PowerTrain: A Learning-based Calibration of McPAT Power Models

Wooseok Lee[1], Youngchun Kim[1], Jee Ho Ryoo[1], Dam Sunwoo[2], Andreas Gerstlauer[1], and Lizy K. John[1]
[1]The University of Texas at Austin*, [2]ARM R&D, Austin, TX

***Abstract*—As research on improving energy efficiency becomes prevalent, the necessity of a tool to accurately estimate power is increasing. Among various tools proposed, McPAT has gained some popularity due to its easy-to-use analytical power models. However, McPAT's prediction has several limitations. Although under- or over-estimated power from unmodeled and mis-modeled parts offset each other, it still incorporates errors in each block. Moreover, the lack of awareness to the implementation details exacerbates the prediction inaccuracies. To alleviate this problem, we propose a new methodology to train McPAT towards precise processor power prediction using power measurements from real hardware. This calibration enables McPAT's power to fit to the target processor power. Once we adjusted the power consumption of each block to best match those in the target processor, our trained McPAT delivered more precise power estimation. We calibrated the outputs of McPAT against a Cortex-A15 within a Samsung Exynos 5422 SoC. We observe that our methodology successfully reduces the errors, particularly for workloads with fluctuating power behaviors. The results show that the mean percentage error and the mean percentage absolute error of the calibrated power against real hardware are 2.04 percent and 4.37 percent, respectively.**

I. Introduction

With power being the primary design concern today, there is a need for accurate models to perform energy estimation in related studies. Among available power models, McPAT [1] has gained popularity due to its ease-of-use and readiness. Hierarchical modeling from a circuit level to high-level system architecture provides reasonably accurate yet fast power estimation for a target architecture. Without having to struggle with RTL designs, McPAT enables researchers to predict power and area using a single configuration file with activity factors.

However, despite McPAT's easy-to-use power estimation, it has limitations predicting power of post-silicon processors. According to Xi et al. [2], original McPAT's estimates contain significant errors predicting dynamic power of the IBM Power7 processor due to its incompleteness, architectural disparities in model, and input and coding errors. In addition, the power characteristic of one processor varies largely depending on the implementation target [3]. However, McPAT has limitations adapting backend implementation design details even with its effort to accommodate some technology nodes and implementation parameters.

In this paper, we propose a novel, learning-based calibration approach called *PowerTrain* aimed at alleviating these problem. Instead of relying solely on McPAT's prediction ability, we propose to calibrate the outputs of McPAT against an existing post-silicon processor implementation. One fundamental reason of McPAT's error is that it cannot model all processors with high level of accuracy. Therefore, in order to obtain precise power models for existing processors, post-implementation calibration work is imperative. Thus, the goal of this paper is to provide a methodology to calibrate McPAT for a precise power model targeting post-silicon processors.

We apply regression to train McPAT. Once McPAT generates power of each block, total power is obtained by aggregating power from each block. Rather than just adding the estimated power, we trained them to fit to the target processor power obtained by measurement on actual hardware. By having each coefficient trained to fix the errors against target processor, the calibrated power naturally fits to the power

*This work is supported in part by SRC grant 2012-HJ-2317

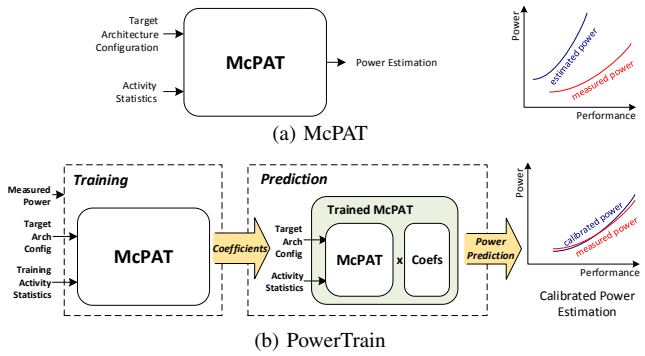


Fig. 1: *PowerTrain* overview.

behavior of the target processor. Figure 1 shows overview of our methodology. Coefficients obtained from training are multiplied to the power of each block, generating calibrated total power estimation. Experimental results show that mean percentage error (MPE) and mean percentage absolute error (MPAE) after calibration is 2.04 and 4.37 percent, respectively.

The contributions of this paper are as follows. First, we propose a novel and simple method to improve McPAT's prediction accuracy. Prior work [2] to increase the accuracy of McPAT requires tremendous engineering efforts and often tedious iterative manual calibration. In addition, detailed power models or real hardware designs such as RTL are needed for each block. With our approach, any researcher can build a precise power model targeting a specific processor using only one total processor power measurement.

Second, our methodology provides a controlled and validated use of McPAT. After training, McPAT delivers precise and reliable prediction for the target processor. The resulting power estimation can be much faster than RTL-based tools while being as accurate. This enables third-party developers to evaluate power consumption of software running on the target processor using virtual platforms without early access to implementation details or real silicon. Furthermore, calibrated model can provide better insights into sensitivity of power to system- or micro-architecture changes.

Lastly, we provide an accurate McPAT baseline of our target processor and quantify the estimation errors when targeting an existing processor. After thorough reviews and iterative tests, the configurations and mappings of performance counters to activity factors are determined. In particular, our work compares and validates the power of an atypical mobile processor other than the high-end processors previously validated with McPAT [1].

II. Related Work

From high level power modeling using activity information to low level approaches with RTL, various processor power modeling methodologies have been proposed. Isci et al. [4] and Sun et al. [5] provide a method to estimate power from performance counters by deriving access rates with domain specific knowledge and piece-wise linear equations that account for the irregularity of power behavior. Bircher et al. [6] proposed a methodology that avoids the need for manual mapping of performance counters by leveraging linear

regression across the top 3 or 4 counters that shows high correlation with power. Lee et al. [7] proposed a way to explore architectural changes with pareto frontier analysis and predictor interaction in regression modeling. Although these high level approaches all allow fast prediction, they have limitations in terms of accuracy since the information used is only power numbers and activity statistics. On the other hand, power models using low-level, information, such as RTL, are able to account for, the details of hardware implementation and glue logics that high-level, counter-based models can not [8]. Sunwoo et al. [9] proposed learning-based power models based on information obtained from low-level implementation simulation. With detailed information, such approaches enable power models at high cycle-level accuracy. However, low-level approaches are slow and often require large effort to obtain necessary details. By contrast, McPAT delivers power estimation with minimal effort while maintaining a relatively high level of accuracy. Lim et al. [10] present an approach to calibrate a McPAT model for GPUs using a manual trial-and-error methodology. *PowerTrain* differs from their work in that we provide a systematic approach to train McPAT without requiring any manual intervention.

III. McPAT Background

McPAT builds up power estimation with a bottom-up approach. From detailed circuit and area modeling, McPAT models major components of power consumption within a processor, such as functional units, predictors, buffers and other memories. Subsequently, it models sub-blocks in microarchitecture into a complete model at the processor level. This bottom-up approach delivers reasonable power estimation for the target block with various architectural parameter changes. In addition, based on domain specific knowledge, the general microarchitecture of a processor is modeled. McPAT power models targeting in-order and out-of-order processors can be constructed with simple changes of configurations.

As such, estimating power with McPAT requires much less efforts than low level approaches, since it leverages only architectural configuration with activity information. It is easy to estimate the power of a new processor without considering low level details. By providing the configuration, McPAT can construct the corresponding analytical power models. The dynamic power is predicted by multiplying dynamic power per operation from those models with activity factors. In the end, static power is summed to the dynamic power, producing one total power consumption of the processor. All the users provide is only target processor configuration with corresponding activity information.

However, even though McPAT's ability to estimate power is better than abstract high-level approaches, it still lacks accuracy compared to low level approaches. Thus, in this paper, our goal is to improve the accuracy by eliminating errors from McPAT using training-based approach. In the following sections, we first identify the sources of errors and propose a new methodology to reduce such errors.

A. Sources of Errors in McPAT

According to Xi et al. [2], the sources of errors in McPAT when predicting power consumption of an IBM Power7 processor are *abstraction error*, *modeling assumption error*, *input error*, and *coding error*, in the decreasing order of significance. As described in detail in [2], sources of errors are:

- *Abstraction error*: This category of error results from the incomplete modeling. Since McPAT supports various architectures, only common components across various architectures are modeled. The rest is unmodeled, which is a huge source of the error.
- *Modeling assumption error*: This category of error is caused by the limitation of McPAT in that it is unable to support all the details of various microarchitectural components. The differences between the modeled components and the actual hardware generate the misprediction.
- *Input error*: This category of error is originated from the vagueness of configuring architectural/microarchitectural parameters and mapping activity factors to the inputs of McPAT.
- *Coding error*: This category of error is attributed to human mistakes when coding the models.

The authors in [2] argued that these errors might work in opposite directions to offset each other, resulting in error canceling effects. For instance, overestimated power of modeled blocks may compensate for the power dissipation of unmodeled blocks. Thus, McPAT seems to be correct even though errors still exist in individual blocks.

In addition to these four errors, limitations of adapting to implementation details exacerbates the prediction error. Processors manufactured from the same RTL can show dissimilar power behavior depending on the backend implementation target. Nvidia Tegra processors [3] harness this implementation diversity to improve energy efficiency. From one RTL, they derived versions of processor for both high performance and low power, switching between them depending on the workloads types. McPAT, however, mainly focuses on microarchitecture level power modeling, as such it is not able to capture the power variations stemming from the backend implementation. With McPAT's estimation, power estimation of all Tegra processor variants would remain the same since the configuration is identical. In this paper, we assume that implementation differences can be captured as model assumption errors by appropriately scaling the existing estimation.

B. McPAT Baseline

No specific rules or methodologies exist on how to configure McPAT to model a specific architecture and processor. In addition, very little systematic studies exist in prior work [11]. This uncontrolled of or lack of knowledge results in huge confusion to researchers when preparing baseline McPAT configuration and activity files. Thus, for the purpose of comparisons in this paper, we define four baselines as follows.

- ***MB0***: From the previous works [11] and to the best of our microarchitectural knowledge [12], we configure and map activity factors. We provide zeros to the activity factors that do not have direct information from performance counters. This results in some block powers being distorted.
- ***MB1***: We added the missing activity factors by extracting constant ratios from available counters based on profiled information. For instance, the number of register file reads can be extrapolated from the number of integer instructions issued. We regard this as our best baseline.
- ***MB2***: We intentionally change MB1 configurations and inputs to induce a type of *modeling assumption error* that has proportional error relationship. By scaling activity factors of L2 cache down, the estimated power becomes lower than *MB1*. We denote this error as *scale error*.
- ***MB3***: We intentionally change MB1 configurations and inputs to induce a type of *modeling assumption error* that has irregular error relationship. By scaling activity factors of L1 Data and L2 caches in opposite directions, the estimated power sometimes fluctuates over or below *MB1*. We denote this error as *irregularity error*.

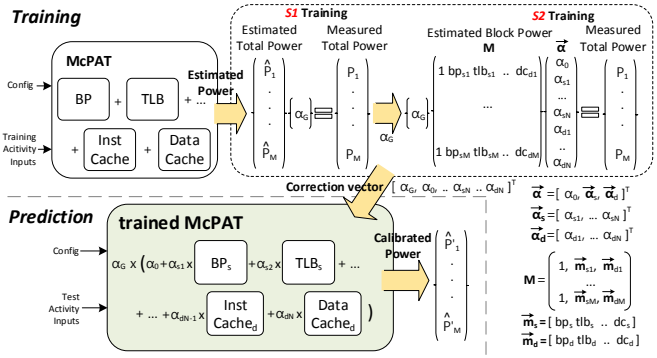


Fig. 2: Training McPAT with regression

The detailed power of each baseline is presented in the following section (Sec. IV-C). With these baseline predictions, we quantify the errors from McPAT against measured power and show how our approach reduces the errors. One noticeable point is that default power of McPAT is fixed to 1V if not specified, but our target processor's operating voltage is about 1.3V. Moreover, providing custom voltage to McPAT generates incorrect numbers. However, we decide not to modify anything arbitrarily to avoid additional errors induced from it. Rather, we regard them as an error and show how the error from voltage disparity can be calibrated by our methodology.

IV. PowerTrain

Power models from McPAT innately has limitations when predicting power of a target processor since it cannot capture all the hardware details of the processor. Particularly, if the target processor is very different from the ones that were modeled and validated in McPAT, errors may exceed the reasonable range to be accepted as a valid prediction. In our paper, we focus on reducing the error of McPAT within a reasonable range so that it serves as a valid predictor.

Two major ways to improve the accuracy exist. A fundamental approach is increasing the level of detail in a model. By increasing the details and subsequently fixing errors, we can increase the accuracy of the model. However, this conventional method encompasses repetitive and manual tasks with in-depth hardware analysis [2]. First, measured and estimated power are compared. If any discrepancy is found, McPAT is corrected either by changing errors such as configuration or by modifying internals of McPAT to fit to the power of the target processor. Once fixed, the same process is repeated to confirm whether the fix actually improves the overall prediction accuracy and to find another source of error. The task should be iterated until the average error rates converge to the acceptable range. In many cases, the real hardware or sufficient information to build precise power models are not available. Hence, it is difficult to obtain precise power models with this approach. Further, this methodology requires similar amount of effort again for power models of all subsequent target processors.

An alternative way to fix the errors is calibration. Instead of increasing architectural details for accuracy, we propose a way to adjust McPAT's estimation to be suitable to the power of the target processor. In particular, this methodology fixes errors by adjusting the power of each block so that the error of total power is kept within acceptable ranges. Our methodology, *PowerTrain*, can deliver a precise power model for a specific processor in a relatively simple manner. It does not require accurate sub-block power model nor any update to McPAT. The following sections describe the details of how we train McPAT and eventually remove errors.

A. Training Framework for McPAT

PowerTrain, which trains McPAT, delivers a precise power model with minimal effort compared to the trial-and-error approach. Figure 2 shows the procedures of how we train and test McPAT. We first extract the power output of each block from McPAT by providing both architectural configurations and training activity factors as inputs. Once power numbers for each block are obtained, we train them to scale each power estimation with the corresponding coefficient. Lastly, the coefficients are applied to McPAT and we test the model with the test inputs. For appropriate operation, we split the coefficients into static and dynamic power.

We employ a two-step adaptive training procedure to fix errors effectively. During the first step, we use both the total power from McPAT and the measured power from hardware to obtain one global α_G. This value is used to shift all predicted power close to the measured power. With this step, the power error induced from voltage and implementation differences can be removed. We define the first step as *S1*. In the second step, defined as *S2*, we calibrate shifted power to the measured power again to have coefficients which control each block's power at a finer granularity. After the second step, most of the errors are reduced, and the calibrated McPAT can be used as a valid source of prediction. Formal definition of these steps are defined as follows:

$$\hat{p_m} = \alpha_G \times (\alpha_0 + \vec{\mathbf{m}}_s \times \vec{\boldsymbol{\alpha}}_s + \vec{\mathbf{m}}_d \times \vec{\boldsymbol{\alpha}}_d) \quad (1)$$

where $\hat{p_m}$ denotes total power estimation, $\vec{\mathbf{m}}_s$ and $\vec{\mathbf{m}}_d$ is $1 \times N$, $\vec{\boldsymbol{\alpha}}_s$ and $\vec{\boldsymbol{\alpha}}_d$ is $N \times 1$, and N is the number of blocks in McPAT. s and d denotes static and dynamic power, and correspondingly α_s and α_d, and m_s and m_d are static and dynamic regression coefficients and each block's estimated power from McPAT, respectively. We also introduce α_G as global scaling factor and α_0 as an overall additive bias to account for unmodeled glue components.

This approach allows us to correct the errors without modifying the internal or external configuration or activity inputs of McPAT. By applying a non-negative least square regression (IV-B), we can obtain a set of coefficients, a correction vector, which controls power of each block to best fit to the total power. Once the correction vector is obtained, it is multiplied to the outputs of McPAT to generate calibrated power estimation of the processor. In this way, we keep original McPAT's prediction ability, yet obtain accurate results closer to the target processor. We perform two-step calibration since experiments have shown that block-level coefficients will vary widely depending on the training set if a global scaling factor is not first accounted for. By including α_G, coefficients are stable across benchmarks. Details of the results and methods for obtaining the coefficients will be described in the following sections.

Moreover, one power measurement, the total processor power consumption, is the only information required per timestamp for our methodology. Conventional compare-and-update methods mandate per-block or more detailed hardware power references for error correction. However, since we rely on regression to find an optimal solution, the total power information is enough to extract the coefficients of each block. This allows an accurate and reliable power model to be constructed using only limited access to widely available hardware resources, such as board-level power and counter measurements.

B. Least Mean Square Error Problem

Diverse approaches to obtain coefficients for optimal solution of least mean square error (LMSE) problem have been widely proposed. However, general LMSE solvers, such as simple linear regression,

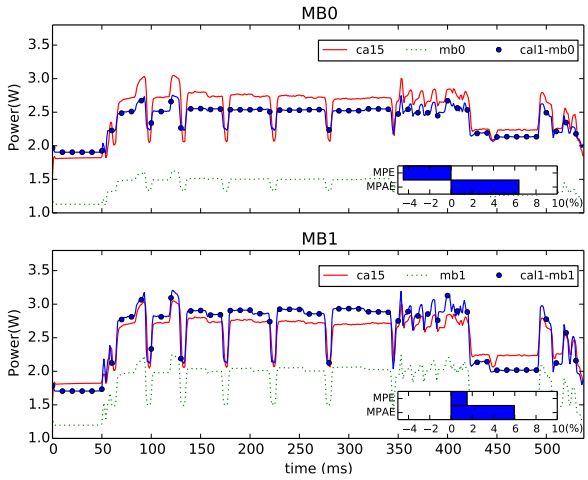


Fig. 3: S1-calibrated Power: MB0 and MB1 (*sift*)

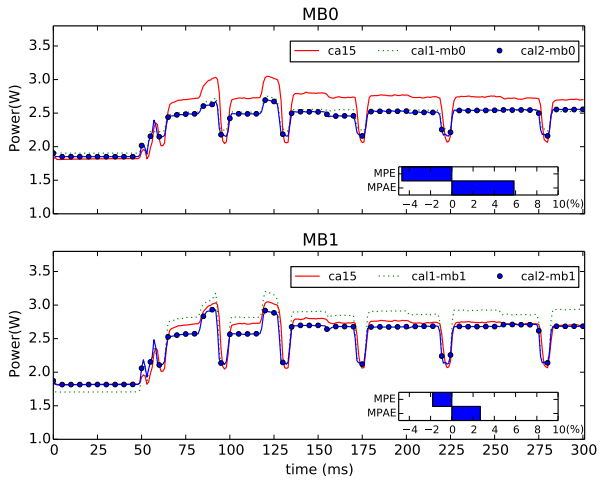


Fig. 4: S2-calibrated Power: MB0 and MB1 (*sift*)

produce negative coefficients for a given problem. Despite negative coefficients generating optimal solution, the negative power inferred from the negative coefficients gives no insight if we try to analyze the power behavior from the equation. Thus, we discretely select non-negative least square solver [13] for our methodology. The problem that we are trying to solve is as follows.

$$\begin{aligned} &minimize\ \|\mathbf{M}\overrightarrow{\boldsymbol{\alpha}} - \overrightarrow{\mathbf{P}}\|^2 + \lambda\|\overrightarrow{\boldsymbol{\alpha}}\|, \\ &subject\ to\ \boldsymbol{\alpha} \geq 0,\ \ \lambda = 0.01 \end{aligned} \tag{2}$$

where $\overrightarrow{\mathbf{P}}$ is $m \times 1$, $\mathbf{M}$ is $m \times n$, α is $n \times 1$. $\overrightarrow{\boldsymbol{\alpha}}$ is regression coefficients, *n* and *m* are the number of coefficients and observations, and let $\mathbf{M}$ and $\overrightarrow{\mathbf{P}}$ denote observations and responses, respectively.

From Equation 2, we successfully derive α_G and the correction vector. Figure 10 shows the correction vector when we apply our methodology to a *MB1* baseline. In general, static power is either slightly suppressed or remains the same, while some dynamic power components are dramatically reduced or increased.

C. Reducing Errors

As will be described in detail in Section V, we have applied our methodology to calibration of a ARM Cortex-A15 processor. To observe how our methodology reduces errors, we first apply our calibration to *MB0* and *MB1* baselines. For appropriate experiments, we select two benchmarks, sift and susan, that show exemplary error behavior we want to fix. The rest of the benchmarks are used as

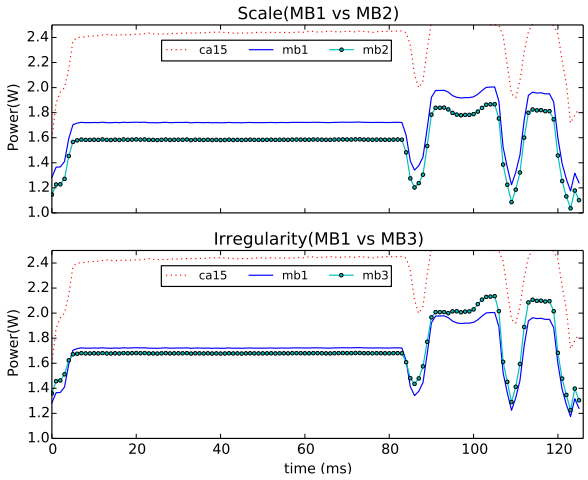


Fig. 5: McPAT baseline: MB2 and MB3 (*susan*)

training sets as described in Section V-A. In all cases, we evaluate overall prediction accuracy as the mean percentage error (MPE) and mean percentage absolute error (MPAE) of predicted versus measured results. In the following, we focus on the specifics of how our methodology reduces the error from lack of information as well as how it achieves overall accuracy improvements.

As mentioned in Section III-B, the first error source in McPAT is a discrepancy in basic technology and operating voltage assumptions. Figure 3 reveals the raw output power from McPAT and subsequent errors against target processor power. However, the first level calibration, *S1*, removes the issue and delivers high level of accuracy for both *MB0* and *MB1*. One noticeable point (Fig. 3) is that some power behavior of *MB0* cannot capture the real processor power behavior (at about 60 ms) due to the lack of activity inputs, while *MB1* captures similar behavior even with slightly large movement.

As discussed previously in Section IV-A, abstraction and input errors originate from unmodeled or missing processor components and input activity. While McPAT shows high level of accuracy when modeling memory parts in micro-architecture, the rest such as inter-block muxes and controls are difficult to model. In addition, due to the implementation differences in each processor, it is hard to generalize the hardware, resulting in unmodeled parts. Since several activity factors of *MB0* is not provided (or provided as zero in reality), *MB0* can be regarded as exemplary for *abstraction* and *input error* when compared to *MB1*, which has all inputs - i.e., zero-provided block power represents the unmodeled parts of the hardware.

Figure 4 shows the results after the second level calibration, *S2*. After we apply *S2*, which enables the fine-grained scaling of each block power, MPAE of *MB1* is reduced to around 2 percent. The adjusted power of each block is either added or subtracted so that the summed-up power behaves much closer to the real power behavior. For calibrated *MB0*, MPAE remains similar to 4 percent. Interestingly, however, the *abstraction error* around 60 ms is corrected to match the real processor power behavior after we apply *S2* to *MB0* due to the correction vector of each block.

Model assumption error is attributed to the differences between hardware and model. To observe how our methodology handles the problem, we intentionally induce two types of errors, *scale* and *irregularity* represented by *MB2* and *MB3*, respectively. Figure 5 shows the baseline McPAT power of MB2 and MB3. As MB2 shows, its power is always less than MB1. Since we reduced the L2 cache power with constant ratios (i.e. scale), total power of MB2 is less than MB1 to some degree proportional to the L2 cache activities.

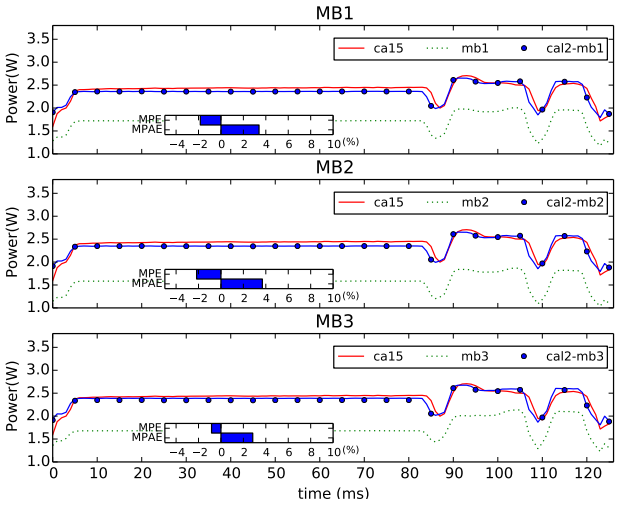


Fig. 6: S2-calibrated Power: MB1, MB2, and MB3 (*susan*)

Conversely, power of MB3 varies against MB1. In some periods, the power of MB3 is less than MB1 while in some other periods MB3 power is bigger than MB1 (i.e. irregularity). Figure 6 shows the calibrated power from MB2 and MB3. Even with the various types of errors, the calibrated power converges to the original power in the end. This result demonstrates that our approach is very effective in removing modeling assumption type of errors.

V. Experimental Results

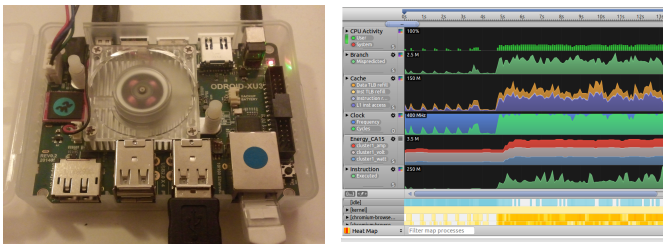
In this section, we start by describing our target board specification and how we measure power and performance. We use ARM Cortex-A15 as our baseline processor since not only does it incorporate the most advanced power saving techniques such as aggressive dynamic clock and power gating, but has a distinctive power characteristics in that power budget, ISA, and micro-architecture is discrepant to the processors that were modeled and validated for McPAT.

A. Experimental Setup

For the power measurement of a post-silicon processor, we use the *ODROID-XU3* board (Fig. 7) running Ubuntu 14.04 OS due to its energy monitor sensors. Specifically, it comes with a Samsung Exynos 5422 system-on-chip (quad Cortex-A15 and Cortex-A7 cores) manufactured on the 28nm HKMG process. Although we use a specific board, our methodology can be applied to any processor with performance counters and power measurement available.

Table I: Cortex-A15 performance counters.

Performance Counter Event	Description
CPU_CYCLES	Cycle
INST_RETIRED	Inst architecturally executed
INST_SPEC	Inst spec exec
LD_SPEC	Inst spec exec, load
ST_SPEC	Inst spec exec, store
ASE_SPEC	Inst spec exec, Advanced SIMD
VFP_SPEC	Inst spec exec, floating-point
DP_SPEC	Inst spec exec, int data processing
BR_IMMED_SPEC	Branch spec exec, immediate branch
BR_RETURN_SPEC	Branch spec exec, procedure return
BR_INDIRECT_SPEC	Branch spec exec, indirect branch
BR_MIS_PRED	Mis- or not predicted branch spec
L1I_CACHE	Level 1 instruction cache access
L1I_CACHE_REFILL	Level 1 instruction cache refill
L1I_TLB_REFILL	Level 1 instruction TLB refill
L1D_CACHE_LD	Level 1 data cache access, read
L1D_CACHE_ST	Level 1 data cache access, store
L1D_CACHE_REFILL_LD	Level 1 data cache refill, read
L1D_CACHE_REFILL_ST	Level 1 data cache refill, write
L1D_CACHE_WB	Level 1 data cache write-back
L1D_TLB_REFILL	Level 1 data TLB refill
L2D_CACHE_LD	Level 2 data cache access, read
L2D_CACHE_ST	Level 2 data cache access, write
L2D_CACHE_REFILL_LD	Level 2 data cache refill, read
L2D_CACHE_REFILL_ST	Level 2 data cache refill, write
L2D_CACHE_WB	Level 2 data cache write-back



(a) Odroid Board (b) DS-5

Fig. 7: *ODROID-XU3* board and ARM DS-5 Streamline tool

The *ODROID-XU3* board allows us to independently monitor the power consumption of the Cortex-A15 and the Cortex-A7 clusters, GPU, and memory. Monitoring the power consumption of individual Cortex-A15 cores is limited since the Cortex-A15 cluster power reports the combined power of the quad cores and the shared L2 cache. Therefore, to measure a single core power consumption, we enable only one Cortex-A15 core while three other cores are disabled.

For power measurement, we use the ARM DS-5 Streamline performance analyzer (Fig 7). This tool allows us to collect periodic real-time performance data from the target processor. However, one problem is that, by default, the current sensors update data every 250ms. Since we want a finer granularity, we changed the device driver to update at 1ms intervals. In addition, we modified *gator* driver, a proxy Linux device driver which collects performance counters from CPU, to capture power every 1ms (the same collection period for the other performance counters).

As a source of activity factors, we collect performance counters from the CPU using the same DS-5 tool and method. Table I shows the twenty-six counters of interest. One challenge when collecting hardware counters is that the number of counters that can be monitored simultaneously on the Cortex-A15 is limited to six. Thus, we run applications in groups and merge the data into one. To reduce the error, application runs are repeated 30 times for a total of 150 executions for each application and data only within half the standard deviation range are averaged.

As training and test input, we select MiBench (basicmath, bitcount, dijkstra, fft, patricia, jpeg, qsort, susan, typeset) [14], and SD-VBS (disparity, mer, multi_ncut, sift, svm, texture_synthesis) [15]. Since single core performance is measured and we need a deterministic and reproducible processor behavior for data synchronization, we only select single-threaded applications.

B. Model Validation

We first employ comprehensive 15-fold cross-validation to test our methodology [16]. Specifically, we evaluate our model using observations of one previously unseen program as test set where the remaining programs are used as training set. Since we have fifteen benchmarks, fifteen total validation processes were conducted. Figure 9 shows MPAEs of *MB0* and *MB1*. The global shift from *S1* calibration removes most of the errors, while the finer granularity corrections(*S2*) improves the accuracy more. Although general accuracy is increased after *S2* (*MB1*), some benchmarks such as *bitcount* and *multi_ncut* show less prediction performance. These benchmarks tend to show relatively stable power movements while others such as *sift* and *jpeg* shows fluctuating power behavior. This matches with

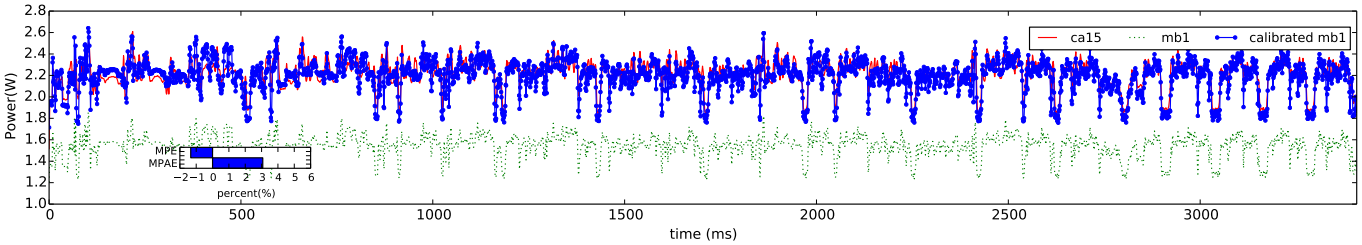


Fig. 8: SPEC CPU2006 *gcc* is calibrated with coefficients from MB1.

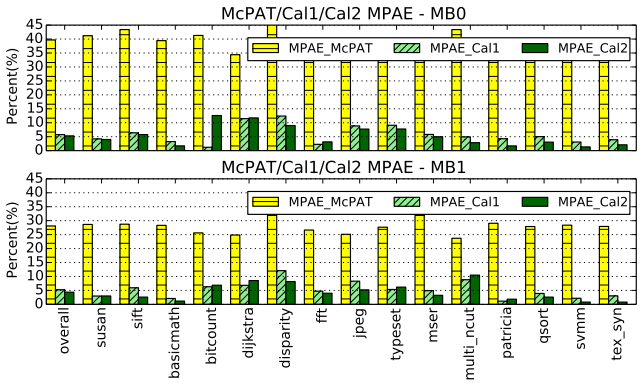


Fig. 9: MPAEs from validation (MB0 and MB1)

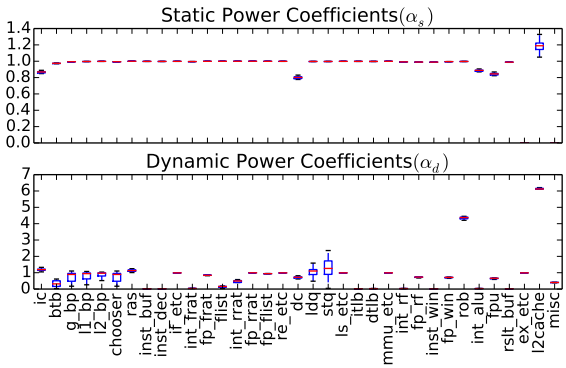


Fig. 10: Correction vector variation (*S2-MB1*)

Table II: Cross-validation results

	S2-MB0	**S2-MB1**	**S2-MB2**	**S2-MB3**
MPE	2.49 (%)	2.04 (%)	2.08 (%)	2.13 (%)
MPAE	5.29 (%)	4.37 (%)	4.60 (%)	4.47 (%)

our previous finding that *S2* performs well when restoring the power behavior and reducing errors. Since *MB0* incorporates more errors, *S2* works better on *MB0* than on *MB1*, except for one outlier, *bitcount*. Table II shows the final results of each baseline. For *MB1* baseline, the average MPE and MPAE is 2.04 percent and 4.37 percent, respectively. Although *MB0* has relatively large abstraction errors, our methodology corrects them and delivers power models which have an error within a reasonable range. In addition, to evaluate the sensitivity of our methodology, we measured the variance of correction vectors. As shown in Figure 10, the coefficients are stable.

Finally, to find out if our model performs well with new programs, particularly with excessive power movements, we add one more test program, *gcc* from SPEC CPU2006 benchmark suite. However, since our sampling rates of 1ms prohibited us from using reference inputs for validation, we tested our model with only test input sets. We believe that our model has similar performance with reference inputs. We calibrate McPAT's output from *gcc* with coefficients from *MB1*. As shown in Fig. 8, calibrated power shows a good match with measured power, and MPAE is about 3 percent.

VI. Conclusions

In this paper, we propose a calibration methodology, *PowerTrain*, to increase the accuracy of McPAT. Instead of manual comparison and correction against hardware measurements, our methodology achieves high accuracy with minimal effort. Our results show that a McPAT power model calibrated using our approach achieves 2% MPE and 4% MPAE compared to measurements from real silicon. We believe that the validated use of McPAT will help researchers to perform more reliable software and system studies. In the future, we plan to work towards release of a general calibration tool and further evaluate accuracy and sensitivity of our approach across a wider range of workloads, platforms and training sets.

References

[1] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, and D. M. Tullsen, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Micro*, 2009.

[2] S. Xi, H. Jacobson, P. Pose, G.-Y. Wei, and D. Brooks, "Quantifying sources of error in McPAT and potential impacts on architectural studies," in *HPCA*, 2015.

[3] "NVIDIA variable SMP," http://www.nvidia.com/content/pdf/tegra_white_papers/tegra-whitepaper-0911b.pdf.

[4] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *Micro*, 2003.

[5] Y. Sun, L. Wanner, and M. Srivastava, "Low-cost estimation of sub-system power," in *IGCC*, 2012.

[6] W. L. Bircher, M. Valluri, J. Law, and L. K. John, "Runtime identification of microprocessor energy saving opportunities," in *ISLPED*, 2005.

[7] B. C. Lee and D. M. Brooks, "Illustrative design space studies with microarchitectural regression models," in *HPCA*, 2007.

[8] J. C. McCullough, Y. Agarwal, J. Chandrashekar, S. Kuppuswamy, A. C. Snoeren, and R. K. Gupta, "Evaluating the effectiveness of model-based power characterization," in *USENIXATC*, 2011.

[9] D. Sunwoo, G. Y. Wu, N. A. Patil, and D. Chiou, "PrEsto: An fpga-accelerated power estimation methodology for complex systems," in *FPL*, 2010.

[10] J. Lim, N. B. Lakshminarayana, H. Kim, W. Song, S. Yalamanchili, and W. Sung, "Power modeling for GPU architectures using McPAT," *ACM TODAES*, vol. 19, no. 3, pp. 26:1–26:24, Jun. 2014.

[11] D. S. Khudia, "From GEM5 Statistics to McPAT Input," http://web.eecs.umich.edu/ dskhudia/blog/gem5–to–mcpat/.

[12] "ARM Cortex-a15 technical reference manual," http://infocenter.arm.com/help/topic/com.arm.doc.ddi0438i/DDI0438I_cortex_a15_r4p0_trm.pdf.

[13] K. Koh, S. jean Kim, S. Boyd, and Y. Lin, "An interior-point method for large-scale l1-regularized logistic regression," *Journal of Machine Learning Research*, 2007.

[14] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge, and R. B. Brown, "Mibench: A free, commercially representative embedded benchmark suite," in *WWC-4*, 2001.

[15] S. K. Venkata, I. Ahn, D. Jeon, A. Gupta, C. Louie, S. Garcia, S. Belongie, and M. B. Taylor, "Sd-vbs: The san diego vision benchmark suite," in *IISWC*, 2009.

[16] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection," in *IJCAI*, 1995.