enabled). We therefore rebuilt models *a*, *b*, *c*, *d* and *e* with the cycle count and named them *a*+, *b*+, *c*+, *d*+ and *e*+, respectively. This improves the existing models but the proposed model still achieves an error 50% lower than any other model (Fig. 13, blue boxes). Note that the cycle counter is *not* used to calculate the IPC (instructions-per-cycle) as it only counts *active* cycles on this platform.

From implementing models of existing work on our platform, we found that: 1) none of the models considered the cycle counter, which has a significant power impact; 2) some of the model inputs were not available in a single PMC event and needed to be calculated from several; and, 3) in one case, a model input could not be deduced from the available PMCs on our platform. These three points highlight the importance of providing a detailed and automated methodology that can be used on CPUs with different ISA, microarchitectures and available PMC events, as we propose. The overhead between the compared models is negligible compared to recording/using the estimated power value; reading a PMC event counter requires a single instruction and the simple model equations have similar complexity, with similar number of used PMC events and model inputs (note that models *c*, *d* and *e* all required extra calculations to derive the required input on our platform).

VIII. CPU Voltage Model

Our power model works with any specified frequency or voltage, which is important for design-space exploration. In online run-time management scenarios, the voltage cannot be directly measured and so the idle voltage for the current operating frequency can be used. However, we find that the static power varies between different workloads at the same frequency because the voltage supplied by the non-ideal voltage regulator varies with the dynamic CPU current draw. This phenomenon greatly affects the power consumption and needs to be taken into account when modeling the power. We present a model built using multiple linear regression to predict the CPU voltage from the current CPU clock frequency, target voltage (idle CPU voltage at that frequency), and dynamic power calculated from the PMC events (Step 4 of our power modeling methodology, Fig. 1). We first use our power model to perform a preliminary prediction of the dynamic power

using the target voltage and then feeding this estimation into our voltage model, along with the target voltage itself (Fig. 14). This voltage model then outputs the estimated run-time voltage and this is used as the voltage input of our main (final) power model. This voltage model is specific to the platform and each frequency needs to be separately considered. To the extent of the authors' knowledge, this is the first work in this area to consider voltage change due to the dynamic load and demonstrate how it can be modeled. To give an example, at a frequency of 1800 MHz, the power model error is 2.6% when using the measured CPU voltage. However, as the measured voltage is not available to the run-time manager of a real device, the target voltage is used, resulting in a significantly larger power error value of 8.5%. If we incorporate our voltage model, we can account for the changing run-time voltage and reduce our power modeling error to 3.0%. We identify this problem, quantify its effect on the error, and demonstrate how our proposed solution is effective in mitigating it. A more detailed analysis of this phenomenon (including the effects of temperature) is the topic of future work.

IX. Conclusion

We have presented a detailed and statistically-rigorous automated methodology and corresponding software tools for building accurate and stable PMC-based run-time power models. We illustrate our approach using measured data from two mobile CPUs with significantly differing microarchitectures: an ARM Cortex-A7 and ARM Cortex-A15. The resulting models achieved an error of 3.8% and 2.8%, respectively, and both achieved an R^2 value of over 0.99. Our approach uniquely considers model stability and we demonstrate how it allows the resulting models to make more accurate predictions on a vast set of diverse scenarios, even when trained on a limited set of workloads. Furthermore, we highlight and address the problem of heteroscedasticity and show how our model formulation and stability allows us to reduce the model formulation experiment time by $100\times$ while trading off less than 0.6% error. We implement our model on a real device and analyse the error and overhead as a function of sample period and we conduct a detailed comparison with the state-of-the-art. We also highlight how the CPU voltage supplied by the non-ideal voltage regulator is sensitive to the dynamic activity of the CPU and we present a CPU voltage model, which improves the accuracy of the power model by as much as 5.5% in situations where the voltage cannot be measured. To address the lack of an established method of collecting PMC data on mobile devices, we present our platform dependent experimental software tools that enables other researchers to make use of high quality, measured data from mobile platforms for investigations requiring measured CPU PMC, temperature, voltage or power data. This paper is supported by online resources (available at [8]) which include the downloadable software tools, usage manuals, raw experimental data and further results, graphs, analysis and explanations. We hope that this work encourages greater statistical rigor in this research area to allow high quality models to be created, models from different works to be compared, and, crucially, the quality and limitations of produced models to be trusted and known.

References

[1] N. Islam and R. Want, "Smartphones: Past, present, and future," *IEEE Pervasive Computing*, no. 4, pp. 89–92, 2014.

[2] D. J. Brown and C. Reams, "Toward energy-efficient computing," *Communications of the ACM*, vol. 53, no. 3, pp. 50–58, 2010.

[3] S. Krishnan, S. V. Garimella, G. M. Chrysler, and R. V. Mahajan, "Towards a thermal moore's law," *IEEE Transactions on Advanced Packaging*, vol. 30, no. 3, pp. 462–474, 2007.

[4] ARM Ltd., "big.LITTLE Technology," http://www.arm.com/products/processors/technologies/biglittleprocessing.php, 2015, [Online; accessed 9-November-2015].

[5] A. K. Das, M. Walker, A. Hansson, B. Al-Hashimi, and G. V. Merrett, "Hardware-software interaction for run-time power optimization: a case study of embedded linux on multicore smartphones," in *Int. Symp. on Low Power Electronics and Design (ISLPED)*. IEEE, 2015.

[6] G. Singla, G. Kaur, A. K. Unver, and U. Y. Ogras, "Predictive dynamic thermal and power management for heterogeneous mobile platforms," in *Proc. 2015 Design, Automation & Test in Europe Conf. & Exhibition*. EDA Consortium, 2015, pp. 960–965.

[7] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[8] M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das, S. Yang, B. M. Al-Hashimi, and G. V. Merrett, "Accurate and stable run-time power modelling for mobile and embedded cpus," http://www.powmon.ecs.soton.ac.uk/powermodeling, Dec 2015, [Online; accessed 22-Dec-2015].

[9] S. Li, J. H. Ahn, R. Strong, J. Brockman, D. Tullsen, and N. Jouppi, "Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Microarchitecture, 2009. MICRO-42. 42nd Annual IEEE/ACM Int. Symp.*, Dec 2009, pp. 469–480.

[10] S. L. Xi, H. Jacobson, P. Bose, G.-Y. Wei, and D. Brooks, "Quantifying sources of error in mcpat and potential impacts on architectural studies," in *High Performance Computer Architecture (HPCA), 2015 IEEE 21st Int. Symp. on*, Feb 2015, pp. 577–589.

[11] S. K. Rethinagiri, O. Palomar, R. Ben Atitallah, S. Niar, O. Unsal, and A. C. Kestelman, "System-level power estimation tool for embedded processor based platforms," in *Proc. 6th Workshop Rapid Simulation and Performance Evaluation: Methods and Tools*, ser. RAPIDO '14. New York, NY, USA: ACM, 2014, pp. 5:1–5:8.

[12] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: methodology and empirical data," in *Microarchitecture, 2003. MICRO-36. Proc. 36th Annu. IEEE/ACM Int. Symp.*, Dec 2003, pp. 93–104.

[13] W. Lee, Y. Kim, J. H. Ryoo, D. Sunwoo, A. Gerstlauer, and L. K. John, "Powertrain: A learning-based calibration of mcpat power models," in *The IEEE Int. Symp. on Low Power Electronics and Design (ISLPED)*, July 2015.

[14] F. Bellosa, "The benefits of event: Driven energy accounting in power-sensitive systems," in *Proc. 9th Workshop on ACM SIGOPS European Workshop: Beyond the PC: New Challenges for the Operating System*, ser. EW 9. New York, NY, USA: ACM, 2000, pp. 37–42.

[15] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E. Ayguade, "Decomposable and responsive power models for multicore processors using performance counters," in *Proc. 24th ACM Int. Conf. Supercomputing*, ser. ICS '10. New York, NY, USA: ACM, 2010, pp. 147–158.

[16] W. Bircher and L. John, "Complete system power estimation using processor performance events," *Computers, IEEE Transactions on*, vol. 61, no. 4, pp. 563–577, April 2012.

[17] S. Sankaran and R. Sridhar, "Energy modeling for mobile devices using performance counters," in *Circuits and Systems (MWSCAS), 2013 IEEE 56th Int. Midwest Symp.*, Aug 2013, pp. 441–444.

[18] G. Da Costa and H. Hlavacs, "Methodology of measurement for energy consumption of applications," in *Grid Computing (GRID), 2010 11th IEEE/ACM Int. Conf.*, Oct 2010, pp. 290–297.

[19] K. Singh, M. Bhadauria, and S. A. McKee, "Real time power estimation and thread scheduling via performance counters," *SIGARCH Comput. Archit. News*, vol. 37, no. 2, pp. 46–55, Jul. 2009.

[20] W. Bircher and L. John, "Complete system power estimation: A trickle-down approach based on performance events," in *Performance Analysis of Systems Software, 2007. ISPASS 2007. IEEE Int. Symp.*, April 2007, pp. 158–168.

[21] R. Rodrigues, A. Annamalai, I. Koren, and S. Kundu, "A study on the use of performance counters to estimate power in microprocessors," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 60, no. 12, pp. 882–886, Dec 2013.

[22] S. Wang, H. Chen, and W. Shi, "Span: A software power analyzer for multicore computer systems," *Sustainable Computing: Informatics and Systems*, vol. 1, no. 1, pp. 23 – 34, 2011.
[23] B. Su, J. Gu, L. Shen, W. Huang, J. L. Greathouse, and Z. Wang, "Ppep: Online performance, power, and energy prediction framework and dvfs space exploration," in *Proceedings of the 47th Annual IEEE/ACM Int. Symp. Microarchitecture*, ser. MICRO-47. Washington, DC, USA: IEEE Computer Society, 2014, pp. 445–457.
[24] M. J. Walker, A. K. Das, G. V. Merrett, and B. Hashimi, "Run-time power estimation for mobile and embedded asymmetric multi-core cpus," in *HIPEAC Workshop on Energy Efficiency with Heterogeneous Computing*. HiPEAC, January 2015. [Online]. Available: http://eprints.soton.ac.uk/372827/
[25] M. Pricopi, T. S. Muthukaruppan, V. Venkataramani, T. Mitra, and S. Vishin, "Power-performance modeling on asymmetric multi-cores," in *Proc. 2013 Int. Conf. Compilers, Architectures and Synthesis for Embedded Systems*, ser. CASES '13, 2013, pp. 15:1–15:10.
[26] M. Kim, K. Kim, J. R. Geraci, and S. Hong, "Utilization-aware load balancing for the energy efficient operation of the big.little processor," in *Proc. Conf. on Design, Automation & Test in Europe*, ser. DATE '14, 2014, pp. 223:1–223:4.
[27] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *Workload Characterization, 2001. WWC-4. 2001 IEEE Int. Workshop on*, Dec 2001, pp. 3–14.
[28] L. McVoy and C. Staelin, "Lmbench: Portable tools for performance analysis," in *Proc. of the 1996 Annu. Conf. on USENIX Annual Technical Conference*, ser. ATEC '96. Berkeley, CA, USA: USENIX Association, 1996, pp. 23–23.
[29] R. Longbottom, "Roy longbottom's pc benchmark collection," http://www.roylongbottom.org.uk, Sept 2014, [Online; accessed 2-Jun-2015].
[30] C. Lee, M. Potkonjak, and W. Mangione-Smith, "Mediabench: a tool for evaluating and synthesizing multimedia and communications systems," in *Microarchitecture, 1997. Proc., 13th Annu. IEEE/ACM Int. Symp.*, Dec 1997, pp. 330–335.
[31] by Michael Kutner, C. Nachtsheim, and J. Neter, *Applied Linear Regression Models*, 4th ed. McGraw-Hill Education, January 2004.
[32] J. F. H. Jr, W. C. Black, B. J. Babin, and R. E. Anderson, *Multivariate Data Analysis*, 7th ed. Prentice Hall, February 2009.
[33] R. Joseph and M. Martonosi, "Run-time power estimation in high performance microprocessors," in *Int. Symp. on Low Power Electronics and Design (ISLPED)*. ACM, 2001, pp. 135–140.
[34] N. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. Hu, M. Irwin, M. Kandemir, and V. Narayanan, "Leakage current: Moore's law meets static power," *Computer*, vol. 36, no. 12, pp. 68–75, Dec 2003.
[35] D. Montgomery, E. Peck, and G. Vining, *Introduction to Linear Regression Analysis*, ser. Wiley Series in Probability and Statistics. Wiley, 2015.
[36] H. Yang, "Visual assessment of residual plots in multiple linear regression: A model-based simulation perspective," *Multiple Linear Regression Viewpoints*, 2012.
[37] R. C. Dorf, *The Technology Management Handbook*, 1st ed. CRC Press, July 1998.
[38] A. F. Hayes and L. Cai, "Using heteroskedasticity-consistent standard error estimators in ols regression: An introduction and software implementation," *Behavior Research Methods*, vol. 39, no. 4, pp. 709–722, 2007.
[39] J. G. MacKinnon and H. White, "Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties," *Journal of Econometrics*, vol. 29, no. 3, pp. 305–325, Sep. 1985.
[40] J. S. Long and L. H. Ervin, "Using heteroscedasticity consistent standard errors in the linear regression model," *The American Statistician*, vol. 54, no. 3, pp. pp. 217–224, 2000.
[41] J. Perktold, S. Seabold, and J. Taylor, "Statsmodels," http://statsmodels.sourceforge.net, Nov 2015, [Online; accessed 16-November-2015].

Matthew J. Walker Matthew J. Walker received his M.Eng. degree in Electronic Engineering (Hons) from the University of Southampton, UK, in 2013. He is a PhD student in Electronic Engineering at the University of Southampton. In 2015 he held an internship at ARM working on run-time power models.

Stephan Diestelhorst Stephan Diestelhorst is a Staff Research Engineer at ARM, Cambridge, UK. He leads a group that predicts and improves power consumption of future SoCs using the gem5 simulator and native hardware. Before joining ARM in 2013, Stephan worked at Advanced Micro Devices, in Dresden, Germany as a researcher in the Operating Systems Research Center. He received his degree in Computer Science from Technische Universität Dresden. He has a number of publications and patents.

Andreas Hansson Andreas Hansson received an MSc in computer science and engineering from Lund Institute of Technology, Sweden, in 2005. The same year he joined Philips Research as part of an industrial PhD. In 2009 he graduated, cum laude, from Eindhoven University of Technology, the Netherlands. Currently, he is working as a strategic technical advisor at ARM Ltd, United Kingdom. His work has been published in numerous conferences and journals, and has received several awards.

Anup Das Dr. Anup Das received the B.Eng. degree in Electronics and Telecommunication Engineering from Jadavpur University, India, in 2004. He received the Ph.D. degree in computer engineering in the area of embedded systems from the National University of Singapore, in 2014. He is currently a post-doctoral research fellow at the University of Southampton. His research interests include reliability and energy-aware system architecture.

Sheng Yang Sheng Yang received the B.E degree in Electronic Engineering from University of Southampton, UK, in 2008, and Ph.D. degree in Electronics Engineering from University of Southampton, UK., in 2013. In 2007 he did an internship with NXP modelling a data hub using systemC. In 2011 he held an internship with ARM investigating data integrity of flip-flops. Currently he is working as a research fellow in the School of Electronics and Computer Science, University of Southampton.

Bashir M. Al-Hashimi Bashir M. Al-Hashimi (M99-SM01-F09) is a Professor of Computer Engineering and Dean of the Faculty of Physical Sciences and Engineering at University of Southampton, UK. He is ARM Professor of Computer Engineering and Co-Director of the ARM- ECS research centre. His research interests include methods and tools for low-power design and test of embedded computing systems. He has published over 300 technical papers, authored or co-authored 5 books and has graduated 33 PhD students.

Geoff V. Merrett Geoff V. Merrett (GSM06-M09) received the BEng degree (Hons) in Electronic Engineering and the PhD degree from the University of Southampton, UK, in 2004 and 2009 respectively. He was appointed as a Lecturer in energy-efficient electronic systems at the University of Southampton in 2008, and was promoted to Associate Professor in 2014. He has published over 100 scientific papers in journals and refereed conference proceedings.