Run-time Power Estimation for Mobile and Embedded Asymmetric Multi-Core CPUs

Mathew J. Walker, Anup K. Das, Geoff V. Merrett, Bashir M. Al-Hashimi
School of ECS, University of Southampton, Southampton SO17 1BJ, UK
{mw9g09,a.k.das,gvm,bmah}@ecs.soton.ac.uk

ABSTRACT

Significant energy savings can be made in mobile devices using intelligent run-time management software (RTM) that can effectively trade off performance and energy. For the RTM to make informed decisions, it needs accurate and responsive run-time knowledge of the power consumption. For desktop and server systems, hardware performance monitoring counters (PMCs) have been successfully used for run-time power estimation. However, for mobile devices there is little research on the topic. This paper presents a PMC-based power model for mobile devices and the associated methodology for identifying the best PMCs and their relationship with power consumption. However, for the majority of mobile devices experimented with, PMCs were difficult or impossible to obtain and so a utilisation-based power model was proposed as a more practical alternative. Both these models are validated, respectively, on an ARM Cortex-A8 and an asymmetric ARM big.LITTLE architecture, which can be found in recent tablets and smartphones, such as the Samsung Galaxy S5. A key feature of the utilisation-based model is that it can derive the power consumed per core and per individual task. Furthermore, the model can predict the likely power consumption if a task currently executing on a particular core/frequency (e.g. Cortex-A7 at 400 MHz) were to be migrated to a different core/frequency (e.g. Cortex-A15 at 1200 MHz). This allows an RTM to explore the consequences of changing to any frequency/core combination in advance. By comparing the two models, it was found that while the PMC model achieves greater accuracy (3.2%), the utilisation model still achieves an impressive accuracy of 5.6% and 7.2% on a Cortex-A7 and Cortex-A15, respectively. As it has a high accuracy and can be easily implemented on all mobile devices, we propose that the utilisation model represents an attractive choice for guiding an RTM to improved energy efficiency.

1. INTRODUCTION

There is an ever-increasing demand for greater computational performance at lower energy costs in modern processors. Gains in performance and power efficiency not only allow applications to run faster while consuming less energy, but they enable innovative and new applications that may not have even been considered beforehand. For example, playing 3D games, having a desktop-class web-browser, and recording high-definition video was not possible on mobile devices 10 years ago.

Historically, processor performance improvement has come about by increasing the clock rate in each new generation. However, the increasing heat density has hampered this, causing the clock rate of processors for desktop and server systems to stall in the last decade. Mobile processors are reaching these speeds already, for example, the Samsung Galaxy S5 has a maximum CPU frequency of 2.5 GHz[15]. Processor designers have been forced to turn to other alternatives in order to increase performance, for example moving towards 64-bit architectures and multi-core processors. However, efficiently allocating tasks between cores to achieve the required performance and energy efficiency is a difficult undertaking that changes with many other factors; such as the user's quality of experience (QoS) requirement and the current battery level. The concept of a run-time manager (RTM) has been recently proposed for improving the energy efficiency of systems. The RTM makes intelligent decisions based on the current requirements and operating conditions and controls the energy-saving techniques of the CPU. For an RTM to be truly effective in finding the optimum trade-off point between performance and power consumption, it should have accurate and responsive per-core, and ideally per-task, power estimations.

A power estimation method for an RTM needs to be light-weight, accurate, responsive and able to provide data 'on-the-fly'. For desktop and server applications there has been much research published on using hardware performance monitoring counters (PMCs) to estimate the run-time power [16, 7, 3, 14, 5, 6]. PMCs are special registers in the CPU that count the occurrence of certain hardware-related events, such as L1 cache misses and branch mispredictions. Their purpose is for performance analysis but, because they provide detailed usage information on sub-architectural units, they can also be effective at estimating power. Accessing and reading PMCs requires very little overhead in contrast to software profilers which are generally not suitable for real-time use on a live system. Most methods consist of running workloads in order to exercise the CPU while measuring the power and then choosing PMCs that correlate highly to build multiple linear regression models. Choosing the PMCs to use based on their correlation with power is not an effective solution as demonstrated in Section 2.2.

While PMCs are easily obtainable on desktop and server systems, they are very difficult or impossible to obtain on mobile and embedded systems, which explains the lack of research with PMCs on mobile devices. Two exceptions are a recent paper by Rethinagiri et al. [13] that developed a PMC model for ARM cores and is discussed further in Section 2 and a paper by Nunez-Yanez et al. [12] that uses

implantation data from the RTL to built an accurate power model.

This paper presents a PMC power model for an embedded platform on which it was possible to obtain PMCs and the methodology, including two successful methods of obtaining PMCs, is discussed in detail. Unlike other works discussed previously, this work measures power directly from the CPU, uses a high sampling frequency and is shown to be responsive by validating with real-time data.

Unfortunately, the fact that PMCs are very difficult or impossible to access on most mobile and embedded platforms makes PMC models undesirable for this purpose. This raised the question of whether a simpler metric, such as CPU utilisation, which provides less information but is easily accessible on many platforms, can be a suitable alternative. In order to answer this question, a model using CPU utilisation to predict the run-time power was built, thoroughly tested on a variety of workloads and contrasted with the PMC-based model. While it did not achieve the very high levels of accuracy of the PMC model, it still achieved a very low average error of 5.6% and 7.2% on an ARM Cortex-A7 and Cortex-A15, respectively. Unlike the PMC-based models, it can be implemented easily on any mobile device. The utilisation model is also able to determine how each running process and each core contributes towards the total power consumption and can also predict how much power would be consumed if it were running at a different frequency or a different core type in a big.LITTLE architecture. This is further discussed in Section 3.

The key contributions of this work are:

- An embedded PMC-based power model and the associated methodology (including details of obtaining PMCs) is presented;
- The problem of implementing PMC models in mobile and embedded devices is highlighted and utilisation based models are shown to be an adequate alternative that can be easily implemented in any device;
- A utilisation-based model of a big.LITTLE architecture is presented that allows the power contribution of each task or core to be analysed and the power a task would consume if it were running of a different core to be estimated from the current execution profile;
- The models are thoroughly validated on a wide range of workloads.

2. PMC-BASED POWER MODEL

2.1 Experimental Setup

A PMC power model was first built for an ARM mobile CPU. The experiments were conducted on a BeagleBoard-xM [4] open-source single-board computer. It has a Texas Instruments DM3730 System-on-Chip (SoC) [17] which is similar to the OMAP35x family of SoCs. The DM3730 contains an ARM Cortex-A8 single-core processor, with a maximum clock rate of 1 GHz. It has a 32 kB L1 instruction cache, an 80 kB L1 data cache, a 64 kB L2 instruction cache and a 32 kB L2 data cache. It also contains an Imagination Technologies SGX GPU and a Texas Instruments TMS320C64x+ DSP. For this work, only the power consumption of the Cortex-A8 was considered.

Unlike the vast majority of related research, the power to the individual CPU was measured directly as opposed to the power of the whole board. This was achieved by lifting an inductor off of the board and re-routing the signal through

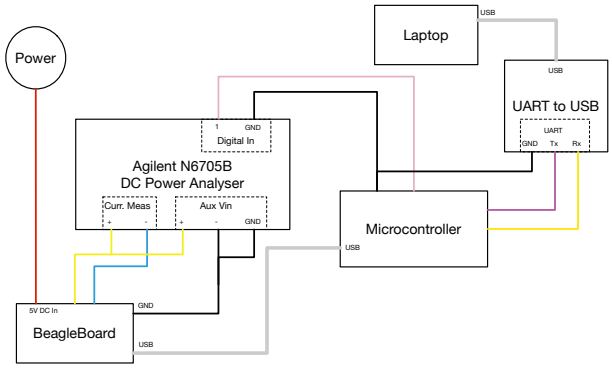

Figure 1. Experimental setup used for characterising and validating the PMC-based power model

the same inductor and an Agilent N6705 Power Analyser [1]. The voltage supplied to the Cortex-A8 was also measured to confirm the DVFS level and to calculate the power consumed. The voltage and current were both recorded every 10 ms. The PMCs were read and stored by custom software running on the board itself. This software triggered the power analyser at the correct time by sending signal to a microcontroller, which in turn sent a signal to the power analyser. This microcontroller also allowed the data to be forwarded to a laptop for storing data and remote monitoring of the experiment. Figure 1 shows a simple block diagram of the experimental setup.

A variety of workloads, including many from the MiBench Benchmark suit [9], video playback and custom workloads designed to exert specific architectural features were used to exercise the processor while data was collected.

Accessing the performance counters proved problematic. While there is clear support for performance counters in ARM's IP, some of the signals that enable the operation of performance counters are dealt with outside of the core itself and their settings depend on the implementation (i.e. by Texas Instruments). To enable User mode access to the Performance Monitor Registers the USEREN had to be written to from the kernel. To do this, a Loadable Kernel Module (LKM) was written and it successfully allowed access to the required registers; the cycle counter could be enabled and read from without problems. The performance counter registers could be written to and read from but the values of the actual counters remained at 0 because the DBGEN register was not set. Two solutions to this were found and both were used for this research. The first method exploited the fact that the DBGEN register is enabled when debugging using JTAG. A Spectrum Digital XDS100v2 JTAG debugger was used with a custom-derived method to enable the DBGEN register when the operating system was running on the CPU. This method was useful as it worked with any unmodified version of the Linux kernel that ran on the BeagleBoard. This method is, however, time consuming, sometimes unreliable, and occasionally results in a corrupted SD card. For this reason, another method was also used where code was written into the kernel itself that accessed the performance counters (without DGBEN having to be set as it was being accessed from code within the kernel) and made them available to the user space via *sysfs* (a virtual file system).

2.2 Model Generation and Validation

Only four PMCs on the Cortex-A8 can be monitored simultaneously. Therefore the experiment was run multiple times to capture the majority of the PMCs (including those relating to the NEON SIMD coprocessor). Once data had been

captured for every performance counter the correlation of each one with the power consumption was calculated. The counter with the highest correlation gave the number of operations issued and had a correlation of 0.71. A model was built with this counter alone as a base model for deriving the best three accompanying event counters. The workloads were broken into sections and the best events were identified by how well they improved this base model across a variety of workloads. The method chose the additional PMCs with a low amount of inter-correlation so that information from the counters would not be repeated and the model is able to work with a wider variety of workload types. To confirm and evaluate this method, an *R* script was created that conducted an exhaustive search that tried building models with every combination of four PMCs. It confirmed the selection of the following events from the BeagleBoard shown in Figure 1.

- Event 4 (E4), Data causes cache access (correlation: 0.53)
- Event 80 (E80), L1 instruction cache access (correlation: 0.52)
- Event 77 (E77), NEON cacheable access for L1 data cache (correlation: 0.17)
- Event 85 (E85), Number of operations issued (correlation: 0.71)

It can be clearly seen by these correlation values that the best performance counters to choose are not necessarily the ones with the highest correlation with power. For example, E77 only occurs when the NEON SIMD processor is used and therefore has a low overall correlation with power. However, when the NEON SIMD processor is used, it has a large energy footprint, resulting in a significant increase in power consumption. The model presented in [13] does not consider the NEON processor and therefore will not accurately track the power consumption when workloads use it.

Once the events to be monitored with PMCs had been chosen, the experiment was then re-run with that combination of events and the model parameters were then obtained by taking the resulting count values and attempting to predict the measured power with multiple linear regression. A model for every frequency (300 MHz, 600 MHz, 800 MHz and 1 GHz) was developed. The model was verified with real-time data and not simply by comparing the average power and average modelled power over the whole duration of the workload, which is the case for the vast majority of related research, including the paper on mobile PMC models [13]. This is a fairer method of validation as it not only tests its accuracy at predicting the average power for a workload, but it tests the responsiveness of the model and how well it can predict small spikes and phase changes. Figure 2 shows a power trace of the system running several workloads over a period of 60 seconds and the corresponding estimated power from the model. Notice how even the small spikes and sub-workload characteristics are accurately tracked. For a small error value the model must be very reactive; a key requirement for real-world implementation with a run-time management system. The average error across every workload and every frequency was less than 1.91% for the workloads that were used to train the model. Other unseen workloads were tested with the model and the workload that caused the highest error was video decoding which resulted in an average error of 3.2%. Equations 1 to 4 are the linear power model equations. The PMC events (e.g. E85) have the units *counts per 10 ms*.

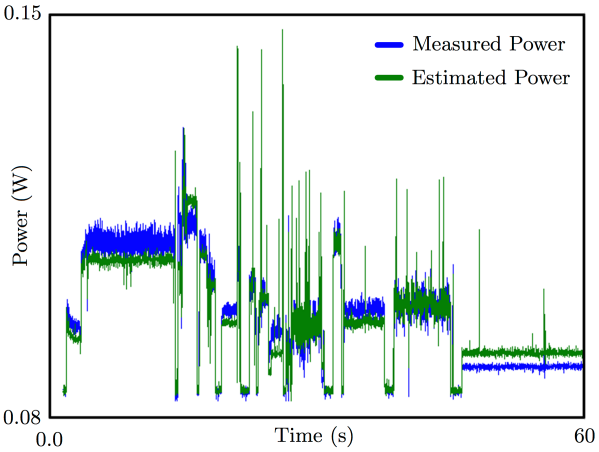

Figure 2. Real-time comparison between the actual power consumption and the PMC-based power model

$$\text{Power (300 MHz)} = 6.178\times10^{-2} + \text{E4}\times1.152\times10^{-8} + \text{E8}\times8.497\times10^{-9} + \text{E77}\times1.187\times10^{-7} + \text{E85}\times5.746\times10^{-9} \quad (1)$$

$$\text{Power (600 MHz)} = 1.598\times10^{-1} + \text{E4}\times2.205\times10^{-8} + \text{E8}\times1.851\times10^{-8} + \text{E77}\times2.016\times10^{-7} + \text{E85}\times7.778\times10^{-9} \quad (2)$$

$$\text{Power (800 MHz)} = 2.614\times10^{-1} + \text{E4}\times2.722\times10^{-8} + \text{E8}\times2.314\times10^{-8} + \text{E77}\times2.500\times10^{-7} + \text{E85}\times9.873\times10^{-9} \quad (3)$$

$$\text{Power (1 GHz)} = 3.238\times10^{-1} + \text{E4}\times2.644\times10^{-8} + \text{E8}\times2.294\times10^{-8} + \text{E77}\times2.541\times10^{-7} + \text{E85}\times1.0004\times10^{-8} \quad (4)$$

It has been shown that the PMC model was able to accurately estimate the power consumption of a variety of workloads. For the sake of interest, the model was recreated but using only one PMC which correlated particularly well with power: E85, the number of operations issued. The purpose of this was to get an idea of how much the PMC model relied on knowing the type of workload as opposed to just the intensity of the workload. This single PMC model, which was built and validated with identical data to the previous model, achieved an average error of 3.25%; a reduction in accuracy by 59%. This demonstrates that having more PMCs to give information on the type of workload, rather than just the number of instructions that are being architecturally executed, can result in a significant improvement in the models accuracy. However, an average error of just 3.25% from a model that has no information on the type of workload, demonstrates that having the extra information that performance counters offer is not necessary in most cases. This suggests that a model using CPU utilisation alone, a metric that correlates with power in a similar fashion to the number of operations issued, should be tested and analysed for its suitability in run-time power estimation.

3. UTILISATION-BASED POWER MODEL

Because PMCs are difficult or impossible to obtain on most embedded platforms, and because it was found single event counter could achieve a high accuracy in a PMC model (Section 2), it was decided to investigate the suitability of a power model that relies on simple CPU utilisation data. The PMC model could only be implemented on specific hardware and software that allowed PMCs to be read. For example

the embedded CPU models described in this paper and in the reviewed literature are implemented on old CPUs as PMCs were available on them (in [13], published in 2014, the newest CPU used was an ARM Cortex-A9, which was announced in 2007[2]).

3.1 Background

CPU Utilisation is a simple metric for understanding how much time the processor spends performing a task compared to being idle. It is often presented as a percentage in programs such as *top* (Linux), *Task Manager* (Microsoft Windows) and *Activity Monitor* (Mac OS X). Care has to be taken when considering using utilisation statistics for estimating power as it is a loosely defined metric. Usually, if the processor is executing any useful instruction it is classed as work and otherwise idle. However, even when executing a useful instruction, not every piece of hardware in the processor will be utilised, i.e. the work could be integer operations, move operations, waiting for IO or floating point operations, and not all of the parts of the processor responsible for all these operations will be utilised at the same time. So the processor will never be fully utilised but the utilisation metric will count the whole processor as being utilised when executing any type of instruction. In multiprocessor systems the utilisation value becomes more confusing because some hardware (e.g. the L3 cache) will be shared between several cores and the effect of these shared resources on the performance is dependent on the workload. This investigation was conducted to see if these factors caused unacceptable errors in the estimation results and whether an accurate power model could be built.

A relatively recent innovation to further improve the power efficiency of mobile processors is the concept of a asymmetric multi-core system, such as the *big.LITTLE* architecture from ARM. A big.LITTLE architecture incorporates two different core types with the same instruction-set architecture (ISA): a high performance, out-of-order superscalar 'big' core and a more power-efficient in-order 'little' core. Older implementations of the big.LITTLE architecture only allowed either all 'little' cores to be used *or* all 'big' cores to be used; different combinations of big and little cores could not be employed. Newer implementations of the big.LITTLE architecture have a feature called Global Task Scheduling (GTS) where the operating system can see both the big and little cores simultaneously and schedule tasks to any of them as required.

There has been little research using utilisation on desktop and server application, largely due to the fact that PMCs provide more information and are easily accessible. However, Dargie et al. [8] used the CPU utilisation to build a power model for servers and discussed other work where the CPU utilisation is used to give a prediction of the server's power consumption. There is work related to power consumption and CPU utilisation on smartphones, for example, Zhang et al. [18] built a battery behaviour model for smartphones and used processor utilisation to give an indication of the current power consumption of the CPU. For this model, the overall power consumption of the whole smartphone is measured, without the power of the individual components, such as the CPU, being measured directly. Kim et al. [11] also proposed scheduling schemes for the ARM big.LITTLE architecture but uses information from CPU utilisation instead of PMCs to improve load balancing. The proposed method results in an energy improvement of 11.35% with little impact on performance. Only the utilisation is used, performance counters do not have to be obtained. This makes the method able to work on any big.LITTLE development board where performance counters are not available. In this work, the run-time management is able to predict the impact of switching to another core type in advance.

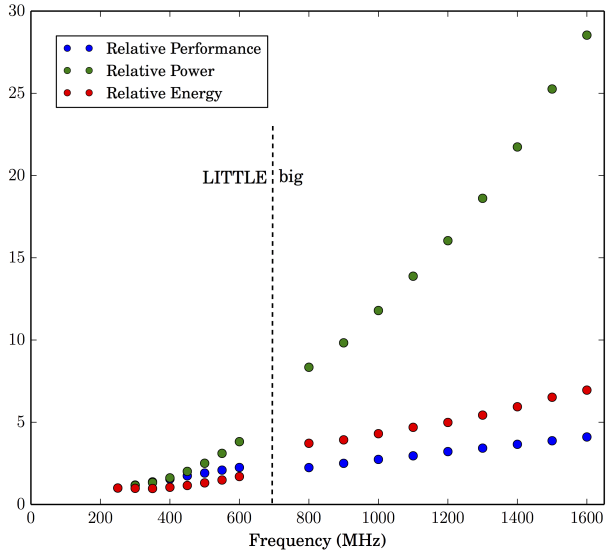

Figure 3. Performance and power trade-off at various frequencies on a big.LITTLE architecture

Figure 3 shows data collected as part of this research and shows characteristics of the big.LITTLE architecture. It can be seen that running a task at a higher frequency or on the 'bigger' core results in a significant increase in power with a proportionally lower performance increase. For example, the maximum frequency (1600 MHz) on the 'big' Cortex-A15 consumes 28× more power on average, across all of the workloads, than the 'little' Cortex-A7 running at the lowest frequency (250 MHz). The performance increase between these two points is 4×, which means that the energy used for a task of finite length is 5× greater. In other words, an average task will always consume 5× more energy when running at the maximum frequency on the 'big' Cortex-A15 core instead of the lowest frequency on the 'little' Cortex-A7. As expected, it is always more energy efficient to run tasks on the 'little' A7 cores at low frequencies. The only reason to use higher frequencies or switch to the 'big' Cortex-A15 cores is to achieve a higher performance and meet deadlines. Interestingly, the very lowest frequency is not the most energy efficient while running the test workloads; the most energy efficient frequency was found to be 350 MHz which is slightly more efficient than the lowest frequency of 250 MHz.

This shows how the big.LITTLE concept gives the operating system more opportunities to trade-off power and performance and find the optimum hardware configuration for the currently running task, the current conditions and the user's requirements. However, to get the maximum potential out of this, the RTM needs to be more complex, have reliable and accurate run-time information and know, in advance, the impact of changing the run-time configuration (i.e. changing frequency or migrating from a 'little' core to a 'big' core).

3.2 Experimental Setup

The experiments were carried out on an ODROID-XU+E development board by Hardkernel [10]. It has an Exynos 5 Octa 5410 SoC that contains an ARM big.LITTLE architecture with four Cortex-A15 'big' out-of-order cores and four Cortex-A7 'little' in-order cores. This SoC also contains an Imagination Technologies PowerVR SGX544MP3 GPU. It is used in the Samsung Galaxy S4 and a variation of this SoC,

which has the same CPU but with Global Task Scheduling enabled, is used in the Samsung Galaxy S5, Chromebook 2 and Samsung Galaxy Note 3, all three of which were released in 2014 .The board was running a Linaro Ubuntu distribution with kernel version 3.4.75.

The ODROID-XU+E board contains voltage and current sensors for each cluster; the total power of the four Cortex-A7 cores and the total power of the four Cortex-A15 cores can be determined. A program was written that calculates the CPU utilisation from the */proc/stat* virtual file and the power from the sensors while workloads are being executed. The sensor and utilisation values were sampled every 50 ms; note that this is not an instantaneous reading but an average utilisation over the preceding 50 ms. The program could then calculate the average utilisation of each core for a particular workload, as well as the average power consumption and execution time.

The MiBench embedded benchmark suite was used to work the processor while the experiments were running. It is aimed towards mobile and embedded systems, as opposed to high performance computers and was chosen because it has a large number of workloads with a high variation in different types of operations (e.g. memory operations, control operations etc.). The workloads were tested at every frequency on both the Cortex-A7 and Cortex-A15 cores. Different workloads were also run simultaneously on different cores to establish the relationship between power and utilisation when different cores were experiencing vastly different loads. Once utilisation and power data had been captured, regression analysis was used to build the models.

3.3 Model Generation and Validation

It was found that the contribution of each core to the overall power consumption of the CPU was linear. For example, Core 1 being 20% utilised and Core 2 being 30% utilised uses the same amount of power as just Core 1 being 50% utilised. Therefore the models were built using the average utilisation of all the cores and the individual core power can be derived from the fraction of the overall utilisation that they make up.

Utilisation models were built for both the 'little' Cortex-A7 cores and the 'big' Cortex-A15 cores at every frequency. From the results it could be seen that there was a very slight in curve at higher frequencies for both core types. It was found that a second-order linear regression model optimally fit the trend. However, the curve is very slight and there is only a small increase in average error when a first-order linear regression model is used (Figure 4). Equation 5 shows the form of the model, where u is the utilisation value. The parameters for different frequencies are given in Table 3.3. The models were validated on 40 'unseen' workloads that each involve running many different tasks on different cores simultaneously. Figure 5 shows how the model accuracy changes with different types of workload. The bar labelled *40 unseen* is the average of all of the unseen workloads and has a relatively high power value due to the fact that different numbers of cores were being exercised by different amounts, as opposed to just one core being exercised. The first-order model achieved an average error across all workloads and frequencies of 7.43% and 8.6% for the Cortex-A7 and Cortex-A15, respectively, while the second-order model achieved an average error of 5.6% and 7.2% for the Cortex-A7 and Cortex-A15, respectively. The accuracy of the 'little' Cortex-A7 is slighter better than that of the Cortex-A15 due to its simpler in-order architecture, making its power more predicable with a wide range of workload types. This second-order model is best suited for real-time run-time management. In such a scenario, the model would be swapped when the operating system changes the system frequency or core type.

Freq (MHz)	p_0	p_1	p_2
250	0.010162804	0.001572937	-1.63E-06
300	0.009521533	0.001955864	-2.80E-06
350	0.010338362	0.002260855	-3.27E-06
400	0.012404617	0.002786165	-4.16E-06
450	0.014982476	0.003470516	-5.15E-06
500	0.019350116	0.004348964	-6.59E-06
550	0.023879147	0.005364183	-8.19E-06
600	0.029694786	0.006573153	-1.04E-05
800	0.036437558	0.015975673	-2.45E-05
900	0.043569437	0.018618245	-2.36E-05
1000	0.05151296	0.022298895	-2.31E-05
1100	0.06267777	0.026027083	-1.69E-05
1200	0.071779829	0.029651724	-9.56E-06
1300	0.082334714	0.03450953	8.47E-06
1400	0.092116725	0.046563344	-9.22E-05
1500	0.100661815	0.060414109	-0.000231806
1600	0.113207437	0.071046296	-0.000356278

Table 1. Second order parameters for utilisation-based power model

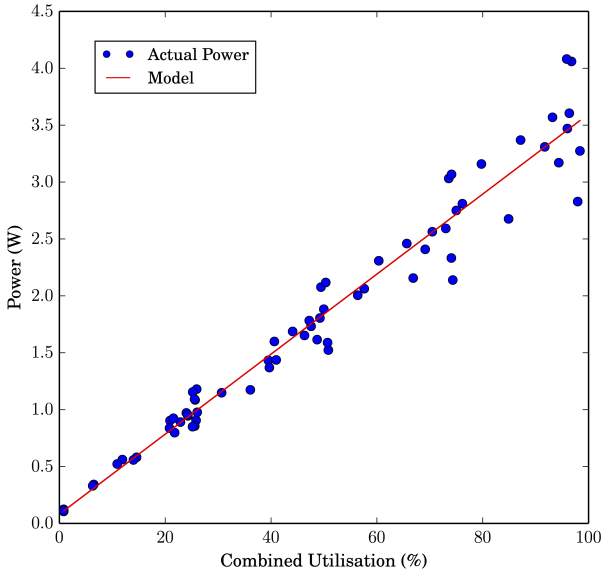

Figure 4. First-order linear regression model at 1300 MHz. Note: 100% refers to all four Cortex-A15 cores being fully utilised

$$\text{Power (W)} = p_0 + up_1 + u^2 p_2 \tag{5}$$

To test the strength of the relationship between voltage, frequency and the power consumption, a single model was built that would predict the power at any frequency. This simplifies the model and allows the hypothetical power at other frequencies, which are not in the voltage frequency table, to be estimated. For example, the maximum frequency of the ODROID-XU is 1600 MHz, but what might the power consumption be if it could run at 1800 MHz? If the big.LITTLE processor is set to a frequency of 600 MHz, it uses the 'little' Cortex-A7 core, but if the Cortex-A15 could be clocked to 600 MHz, how would the energy efficiency compare? To accomplish this, the model is broken up into two parts: Firstly, the parameters of the model are themselves calculated from a first-order linear regression model, with frequency as the input (Equation 6). Once the parameters for that particular frequency have been derived, the power can be calculated as before with the utilisation value (Equation 7). This model achieved an accuracy of

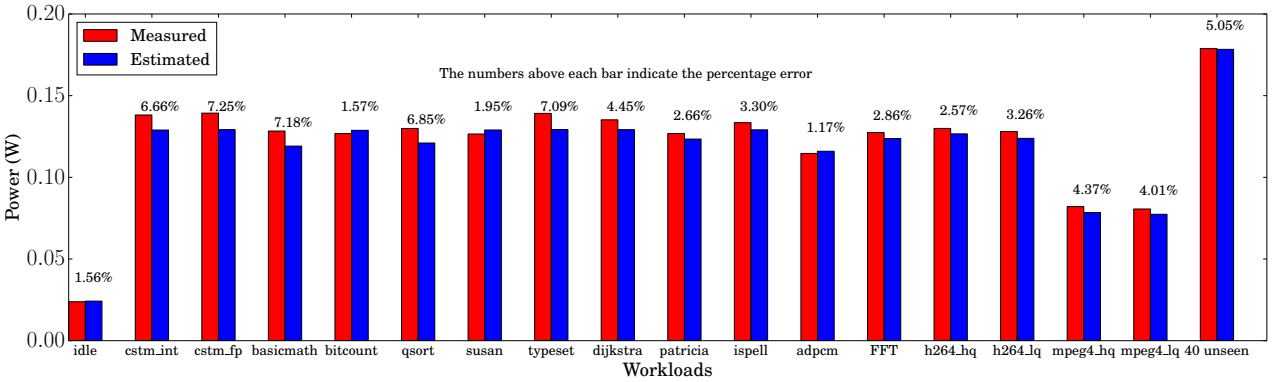

Figure 5. Estimated and measured power for each workload and corresponding average error, at 500 MHz

10.4% and 8.5% for the Cortex-A7 and Cortex-A15, respectively.

$$p_0 = x_{00} + fx_{01}$$

$$p_1 = x_{10} + fx_{11} \quad (6)$$

$$p_2 + x_{10} + fx_{21}$$

$$\text{Power (W)} = p_0 + up_1 + u^2 p_2 \quad (7)$$

Both the per core and per task (by PID) power can be estimated by the utilisation model. Such information is tremendously useful to a run-time management system and can allow it to make well-informed decisions, leading to better energy efficiency. Furthermore, a model to estimate how much utilisation a task would consume if it were running at a different frequency or on a different core type was implemented. From the vast amount of data gathered from running the training workloads, a lookup table was built that allows the conversion from one frequency to another frequency and between the Cortex-A7 and Cortex-A15. Importantly, as before, the model was verified using a wide variety of unseen workloads, instead of using the same training workloads that were used to build the model, as is the case in many similar papers. Every switching combination between one frequency to another (which includes switching between core types) was tested and an average power estimation error of 10.3% was achieved, which includes both the error in the power models (the second-order fixed models were used) and the error in estimating the utilisation when running on a different core or at a different frequency.

4. CONCLUSION

A PMC-based and a utilisation-based power model is presented in this paper. The models were simple but accurate, allowing a run-time manager to estimate power consumption with a low overhead. The utilisation-based power model has several practical advantages over the PMC-based model, for example, it can be implemented on virtually any mobile device. We demonstrate the accuracy of the model in predicting the power consumption per core, per task, and also across cores of an asymmetric big.LITTLE architecture.

5. ACKNOWLEDGMENTS

This work was supported by the Engineering and Physical Sciences Research Council Programme Grant, EP/K034448/1. See www.prime-project.org for more information about the PRiME programme.

References

[1] Agilent Technologies. Agilent n6700 modular power system family [data sheet]. `http://www.home.agilent.com/en/pd-1842303-pn-N6705B/dc-power-analyzer-modular-600-w-4-slots`, June 2012.

[2] ARM Holdings. ARM Unveils Cortex-A9 Processors For Scalable Performance and Low-Power Designs. `http://www.techradar.com/news/`, October 2007. [Online; accessed 17-November-2014].

[3] R. Basmadjian and H. De Meer. Evaluating and modeling power consumption of multi-core processors. In *Future Energy Systems: Where Energy, Computing and Communication Meet (e-Energy), 2012 Third International Conference on*, pages 1–10, May 2012.

[4] BeagleBoard. What is BeagleBoard-xM? `http://beagleboard.org/beagleboard-xm`, June 2014. [Online; accessed 20-July-2014].

[5] R. Bertran, M. Gonzelez, X. Martorell, N. Navarro, and E. Ayguade. A systematic methodology to generate decomposable and responsive power models for cmps. *Computers, IEEE Transactions on*, 62(7):1289–1302, July 2013.

[6] W. Bircher and L. John. Complete system power estimation: A trickle-down approach based on performance events. In *Performance Analysis of Systems Software, 2007. ISPASS 2007. IEEE International Symposium on*, pages 158–168, April 2007.

[7] G. Da Costa and H. Hlavacs. Methodology of measurement for energy consumption of applications. In *Grid Computing (GRID), 2010 11th IEEE/ACM International Conference on*, pages 290–297, Oct 2010.

[8] W. Dargie and J. Wen. A probabilistic model for estimating the power consumption of processors and network interface cards. In *Trust, Security and Privacy in Computing and Communications (TrustCom), 2013 12th IEEE International Conference on*, pages 845–852, July 2013.

[9] M. Guthaus, J. Ringenberg, D. Ernst, T. Austin, T. Mudge, and R. Brown. Mibench: A free, commercially representative embedded benchmark suite. In *Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop on*, pages 3–14, Dec 2001.

[10] Hardkernel. Odroid-xu. `http://www.hardkernel.com/`, 2013. [Online; accessed 20-July-2014].

[11] M. Kim, K. Kim, J. Geraci, and S. Hong. Utilization-aware load balancing for the energy efficient operation of the big.little processor. In *Design, Automation and Test in Europe Conference and Exhibition (DATE), 2014*, pages 1–4, March 2014.

[12] J. Nunez-Yanez and G. Lore. Enabling accurate modeling of power and energy consumption in an arm-based system-on-chip. *Microprocessors and Microsystems*, 37(3):319 – 332, 2013.

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

[14] R. Rodrigues, A. Annamalai, I. Koren, and S. Kundu. A study on the use of performance counters to estimate power in microprocessors. *Circuits and Systems II: Express Briefs, IEEE Transactions on*, 60(12):882–886, Dec 2013.

[15] Samsung. Specs. `http://www.techradar.com/news/`, April 2014. [Online; accessed 17-November-2014].

[16] K. Singh, M. Bhadauria, and S. A. McKee. Real time power estimation and thread scheduling via performance counters. *SIGARCH Comput. Archit. News*, 37(2):46–55, July 2009.

[17] Texas Instruments. Dm3730, dm3725 digital media processors [data sheet]. `http://www.ti.com/product/dm3730`, July 2011.

[18] L. Zhang, B. Tiwana, R. Dick, Z. Qian, Z. Mao, Z. Wang, and L. Yang. Accurate online power estimation and automatic battery behaviour based power model generation for smartphones. In *CODES+ISSS IEEE/ACM/IFIP*, pages 105–114, Oct 2010.