Alternating Blind Identification of Power Sources for Mobile SoCs

Sofiane Chetoui
Brown University
Providence, USA
sofiane_chetoui@brown.edu

Michael Chen
Brown University
Providence, USA
michael_chen@brown.edu

Abhinav Golas
Meta Platforms
Burlingame, USA
agolas@fb.com

Farrukh Hijaz
Meta Platforms
Redmond, USA
fhijaz@fb.com

Adel Belouchrani
Ecole Nationale Polytechnique
Algiers, Algeria
adel.belouchrani@g.enp.edu.dz

Sherief Reda
Brown University
Providence, USA
sherief_reda@brown.edu

ABSTRACT

The need for faster Systems on Chip (SoCs) has accelerated scaling trends, leading to a considerable power density increase and raising critical power and thermal challenges. The ability to measure power consumption of different hardware units is essential for the operation and improvement of mobile SoCs, as well as the enhancement of the power efficiency of the software that runs on them. SoCs are usually enabled with embedded thermal sensors to measure the temperature at the hardware unit level; however, they lack the ability to sense the power. In this paper we introduce an Alternating Blind Identification of Power sources (Alternating-BPI), a technique that accurately estimates the power consumption of individual SoC units without the use of any design based models. The proposed technique uses a novel approach to blindly identify the sources of power consumption, by relying only on the measurements from the embedded thermal sensors and the total power consumption. The accuracy and applicability of the proposed technique was verified using simulation and experimental data. Alternating-BPI is able to estimate the power at the SoC hardware unit level with up to 98.1% accuracy. Furthermore, we demonstrate the applicability of the proposed technique on a commercial SoC and provide a fine-grain analysis of the power profiles of CPU and GPU Apps, as well as Artificial Intelligence (AI), Virtual Reality (VR) and Augmented Reality (AR) Apps. Additionally, we demonstrate that the proposed technique could be used to estimate the power consumption per-process by relying on the estimated per-unit power numbers and per-unit hardware utilization numbers. The analysis provided by the proposed technique gives useful insights about the power efficiency of the different hardware units on a state-of-the-art commercial SoC.

CCS CONCEPTS

• **Hardware → Chip-level power issues**; *Power estimation and optimization*; *Temperature monitoring*; Digital signal processing.

ICPE '22, April 9–13, 2022, Bejing, China

ACM ISBN 978-1-4503-9143-6/22/04…$15.00
https://doi.org/10.1145/3489525.3511676

KEYWORDS

Power modeling, Power characterization, Mobile SoCs, Power consumption, Power efficiency

ACM Reference Format:
Sofiane Chetoui, Michael Chen, Abhinav Golas, Farrukh Hijaz, Adel Belouchrani, and Sherief Reda. 2022. Alternating Blind Identification of Power Sources for Mobile SoCs. In *Proceedings of the 2022 ACM/SPEC International Conference on Performance Engineering (ICPE '22), April 9–13, 2022, Bejing, China.* ACM, Beijing, China, 12 pages. https://doi.org/10.1145/3489525.3511676

1 INTRODUCTION

The leaps that Systems on Chip (SoCs) have taken in the past decade led to an explosive growth of mobile devices. Mobile SoCs have become the leading product in the semiconductor industry, due to its continually improving performance and decreasing cost. Their designs have rapid design iteration, at least a new design is released each year, and new designs tend overshadow the older ones. Thus, it is important to understand the current design trends and their implications on future designs [18].

Technology scaling and heterogeneous multiprocessor designs are among the main factors enabling the performance increase [39]. The increasing processing capability has been also driven by ambitious user requirements, like performance, energy efficiency and thin form factors. This has enabled new applications that require intensive computation, raising critical power and thermal challenges [40]. For instance, the higher power density and limited cooling solutions for mobile SoCs is pushing mobile devices towards a major performance bottleneck [19]. Additionally, despite the fact that battery powered devices have become the mainly used computing devices in daily life, their battery lifetime is still one of the top usability concerns [38].

Thus, devising techniques and tools that help in profiling at a fine-grain level the existing SoCs, and the software that runs on SoCs, is a major step towards implementing efficient power and thermal management techniques, as well as, designing the architecture of the next generation Mobile SoCs.

Sensing the power and temperatures of the different hardware units in modern SoCs is a key to enable efficient and optimal power and thermal management techniques. Additionally, a fine-grain map of the power consumption of modern SoCs across different software applications would provide valuable insights to improve the performance of SoCs at the hardware and software level. However,

modern processors provide only coarse-grain power measurements of all cores using the running average power limit (RAPL) [11].

Various techniques from the literature relying on different assumptions and offering different levels of accuracy, practicality, and feasibility have been proposed [5, 7, 12, 26]. The availability of thermal sensors on a per-hardware unit basis on SoCs, makes it possible to estimate the power consumption at the unit level using blind identification techniques. Blind Power Identification (BPI) [30, 31, 33] was proposed to estimate the power at the SoC unit level, based on the individual thermal measurements and the total power consumption. However, BPI [30] relies on the thermal steady state data to estimate the power model parameters, which is hard to generate in practice and might affect the accuracy.

The reliance on certain tools and assumptions, the insufficient accuracy and the lack of practicality of the available power identification techniques has led to the lack of techniques that provide a fine-grain power measurements of modern SoCs.

Thus, in this contribution, we investigate the use of a new approach that relies on an Alternating-BPI algorithm to blindly estimate the power at the SoC unit level. We use the proposed technique to develop a plug and play tool that allows to identify the power consumption of the different SoC units. Using the proposed technique, we provide a fine-grain power analysis of a commercial SoC and provide useful insights about its power efficiency. The contributions of this paper are as follows:

- We introduce the first Alternating Blind Identification of Power sources (Alternating-BPI). The new approach allows a better accuracy and practicality than previous blind identification techniques, and works on both simulation and experimental data, as it does not require steady thermal states.
- The proposed technique substantially decreases the power estimation error, especially for heterogeneous SoCs with multiple hardware units. Simulation data has shown that the proposed technique decreases the power estimation error as low as 1.9%, as compared to 11.2% for BPI [30]. Furthermore, we show that the accuracy of the proposed technique remains stable when moving from homogeneous to heterogeneous architectures, and remains stable when the number of hardware units increase. As opposed to BPI [30] and BPISS [33], whose accuracy dropped when increasing the number of units and moving to heterogeneous architectures.
- The technique is used to design a plug and play tool, that is made publicly available [9], and that allows to estimate the power consumption of SoC units.
- The proposed technique is demonstrated using simulated and experimental data. Then it is used to characterize the power profile of several benchmarking Apps on a commercial SoC, including : CPU, GPU, Artificial Intelligence (AI) , Virtual Reality (VR), Augmented Reality (AR) Apps. The power characterization provides insights about the the power efficiency of the different hardware units on a state-of-the-art commercial SoC. Some of the insights include: (1) Even with the new integrated computing units to modern Mobile SoCs, like the GPU, the image signal processor and neural engine, the CPU is still the main source of power consumption, representing around 60% to 75% of the total SoC power. (2) The little CPU cluster plays a major role in saving power, with a power consumption that is 5x less than the big CPU cluster. (3) The GPU power consumption for AR Apps, represents only 9% of the total SoC power consumption, while the CPU represents 75% of the total SoC power consumption.
- Using the proposed technique and the hardware utilization numbers of the different processes, we show that the proposed technique could be used to get the power consumption per process. Applying the approach on AR apps, we show that the sensor fusion, tracking and mapping computation performed through the SLAM algorithm represents 53% of the total power consumption for AR apps.

The rest of the paper is organized as follows: Section 2 introduces related work and provides some background about power modeling. Section 3 describes the proposed technique and the underlying physical and mathematical concepts, and it introduces the Alternating-BPI tool. Section 4 presents the evaluation results of our technique compared against state-of-the-art techniques, as well as the power characterization of different benchmarking Apps on a commercial SoC. Section 6 concludes the paper.

2 RELATED WORK

Various studies investigated a wide range of methods for thermal and power modeling of heterogenuous SoCs. The standard approach used by these techniques try to identify the state space model that links temperature to power [3, 10, 30, 33, 37] :

$$\mathbf{t}(k) = \mathbf{A}\mathbf{t}(k-1) + \mathbf{B}\mathbf{p}(k), \tag{1}$$

where $\mathbf{t}$(k) an $\mathbf{p}$(k) are vectors that denote the temperature and power of the SoC hardware units at time k, respectively. Both $\mathbf{A}$ and $\mathbf{B}$ are two modeling matrices that capture the physical relationship between power and temperature. More precisely, the $\mathbf{A}$ matrix represents the thermal conductance matrix, which describes the natural response of the system, in the absence of power. The $\mathbf{B}$ matrix represents the forced response matrix, and it is function of the thermal capacitance and the thermal conductance matrices. Both the $\mathbf{A}$ and $\mathbf{B}$ are square matrices, whose dimension is equal to the number of power sources. In our case, the power sources are the hardware units for which we have the thermal measurements, and for which we would like to identify the power consumption. Being able to compute the state space modeling matrices would make it possible to estimate and predict the power consumption, at the same level of granularity as the available thermal measurements. In the case where thermal measurements are unavailable, and power measurements are available, being able to compute the modeling matrices would make it possible to predict the temperature, at the same level of granularity as the available power measurements.

The state space model in Equation 1 is derived from the heat diffusion equation [25], which describes the power-thermal interaction by taking into consideration the thermal conductivity, the density and the specific heat of the material. The model in Equation 1 is derived by applying a spatial discretization on the heat diffusion equation [25] , followed by a temporal discretization. There are three general approaches to identify the state space model:

- Design based approach: it is usually based on pre-silicon data and requires access to the design proprietary information, such as its layout and gate level netlist, its materials and heat sink configuration [7, 20, 26]. Thus, this approach assumes the availability of **A** and **B** and attempts to estimate **p**(k). Skadron *et al.* [35] designed an approach that models thermal behavior of the die and its package as a circuit of thermal resistances and capacitances, that correspond to functional blocks at the architecture level, which helps in capturing the physical relationship that relates power to temperature, which is similar to finding the **A** and **B** matrices. However, this is not a practical solution, since it is design specific, and it is prone to errors related to variability in the semiconductor manufacturing process. Additionally, it is computationally challenging to conduct large-scale gate-level simulation.
- Runtime based approach: it considers the processor as a gray box, and it identifies the state-space models based on physical measurements. Most of these techniques use system identification techniques and mainly rely on the existence of sensors for the power sources [3, 5, 24]. Thus, this approach assumes the availability of **p**(k) and attempts to estimate **A** and **B**. However, such fine-grain power sensors are unavailable on mobile SoCs. Another type of runtime approach, relies on the usage of infrared imaging and performance counter measurements [12, 32], however, performing infrared imaging might be invasive and prone to noisy measurements. Other runtime based techniques focus on the usage of performance counters to model the power, for instance, Min *et al.* proposed a general approach to build system power estimation models based on hardware performance counters. Karan *et al.* [34] derive functions for real time estimation of system power consumption using performance counters. Kim *et al.* [22] proposed a statistical approach for building power models using performance counters as effective proxies for x86 systems. However, all the previous techniques assume the existence of power sensors and do not perform their power modeling at the fine-grain level.
- The blind identification approach: it makes no assumption about the availability of modeling matrices and the fine-grain power sensors. This approach relies on the total power and the fine-grain thermal sensors to estimate both the modeling matrices and the power sources [30]. This technique relies on the steady state thermal data to estimate the **B** matrix, However, usually in practice it is not possible to reach steady thermal state on modern SoCs. This affects the accuracy of the model and makes it less convenient to use in practice.

The challenging task of getting fine-grain power measurements has led to the existence of only few studies that provide useful insights about the power consumption and efficiency of the different SoC hardware units while stressed by various software applications [8, 27, 33].

In this paper, we introduce the first Alternating Blind Identification technique of Power sources (Alternating-BPI). The proposed approach achieves a better accuracy than previous blind identification techniques. It is verified and used to characterize the power profile of several benchmarking Apps on a commercial SoC, including : CPU, GPU, Artificial Intelligence (AI) , Virtual Reality (VR), Augmented Reality(AR) Apps. Finally, the paper provides insights about the main sources of power consumption and the power efficiency of the different hardware units.

Algorithm 1: Alternating Blind Identification of Power Sources

Input: Temperatures **t**(k) , Total power of **p**(k)
Output: Natural Response Matrix **A**, Forced Response Matrix **B**, Power Profiles **p**(k)

1 Find the Natural Response Matrix **A** through the least square minimization:

$$\min |\mathbf{t}(k) - \mathbf{A}\mathbf{t}(k-1)|^2$$

under the constraint $\mathbf{A} \geq 0$

2 Initialize the **B** matrix :

$$\mathbf{R} = (\mathbf{J} + \mathbf{I})/3$$

$$\mathbf{B} = (\mathbf{I} - \mathbf{A})\mathbf{R}$$

where I is the identity matrix, J is an all ones matrix, and R is the thermal transfer matrix

3 **Repeat** **P**ower and **B**-Matrix estimation steps for n times:

4 **P**-*step*: Using quadratic programming find $\mathbf{p}(k) \geq 0$ such that:

$$\min \|\mathbf{B}\mathbf{p}(k) - (\mathbf{t}(k) - \mathbf{A}\mathbf{t}(k-1))\|_2$$

$$\|\mathbf{p}(k)\|_1 = \mathbf{p_{tot}}(\mathbf{k})$$

where $\mathbf{p_{tot}}(\mathbf{k})$ is the measured total power at time k

5 **B**-*step*: Using least square minimization find **B** given **p**(k) and **A** :

$$\min |\mathbf{B}\mathbf{p}(k) - (\mathbf{t}(k) - \mathbf{A}\mathbf{t}(k-1))|^2$$

6 **Runtime estimation:** Given the **A** and **B**, and the target thermal and total power data, solve the quadratic programming of the **P** step.

3 ALTERNATING BLIND IDENTIFICATION OF POWER SOURCES

3.1 The proposed approach

The proposed technique consists of two steps: *Off-line Training* step and *Runtime Estimation* step. The *Off-line Training* step needs to be run only once for each SoC, to capture the modeling matrices. The *Runtime Estimation* needs to be run each time the per-SoC unit power needs to be identified, its little computation overhead allows it to run on the device in runtime.

In the *Off-line Training* step a training data is required as input. It consists of the per-unit SoC thermal measurements and the total power consumption. The thermal measurements are generated while stressing the SoC hardware units using different patterns separated by sleep periods. This allows to capture thermal transients,

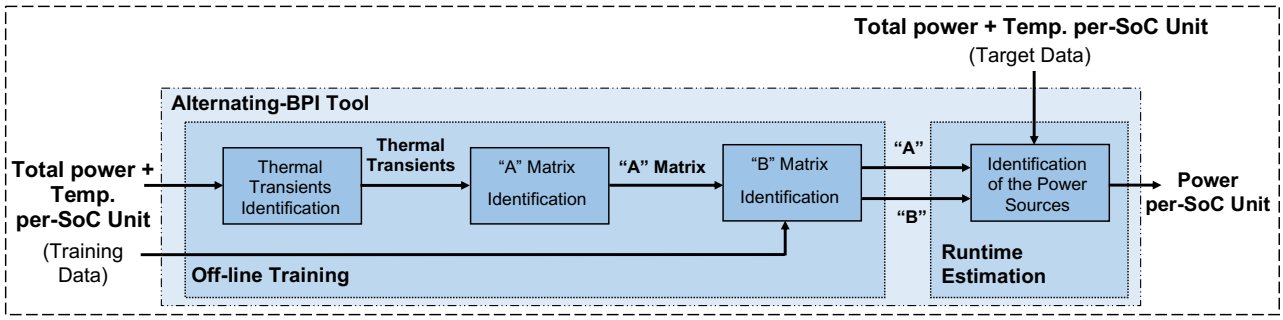

Figure 1: The Alternating-BPI tool

and the contribution of the different units to the total power. The output of the *Off-line Training* is the state space model matrices **A** and **B**. On Algorithm 1, the *Off-line Training* step is shown through lines 1 to 5, which shows how the **A** and **B** matrices of model (1) are identified.

First, the natural response matrix is estimated using the transient temperature traces. After stressing the cores with a workload, the power is forced to $\mathbf{p}(k)=0$. In practice, this could be achieved by stopping the running workload, and turning-off the target hardware units. Thus from Equation 1, we get:

$$\mathbf{t}(k) = \mathbf{A}\mathbf{t}(k-1), \tag{2}$$

Equation 2 is used to determine the **A** matrix through least square minimization, as shown in line 1 of Algorithm 1. This minimization is solved under the positivity constraint of the A matrix, since the A matrix represents the thermal conductance matrix, as explained earlier in Section 2.

Next, the goal is to make an initial guess about the **B** matrix. During a steady temperature state, there is no thermal variation, so the temperature at time k would be equal to temperature at time $k-1$, i.e. $\mathbf{t}(k) \approx \mathbf{t}(k-1) = \mathbf{t}_s$, which gives the following using Equation 1:

$$\mathbf{t}_s \approx \mathbf{A}\mathbf{t}_s + \mathbf{B}\mathbf{p}_s, \tag{3}$$

$$\mathbf{t}_s \approx (\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}\mathbf{p}_\mathbf{s}, \tag{4}$$

$$\mathbf{t}_\mathbf{s} \approx \mathbf{R}\mathbf{p}_\mathbf{s}, \tag{5}$$

where $\mathbf{t_s}$ and $\mathbf{p_s}$ represent the temperature and the power at the steady-state, and the **R** matrix represents the *steady-state thermal transfer matrix*. The previous equations help in defining the thermal transfer matrix **R** as:

$$\mathbf{R} = (\mathbf{I}-\mathbf{A})^{-1}\mathbf{B}, \tag{6}$$

based on the physical relationship between power and temperature on a multi-unit configuration, the **R** matrix should be a symmetric matrix with maximum values on the diagonal. Thus, the initialization shown in line 2 of Algorithm 1. The initial guess about the **R** matrix is then used to initialize the **B** matrix using:

$$\mathbf{B} = (\mathbf{I}-\mathbf{A})\mathbf{R}, \tag{7}$$

after initializing the **B** matrix, in line 3 to 5 of Algorithm 1 we determine the B matrix by alternating n times between two steps, n being a hyperparameter:

- **P**-*step*: estimates $\mathbf{p}(k)$, the power consumption per-SoC unit, given an initial guess of the **B** matrix. This is achieved by solving the quadratic programming optimization shown in line 4 of Algorithm 1. The optimization is solved under two constraints. The first constraint is the positivity constraint of $\mathbf{p}(k)$, since it represents power values. The second constraint ensures that the sum of the power consumption of the SoC units at time k, is equal to $\mathbf{p_{tot}}(\mathbf{k})$, the measured total power at time k. The number of unknowns in this step is equal to the number of the target SoC units times the number of timestamps.
- **B**-*step*: estimates the B matrix using least squares minimization, as shown in line 5 of Algorithm 1, given $\mathbf{p}(k)$ from the **P**-*step* and the **A** matrix from line 1 of Algorithm 1. The number of unknowns in this step is equal to number of elements of the **B** matrix, which is the square of the number of SoC units, for which the power is identified.

After identifying the modeling matrices **A** and **B**, in the *Runtime Estimation* step any thermal and power data can be given as input to estimate the power per-SoC unit. The *Runtime Estimation* step estimates the power per-SoC unit by solving the same quadratic programming as in the **P**-*step*. The user will have to only provide the temperature values per-SoC unit and the total power consumption, this data will be used along with the **A** and **B** matrices determined in the off-line training step, to determine the power consumption per-SoC unit.

3.2 The Alternating-BPI tool

The goal of the Alternating-BPI tool is to make the fine-grain power analysis under various devices seamless and straightforward. This would enable the research community with a tool that helps in providing useful insights, in order to improve modern SoCs. The Alternating-BPI tool [9] , shown in Figure 1, puts under the same

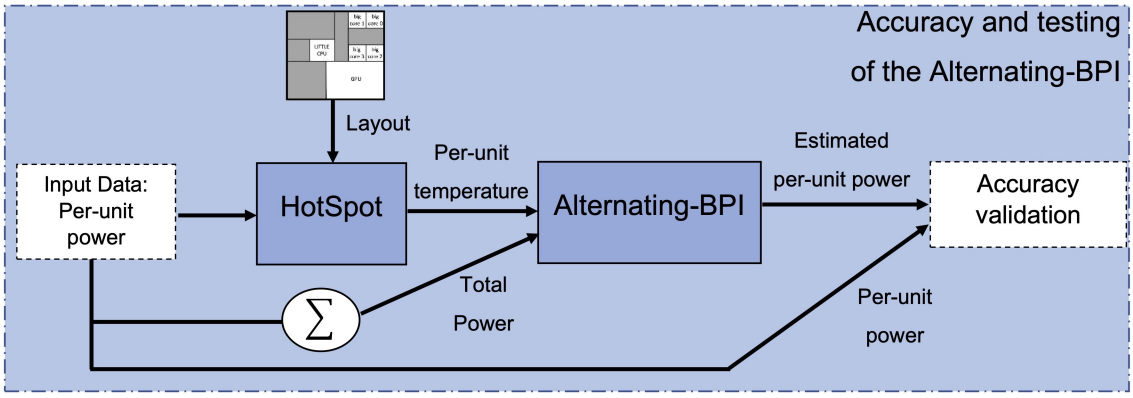

Figure 2: The verification and testing flow of the Alternating-BPI

package Algorithm 1 and the necessary data processing techniques to automate the whole process of the blind power identification.

As shown in Figure 1, the tool is composed of an *Off-line Training* step and *Runtime Estimation* step. During the *Off-line Training* the tool processes the data and estimates the **A** and **B** matrices. Then, during the *Runtime Estimation*, it estimates the power per SoC unit of any given data. The composing elements of the tool shown in Figure 1 are explained below:

- The training data: is a matrix that includes the total power consumption of the SoC on the first column, and the temperature per-SoC unit on the remaining columns. The training data should be generated by stressing the different SoC units in different patterns, separated by idle states, where a pattern is identified as a combination of active and idle SoC units. The best results are obtained when the data is collected for all the possible patterns, which requires going through all the possible configurations of active and idle SoC units. Collecting the thermal data for different patterns helps in estimating the contribution of each SoC unit to the total power, while the idle states help in creating transient thermal state data, that is used to estimate the **A** matrix.
- The target data: this data does not have to follow any specific patterns as required for the training data. The target data represents the data points for which the user wants to identify the power consumption per-SoC unit. It consists of the total power consumption and the temperature per-SoC unit.
- Thermal transients identification: the goal of this step is to identify the thermal transients on the training data. The thermal transients correspond to a natural response temperature variations that are described by Equation 2. The thermal transients are identified by tracking the thermal variation over a sliding window, if the thermal variation exceeds a certain predefined threshold, it is considered as a thermal transient. When a thermal transient is detected, the thermal data points within a predefined range are considered as part of the thermal transient trace and are saved in a matrix. This process is applied to all the training data until all the transient states are detected and recorded. The output of this step, is the thermal transient states.
- **A** Matrix identification: This step uses the thermal transients identified in the previous step to identify the **A** matrix, as shown on line 1 of Algorithm 1.
- **B** Matrix identification: This step estimates iteratively the **B** Matrix using the training data given as input, by alternating between the estimation of the **B** matrix and **p**(k), as shown on line 2 to 5 of Algorithm 1.
- Identification of the power sources: This step is about the identification of the power consumption per-SoC unit of the target data, given the **A** and **B** matrices. This corresponds to the runtime estimation step explained in Algorithm 1. This step is fast enough to generate the power values in real-time, for instance, using an Intel I5 CPU processor, it takes as low as 1.5 ms to generate the power data of 8 SoC units for one timestamp. This makes the proposed technique fast enough to be deployed to generate power predictions in real-time.

4 EXPERIMENTS AND RESULTS

4.1 Experimental setup

Simulation setup: the accuracy of the proposed technique is verified using the HotSpot thermal simulator [20]. As shown on Figure 2, for a given design layout, the HotSpot thermal simulator [20] takes as input the per-unit power traces, and produces the corresponding per-unit temperature traces. Figure 2 shows that the per-unit temperature traces, from the HotSpot simulator [20], are then taken as input by the proposed Alternating-BPI tool along with the total power. The estimated per-unit power, by the Alternating-BPI tool, is then computed. Finally, the accuracy of Alternating-BPI is computed by comparing the difference between the per-unit power traces given to HotSpot [20] as input, and the estimated per-unit power by the Alternating-BPI.

We follow the same methodology as in [31], and we choose three different floorplan benchmarks:

- 2x2 mesh : a floorplan composed of 4 units with a total maximum power budget of 80 W, and 1 cm x 1 cm as the floorplan dimensions.

- 3x3 mesh : a floorplan composed of 3 units with a total maximum power budget of 80 W, and 1 cm x 1 cm as the floorplan dimensions. This floorplan was chosen to test how the increase in the number of units could affect the per-unit power accuracy.
- big.LITTLE+GPU : as shown on Figure 3, this floorplan is composed of 6 units, with a total maximum power budget of 15 W, and 1 cm x 1 cm as the floorplan dimensions. This floorplan was chosen to test how the heterogeneity of the architecture could affect the per-unit power accuracy. Additionally, this floorplan benchmark reflects better the existing SoCs, which are mainly based on heterogeneous architectures.

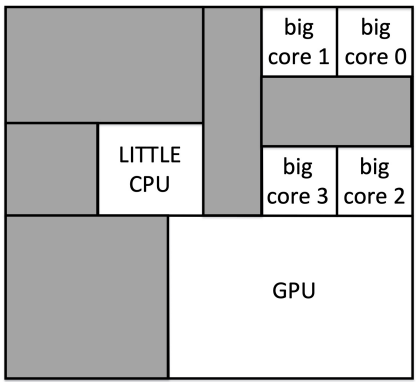

Figure 3: Layout of the big.LITTLE+GPU SoC [16] used for the testing of the Alternating-BPI

Development board setup: There is no way to verify the accuracy of the per-unit power estimation on a real device, due to the lack of per-unit power sensors. Actually, the true motivation behind the proposed Alternating-BPI technique is to provide per-unit power estimations, based on the per-unit temperature measurements, due to the non-existence of per-unit power sensors. Despite this, a validation test could still be performed on a real device, by running different workloads and contrasting the per-unit power numbers based on the hardware specification of each unit, which could validate the results. For instance, the power consumption of the GPU should be significantly higher when running a GPU benchmark, as compared to running a CPU Benchmark. Additionally, we know from the hardware specification that certain CPU cores are designed for power efficiency, while other cores are designed to provide maximum performance, based on this we would expect the power numbers of the power efficient cores to be significantly lower, after taking into consideration the hardware utilization numbers. However, the exact accuracy of the proposed technique would still be verified based on the simulation data.

Thus, the applicability of the proposed Alternating-BPI is tested on the Snapdragon-865 hardware development board [28], shown on the left side of Figure 4, which runs using the state of the art Snapdragon-865 SoC. It has a 4+3+1 CPU based on the ARM big.LITTLE architecture. More precisely, the CPU is composed of four "LITTLE" cores that are designed for energy efficient computing, they have a maximum frequency of 1.8 GHz, four "big" cores that are designed to provide maximum performance, three of them run at a maximum frequency of 2.42 GHz, and one big core that runs at a maximum frequency of 2.84 GHz [2, 29]. The SoC integrates: the Adreno 650 GPU, the Qualcomm Hexagon 698 DSP, which is referred to as CDSP in this paper, and the Qualcomm Spectra 480 image signal processor, which is referred to as SDSP. We divide the previously mentioned hardware blocks to six clusters, as shown in Table 2, for which we try to identify the power. In order to get insights that reflect the real behavior of mobile devices, the frequency is dynamically scaled by the default governor of the device during the experiments related to the power analysis .

The thermal traces are collected by reading the SoC embedded thermal sensors using a C code that runs on the device. As shown on the right side of Figure 4, we used the Monsoon HV power monitor AAA10F to measure the total power.

4.2 Results

Verification using simulation: We choose three different floorplans as benchmarks, including a big.LITTLE+GPU floorplan, shown on Figure 3, which is similar to the architecture of the Snapdragon-865 SoC used later for the experimental validation of the Alternating-BPI.

We compare against BPI [30], which is the first work to introduce a blind approach for the identification of power sources. BPI [30] relies on the non negative matrix factorization (NMF) [23] to identify the **B** matrix. However, the accuracy of the NMF [23] output is sensitive to the initialization step. Thus, we additionally compare against another version of BPI [33] that improved the initialization step by relying on the steady state temperatures, which we refer to as BPISS [33].

As shown in Table 1, the proposed technique was able to estimate the power with a better accuracy than BPI [30] and BPISS [33] for the three floorplan benchmarks. The power identification is supposed to be more difficult for a higher number of clusters, as well as, for heterogeneous systems like the big.LITTLE+GPU architecture.

As shown in Table 1, contrasting the results of the 2x2 mesh with the 3x3 mesh, we realize that as the number of units increased, the estimation error of both BPI [30] and BPISS [33] increased to 9.92% and 6.5%, respectively. On the other hand, the proposed Alternating-BPI was able to estimate the power with a better accuracy, without any significant increase in the estimation error, as the number of units increased. More precisely, the estimation error of the Alternating-BPI for the 2x2 mesh and the 3x3 mesh was 2.44% and 2.48%, respectively.

Contrasting the results of the 3x3 mesh to the big.LITTLE+GPU, we notice that as we moved to a heterogeneous design,the estimation error of both BPI [30] and BPISS [33] increased to 11.19% and 8.84%, respectively. On the other hand, the proposed Alternating-BPI was able to estimate the power with a better accuracy, as low as 1.92%, without any increase due to the heterogeneity of the architecture.

Figure 5 shows the predicted power by Alternating-BPI and BPISS [33], as compared to the actual power per-cluster for the

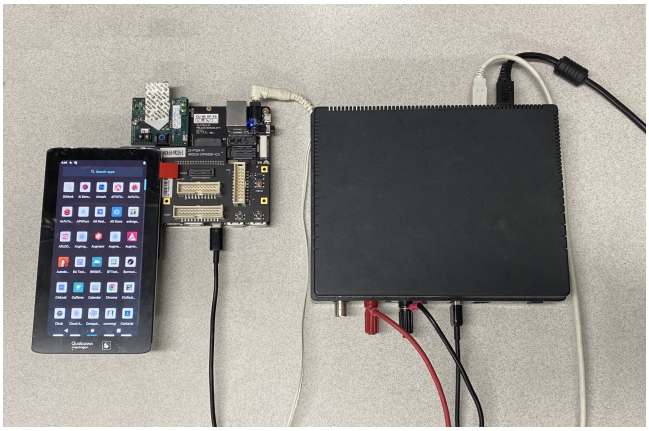

Figure 4: The used setup for the experimental verification of Alternating-BPI: the 865-HDK on the left side, and the Monsoon power monitor on the right side.

Table 1: The power estimation error of the Alternating-BPI against BPI and BPISS using three floorplan benchmarks

	BPI [30]	BPISS [33]	Alternating-BPI
2x2 mesh (4 units)	4.42 %	4.40 %	2.44 %
3x3 mesh (9 units)	9.92 %	6.5 %	2.48 %
big.LITTLE+GPU (6 units)	11.19 %	8.84 %	1.92 %

big.LITTLE+GPU floorplan benchmark. As shown, BPISS [33] ability to correctly estimate the power varies across the different power pulses and clusters. On the other hand, Alternating-BPI was able to predict with a high accuracy the power of the different clusters, and maintain such level of accuracy across the different power pulses and clusters. For instance, BPISS [33] was able to achieve a much better accuracy for Cluster 3, as compared to the estimation accuracy for Cluster 1 and Cluster 2. On the other hand, the power estimation of the proposed Alternating-BPI for the three clusters was equally accurate, while out-performing the accuracy of the other technique significantly.

The accuracy improvement of the per-unit power estimation of the Alternating-BPI, as compared to BPI [30] and BPISS [33], is mainly due to the better estimation of the **B** matrix. Both BPI [30] and BPISS [33] rely on the steady state-thermal data to estimate the **B** matrix. On the other hand, the Alternating-BPI relies on the iterative process introduced in Algorithm 1, that uses the whole data to improve the accuracy across the different iterations.

Development board testing: We used the Snapdragon-865 HDK to test the proposed technique by identifying the state space model matrices and estimating the power per cluster. As previously mentioned, we divide the SoC to 6 clusters. As shown on Table 2:

- Cluster 1: "LIT" is composed of the four little cores of the CPU. The four cores run at the same frequency and they could reach a maximum frequency of 1.80 GHz.
- Cluster 2: "Big" is composed of the first three cores of the big CPU cluster. The three cores run at the same frequency and they could reach a maximum frequency of 2.42 GHz.

Table 2: The hardware blocks composition of each cluster

	Clus. 1: LIT	Clus. 2: Big	Clus. 3: Big4	Clus. 4: GPU	Clus. 5: CDSP	Clus. 6: SDSP
Hardware Blocks	Core 1 Lit. Core 2 Lit. Core 3 Lit. Core 4 Lit.	Core 1 Big Core 2 Big Core 3 Big	Core 4 Big	GPU	CDSP	SDSP

- Cluster 3: "Big4" is composed of the fourth core of the big CPU cluster. This core runs at a different frequency than the "Big" cluster, and could reach a maximum frequency of 2.84 GHz.
- Cluster 4: "GPU" is composed of the Adreno 650 GPU and could reach maximum frequency of 587 MHz.
- Cluster 5: "CDSP" is composed of the Qualcomm Hexagon 698 DSP, which is supposed to be the neural engine of the 865 SoC.
- Cluster 6: "SDSP" the Qualcomm Spectra 480 image signal processor.

The training data for the Snapdragon-865 HDK was generated by coding software kernels that stress the different SoC units in various patterns, while collecting thermal and power data, around 80 pulses were collected. The training data was used to train BPI [30], as well as the proposed Alternating-BPI technique. Even though in practice we can not verify the accuracy of the results, analysing the per-SoC unit power traces as compared to the stress patterns helps in making a guess about the validity of the results. The BPI [30] results for 80 pulses seemed to be invalid, because of the existence of various red-flags. For instance, for the same hardware utilization, the power consumption of the little cluster was predicted by BPI [30] to be higher than the power of the Big cluster. This implies that BPI [30] needs much more data to start converging towards more reasonable results.

On the other hand, Figure 6 shows the temperature per cluster, the total power and the estimated power per-cluster, as estimated by the proposed Alternating-BPI, using 80 pulses, knowing that more data would help in making better power estimation. It has to be mentioned that a hardware SoC unit could highly affect the temperature of other units, based on their location in the layout, and the amount of power being dissipated. In the following we show an analysis of Figure 6:

- **[0 s - 500 s]:** we stress the Little, the Big, the Big4 and the GPU clusters, and we can see their estimated power profile increase accordingly. The Big cluster has the highest power profile, which makes sense, because it is composed of 3 big ARM cores. On the other hand, the little cluster shows a low power profile as compared to the big cores and the GPU, which is due to the fact that the little cores are designed to be power efficient and they run at a lower frequency.
- **[500 s - 1000 s]:** we stress only the Big cluster, and as shown in Figure 6, Alternating-BPI predicted the Big cluster power to increase to the same level as in the first 500 seconds. The minor power increase for the other clusters might be related to leakage power and some minor computation triggered by the OS.

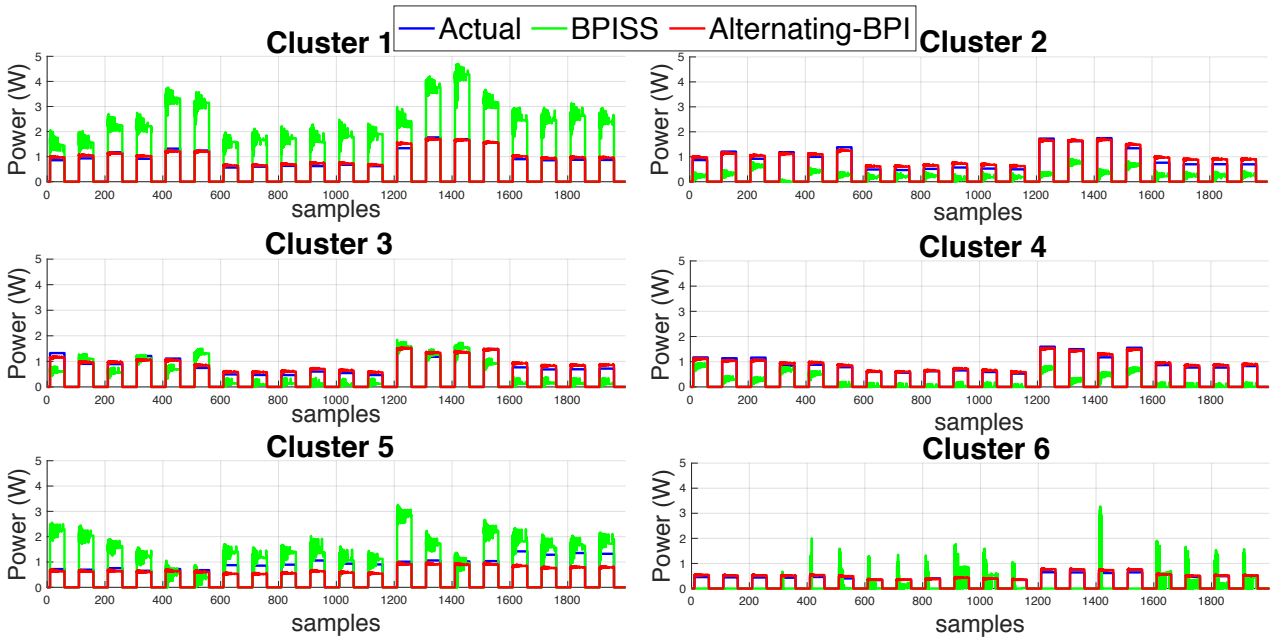

Figure 5: The accuracy of BPISS vs Alternating-BPI in predicting the power per-cluster for the big.LITTLE+GPU floorplan

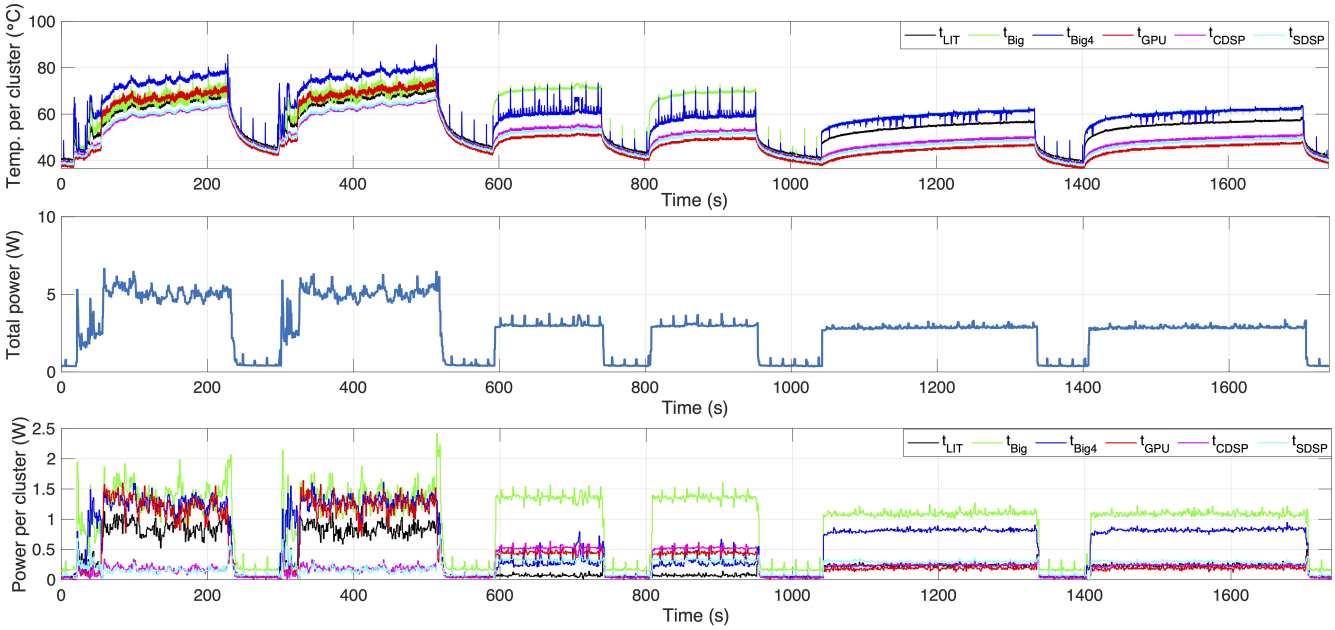

Figure 6: The Alternating-BPI estimated power per-SoC Unit of the Snapdragon-865

- **[1000 s - 1700 s]:** we stress the Big and the Big 4 clusters, and as shown in Figure 6, Alternating-BPI was able to predict that the highest power dissipation is coming from the stressed clusters.

Per-unit power characterization of mobile apps: we perform a fine-grain power characterization of various mobile Apps on the Snapdragon-865 SoC using Alternating-BPI. The used benchmarking apps and the clusters they use are shown on Table 3. The chosen list includes : CPU, GPU, Virtual Reality (VR), Artificial Intelligence (AI) and Augmented Reality (AR) Apps. These benchmarks stress all the clusters in different patterns and different utilization levels. The bar graph in Figure 6 shows the average power per cluster, per benchmarking app:

Table 3: The clusters used by the set of benchmarking apps

	Clus. 1: LIT	Clus. 2: Big	Clus. 3: Big4	Clus. 4: GPU	Clus. 5: CDSP	Clus. 6: SDSP
Geekbench [15]	Yes	Yes	Yes	No	No	Yes
3DMark [1]	Yes	Yes	No	Yes	No	No
VRMark [36]	Yes	Yes	No	Yes	No	No
AI Benchmark [4]	Yes	Yes	Yes	No	Yes	Yes
AR Civilisations [17]	Yes	Yes	Yes	Yes	No	Yes

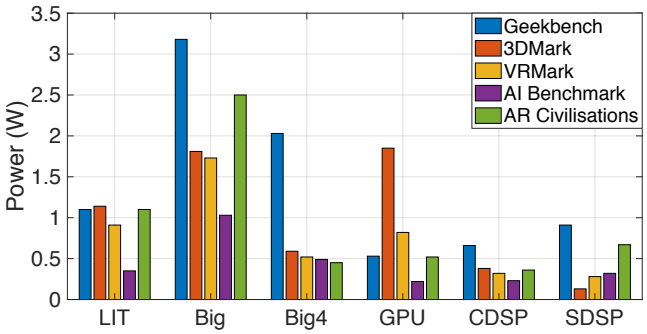

Figure 7: The Alternating-BPI estimated power per-SoC Unit of the Snapdragon-865 for the benchmarking apps

- **Geekbench [15]:** includes 25 multi-threaded workloads of four different sections: cryptography, integer, floating point and memory workloads. As shown in Figure 7, the Big and Big4 clusters consume up to 5 Watts, as compared to less than 3 watts for most of the other benchmarking apps. Geekbench relies as well on the little cluster and the SDSP with a combined power consumption of 2 watts. The results show that the little cluster, which is designed for low-power computing, consumes 5x less power than the Big and Big4 clusters. Additionally, the Big4 cluster consumes up to 2x more power than the four cores of the little cluster, which is due to the power-hungry architecture and high operational frequency, that can reach 2.84 GHz. Figure 8 shows that the power consumption of the CPU (Little + Big + Big4 clusters) account for more than 75% of the power consumption of the Snapdragon-865 SoC.
- **3DMark [1]:** is a GPU-CPU intensive benchmark that tries to mimic gaming apps. As shown in Figure 7, the main power goes to the GPU and the Big clusters, with a combined power of 3.6 Watts, representing more than 65% of the total power consumption. Additionally, even when the GPU is highly utilized, its power consumption is 2.5x less than the power consumption of the Big + Big4 clusters. Thus, the CPU Big cluster remains the biggest source of power consumption in the SoC.
- **VRMark [36]:** is a virtual reality benchmark that is mainly CPU intensive. Figure 7 shows that VRMark [36] has the same power profile as 3DMark [1], except for the GPU that consumes 2x less power the VR benchmark, as compared to the GPU benchmark. Figure 8 shows that for VRMark more than 65% of the power consumption is coming from the CPU.
- **AI Benchmark [4]:** runs 46 AI computer vision tests, that mainly run on the CPU. Figure 7 shows that AI Benchmark [4] has lowest power profile amongst the benchmarking Apps, with 3x less power consumption than Geekbench on the CPU. Figure 8 shows that 70% of the power consumption is coming from the CPU.
- **AR Civilisations [17]:** is an Augmented Reality (AR) educational app. Figure 7 shows that the AR App is CPU-hungry, with up to 3 watts consumed by the Big + Big4 clusters, which represents 50% of the total power consumption. Even if it is an AR App, Figure 8 shows the GPU represents only 10% of the total power consumption, while the little cluster, which is designed for low-power computing, consumes up to 20% of the total power consumption. This should be related the low 3D rendering utilization of AR Apps, as compared to Gaming Apps.

The main insights that Figures 7 and 8 show are:

- **The CPU remains the main source of power consumption**: even with the new computing units that have been integrated to modern Mobile SoCs, like the GPU, the image signal processor and neural engine, the CPU is still the main source of power consumption, representing around 60% to 75% of the total power for CPU, GPU, VR, AI or AR Apps.
- **The little cluster plays a major role in saving power**: the four cores of the little cluster are highly utilized by most Apps, yet the power consumption of the little cluster is 5x less than the Big and Big4 clusters. The inclusion of more than four cores, based on the LITTLE cores architecture, would strongly enable power efficient computing.
- **The CPU frequency boost has a very low power efficiency**: the 20% frequency boost of the Big4 core, as compared to the cores of the Big cluster, increases the power consumption of the Big4 core by almost 2x. Mathematically, this makes sense because the dynamic power is proportional to the voltage square multiplied by the frequency, knowing that the frequency and the voltage on Mobile SoCs are scaled dependently of one another, and this dependence is approximately linear. Thus, the dynamic power is proportional to the cubic of the frequency [21]. This makes the frequency boost highly costly from a power consumption perspective, and makes the power efficiency drop drastically, because a 20% frequency boost will not translate to 2x performance boost.
- **The Augmented Reality Apps have a high CPU power profile and a low GPU power profile**: the Augmented Reality (AR) App shows an average CPU power consumption of 4.5 Watts, which accounts for 75% of the total power consumption of the SoC. On the other hand, The GPU which runs the 3D rendering, accounts for only 9% of the total power consumption.
- **The GPU is highly power efficient**: 3DMark [1] is mainly about 3D rendering and the GPU handles most of its computation, however, it is average power consumption is only 1.85 Watts, which represents only 30% of the total power consumption, and which is 2x less than the total CPU power

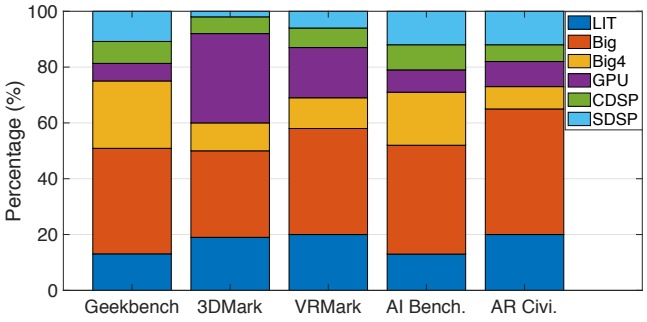

Figure 8: The Alternating-BPI estimated percentage power consumption of per-SoC Unit of the Snapdragon-865 for the benchmarking apps

when running GPU workloads, and 3.5X less than the total CPU power when running CPU workloads.

Per-process power characterization: the enabled power characterization at the hardware block level, using the proposed technique, could be used to take the power characterization to even a more fine grain level. More precisely, estimating the power consumption per process running on the device. The per-unit power characterization allows to give insights that are more hardware oriented. For instance, estimating the power efficiency of a certain SoC unit as compared to other units. On the other hand, the per-process power characterization allows to give insights that are more software oriented. For example, it allows to identify the most power hungry processes and to improve their efficiency.

Estimating the per-process power consumption could be achieved by measuring the hardware utilization per-block of the different processes, then partitioning the per-block power values across the different processes, based on the utilization numbers. For instance, if a certain process is taking 30% of the hardware utilization of a certain unit, while a second process is taking 70% utilization, then the power consumption of the first process should represent 30% of the total power consumption of the unit, while the power consumption of the second process should account for 70% of it. It is worth mentioning that the utilization that is being referred to in here, is the normalized utilization, which takes into account both the frequency and the utilization of the corresponding unit. More precisely, the normalized utilization is computed by multiplying the current frequency by the current utilization and dividing by the maximum possible frequency of the unit.

In order to demonstrate this, we collect the hardware utilization of the different Augmented Reality processes, using Snapdragon Profiler, while running AR Civilisation on the 865 HDK. Using the Alternating-BPI we get the power consumption per hardware block, and then we partition the power across the different tasks. The processes in question are:

- Operating system processes: this includes all system related threads. Most of these threads are kernel threads, and they usually run on the little cluster and the Big4 core, taking around 19% of the little cluster utilization, and roughly 1% of the Big4 core utilization.

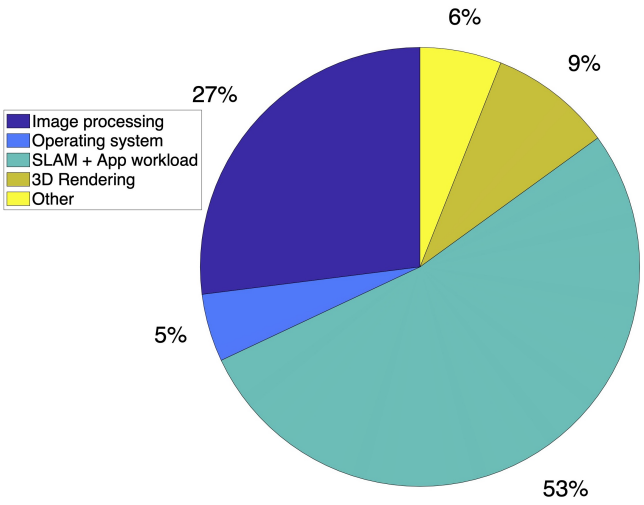

Figure 9: The percentage power consumption of the Augmented Reality processes, while running AR Civilisation on the 865 HDK

- Image processing processes: this includes all the threads that handle the camera sensor readings, and the image filtration and calibration. These processes usually run on the little cluster and the SDSP. They take around 54% of the utilization of the little cluster, and 20% of the utilization of the SDSP.
- SLAM + App workload processes: these threads are related to Simultaneous localization and mapping (SLAM) algorithm, which includes computer vision and object tracking, positional and GPS data reading, synchronization and sensor fusion. These threads usually run on the Big cluster, taking around 26% of its utilization, and on the Big4 core, taking 10% of its utilization in average.
- 3D Rendering processes: this simply include the processes that run on the GPU to execute the 3D rendering the Augmented Object. This usually takes around 15% of the GPU utilization in the case of AR Civilisation.
- Other: this includes all other threads that are not part of the previously mentioned processes, and that get triggered randomly.

Figure 9 shows the percentage power consumption of each Augmented Reality process of AR Civilisation:

- Operating system processes: consume around 5% of the total power, which is mainly due to the fact that the system threads run on the little cluster, making it power efficient to run the system tasks.
- Image processing processes: consume around 27% of the total power, which is the second highest power hungry task of the AR pipeline. The image processing is a substantial component of the total power in AR civilisation, because the App relies constantly on the camera feed. This power is mainly consumed on the little cluster and the SDSP.
- SLAM + App workload processes: are the most power hungry processes, representing around 53% of the total power consumption. The localization and mapping computation is

quite intensive, and many papers from the literature have been trying to improve the computational efficiency of SLAM algorithms [6, 13, 14]. Most of the SLAM power is consumed on the Big cluster cores.

- 3D Rendering processes: consumes only 9% of the total power consumption. This makes sense, because the hardware utilization of the GPU is pretty low when using AR Civilisation, which is due to the fact that only the AR object is being rendered, while most pixels are just coming from the camera feed, as opposed to 3D games or Virtual Reality Apps, where the whole scene is being rendered.

These insights could be used to improve the power efficiency of Augmented Reality Apps by focusing on the most power hungry tasks, for instance, one direction could be to off-load the SLAM algorithm to the cloud.

Most importantly, the previous experiments demonstrate that the proposed tool could be used to conduct experiments and investigations to characterize the power consumption at a fine grain-level, which is an important step towards making, both the hardware and the software, power-efficient.

5 ACKNOWLEDGEMENT

This research was supported by Meta Platforms.

6 CONCLUSION

Mobile SoCs lack the ability to sense the power at SoC unit level. The existing power identification techniques rely on certain assumptions, and they lack accuracy and practicality, which makes it challenging to get useful insights about the fine-grain power profiles of mobile SoCs, knowing that such insights are critical for the design and improvement of these SoCs. In this work we proposed a new technique for blind identification of power sources. The technique relies on an Alternating-BPI approach, which allows a faster convergence, a better accuracy and practicality than previous blind identification techniques, as it does not require steady thermal states. The proposed approach decreases the estimation error to 1.9%, as compared to 11.2% for BPI [30]. The accuracy of proposed work was verified using simulation and validated using experimental data on a commercial development board. Additionally, the new approach was used to characterize the per-unit power profile of several bencharmaking Apps on a commercial SoC, including : CPU, GPU, Artificial Intelligence (AI) , Virtual Reality (VR), Augmented Reality (AR) Apps. The power characterization provides insights about the main sources of power consumption and the power efficiency of the different hardware units. Some of the insights include: (1) Even with the new computing units integrated to modern Mobile SoCs, the CPU is still the main source of power consumption, representing around 60% to 75% of the total SoC power. (2) The little CPU cluster plays a major role in saving power, with a power consumption that is 5x less than the big CPU cluster. (3) The GPU power consumption for AR Apps, represents only 9% of the total SoC power consumption, while the CPU represents 75% of the total SoC power consumption. Furthermore, using the hardware utilization of the different processes and the per-unit power provided by the proposed technique, we perform a per-process power characterization of the different processes of an Augmented Reality application, and we show the percentage power consumption of each process. Finally, the technique is used to design a plug and play tool, that is made publicly available [9] , and that allows to estimate the per-unit power consumption.

REFERENCES

[1] Benchmark 3DMark. 2018. 3DMark GPU Benchmark. https://www.3dmark.com/

[2] Android Authority. 2019. Qualcomm Snapdragon 865 specs: Everything you need to know. https://www.androidauthority.com/qualcomm-snapdragon-865-specs-1058483/

[3] Andrea Bartolini, Matteo Cacciari, Andrea Tilli, and Luca Benini. 2011. A distributed and self-calibrating model-predictive controller for energy and thermal management of high-performance multicores. In *2011 Design, Automation & Test in Europe*. IEEE, 1–6.

[4] AI benchmark. 2020. AI Benchmark. https://ai-benchmark.com/

[5] Francesco Beneventi, Andrea Bartolini, Andrea Tilli, and Luca Benini. 2012. An effective gray-box identification procedure for multicore thermal modeling. *IEEE Trans. Comput.* 63, 5 (2012), 1097–1110.

[6] Kevin Brink, Ryan Sherrill, Jamie Godwin, Jincheng Zhang, and Andrew Willis. 2020. Maplets: An Efficient Approach for Cooperative SLAM Map Building Under Communication and Computation Constraints. In *2020 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. IEEE, 367–374.

[7] David Brooks, Vivek Tiwari, and Margaret Martonosi. 2000. Wattch: A framework for architectural-level power analysis and optimizations. *ACM SIGARCH Computer Architecture News* 28, 2 (2000), 83–94.

[8] Huixiang Chen, Yuting Dai, Hao Meng, Yilun Chen, and Tao Li. 2018. Understanding the characteristics of mobile augmented reality applications. In *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. IEEE, 128–138.

[9] Sofiane Chetoui. [n. d.]. SCALE lab tools. https://scale.engin.brown.edu/software/

[10] Ryan Cochran and Sherief Reda. 2010. Consistent runtime thermal prediction and control through workload phase detection. In *Design Automation Conference*. IEEE, 62–67.

[11] Howard David, Eugene Gorbatov, Ulf R Hanebutte, Rahul Khanna, and Christian Le. 2010. RAPL: Memory power estimation and capping. In *2010 ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)*. IEEE, 189–194.

[12] Kapil Dev, Abdullah Nazma Nowroz, and Sherief Reda. 2013. Power mapping and modeling of multi-core processors. In *International Symposium on Low Power Electronics and Design (ISLPED)*. IEEE, 39–44.

[13] Taosha Fan, Hanlin Wang, Michael Rubenstein, and Todd Murphey. 2020. CPL-SLAM: Efficient and certifiably correct planar graph-based SLAM using the complex number representation. *IEEE Transactions on Robotics* 36, 6 (2020), 1719–1737.

[14] Qiang Fu, Hongshan Yu, Xiaolong Wang, Zhengeng Yang, Hong Zhang, and Ajmal Mian. 2020. FastORB-SLAM: A

fast ORB-SLAM method with Coarse-to-Fine descriptor independent keypoint matching. *arXiv preprint arXiv:2008.09870* (2020).

[15] geekbench. 2020. Geekbench 4. https://www.geekbench.com/geekbench4/

[16] Young-Ho Gong, Jae Jeong Yoo, and Sung Woo Chung. 2017. Thermal modeling and validation of a real-world mobile ap. *IEEE Design & Test* 35, 1 (2017), 55–62.

[17] Google. 2020. Civilisations AR. https://play.google.com/store/apps/details?id=uk.co.bbc.civilisations&hl=en_US&gl=US

[18] Matthew Halpern, Yuhao Zhu, and Vijay Janapa Reddi. 2016. Mobile CPU's rise to power: Quantifying the impact of generational mobile CPU design trends on performance, energy, and user satisfaction. In *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 64–76.

[19] Yin Hang and Hussameddine Kabban. 2015. Thermal management in mobile devices: challenges and solutions. In *2015 31st Thermal Measurement, Modeling & Management Symposium (SEMI-THERM)*. IEEE, 46–49.

[20] Wei Huang, Shougata Ghosh, Sivakumar Velusamy, Karthik Sankaranarayanan, Kevin Skadron, and Mircea R Stan. 2006. HotSpot: A compact thermal modeling methodology for early-stage VLSI design. *IEEE Transactions on very large scale integration (VLSI) systems* 14, 5 (2006), 501–513.

[21] Jae Min Kim, Young Geun Kim, and Sung Woo Chung. 2013. Stabilizing CPU frequency and voltage for temperature-aware DVFS in mobile devices. *IEEE Trans. Comput.* 64, 1 (2013), 286–292.

[22] Yeseong Kim, Pietro Mercati, Ankit More, Emily Shriver, and Tajana Rosing. 2017. P 4: Phase-based power/performance prediction of heterogeneous systems via neural networks. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. IEEE, 683–690.

[23] Daniel D Lee and H Sebastian Seung. 1999. Learning the parts of objects by non-negative matrix factorization. *Nature* 401, 6755 (1999), 788–791.

[24] Duo Li, Sheldon X-D Tan, Eduardo H Pacheco, and Murli Tirumala. 2010. Parameterized architecture-level dynamic thermal models for multicore microprocessors. *ACM Transactions on Design Automation of Electronic Systems (TODAES)* 15, 2 (2010), 1–22.

[25] A Munier, JR Burgan, J Gutierrez, E Fijalkow, and MR Feix. 1981. Group transformations and the nonlinear heat diffusion equation. *SIAM J. Appl. Math.* 40, 2 (1981), 191–207.

[26] Farid N Najm. 1995. Power estimation techniques for integrated circuits. In *Proceedings of IEEE International Conference on Computer Aided Design (ICCAD)*. IEEE, 492–499.

[27] Edson Luiz Padoin, Laércio Lima Pilla, Márcio Castro, Francieli Z Boito, Philippe Olivier Alexandre Navaux, and Jean-François Méhaut. 2015. Performance/energy trade-off in scientific computing: the case of ARM big. LITTLE and Intel Sandy Bridge. *IET Computers & Digital Techniques* 9, 1 (2015), 27–35.

[28] Snapdragon Qualcomm. 2020. Snapdragon 865 Mobile Hardware Development Kit. https://developer.qualcomm.com/hardware/snapdragon-865-hdk

[29] Robin Randhawa. 2013. Software Techniques for ARM big. LITTLE Systems. *ARM, Apr* (2013).

[30] Sherief Reda and Adel Belouchrani. 2017. Blind identification of power sources in processors. In *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2017*. IEEE, 1739–1744.

[31] Sherief Reda, Kapil Dev, and Adel Belouchrani. 2017. Blind identification of thermal models and power sources from thermal measurements. *IEEE Sensors Journal* 18, 2 (2017), 680–691.

[32] Sherief Reda, Abdullah N Nowroz, Ryan Cochran, and Stefan Angelevski. 2013. Post-silicon power mapping techniques for integrated circuits. *Integration* 46, 1 (2013), 69–79.

[33] Mostafa Said, Sofiane Chetoui, Adel Belouchrani, and Sherief Reda. 2018. Understanding the Sources of Power Consumption in Mobile SoCs. In *2018 Ninth International Green and Sustainable Computing Conference (IGSC)*. IEEE, 1–7.

[34] Karan Singh, Major Bhadauria, and Sally A McKee. 2009. Real time power estimation and thread scheduling via performance counters. *ACM SIGARCH Computer Architecture News* 37, 2 (2009), 46–55.

[35] Kevin Skadron, Mircea Stan, Marco Barcella, Amar Dwarka, Wei Huang, Yingmin Li, Yong Ma, Amit Naidu, Dharmesh Parikh, Paolo Re, et al. [n. d.]. Hotspot: Techniques for modeling thermal effects at the processor-architecture level. Citeseer.

[36] VRMark VRMark. 2018. https://benchmarks.ul.com/vrmark

[37] Yefu Wang, Kai Ma, and Xiaorui Wang. 2009. Temperature-constrained power control for chip multiprocessors with online model estimation. *ACM SIGARCH computer architecture news* 37, 3 (2009), 314–324.

[38] Kaige Yan, Xingyao Zhang, and Xin Fu. 2015. Characterizing, modeling, and improving the QoE of mobile devices with low battery level. In *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 713–724.

[39] Geoffrey Yeap. 2013. Smart mobile SoCs driving the semiconductor industry: Technology trend, challenges and opportunities. In *2013 IEEE International Electron Devices Meeting*. IEEE, 1–3.

[40] Ying-Ju Yu and Carole-Jean Wu. 2017. Understanding the thermal challenges of high-performance mobile devices with a detailed platform temperature model. In *2017 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 122–123.