UNIVERSITY OF SOUTHAMPTON

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES

School of Electronics and Computer Science

Hardware-Validated Performance and Power Modelling of Heterogeneous Multi-Processing Architectures

by

Matthew James Walker

ORCID ID: 0000-0001-6368-3644

Thesis for the degree of Doctor of Philosophy

June 2019

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING AND PHYSICAL SCIENCES
School of Electronics and Computer Science

Doctor of Philosophy

Hardware-Validated Performance and Power Modelling of Heterogeneous Multi-Processing Architectures

by Matthew James Walker

Modern processors are becoming increasingly more complex and utilise higher numbers of Heterogeneous Multi-Processing (HMP) cores. Energy-efficiency has become the primary design constraint in recent years, and improvements enable battery-powered devices to run longer and reduce the energy and cooling costs in data centres. Moreover, increased energy-efficiency enables greater peak performance under the thermal and power constraints, enabling innovative new uses and applications. Accurate run-time power estimations are critical in guiding online energy-saving techniques and energy-aware scheduling decisions to find the optimum performance, power and energy trade-off. This thesis presents a statistically-rigorous methodology for developing accurate and stable empirical power models for providing run-time power estimations to a run-time manager (RTM) while considering thermal variation, coefficient stability, heteroscedasticity, robust model specification, and non-ideal voltage regulation. The novel methodology ensures that the models perform significantly more accurately across a wider range of workloads when compared with existing run-time power modelling methodologies, achieving average errors lower than four percent. Practical considerations and shortcomings in existing approaches are also identified and addressed.

Furthermore, the recent slowdown in technology scaling has forced researchers and engineers to rely on micro-architectural advances and system-level optimisations to drive performance improvement, the development of which is underpinned by simulation tools. However, such simulation tools inevitably have limitations and contain sources of error which, if not understood by the user, can lead to inaccurate results and incorrect conclusions. This thesis presents a methodology for evaluating CPU performance models and identifying specific sources of error, allowing such models to be improved; extended to other CPUs; validated after changes; and tested for suitability to a specific use case. These hardware-validated performance models are combined with the empirical power models to enable accurate and reliable performance, power and energy simulation.

Moreover, the *Powmon* and *GemStone* software tools are presented, which implement the methodologies for developing power models and validating performance models, respectively.

Contents

List of Figures

List of Tables

Accompanying Material

All data supporting this study are openly available from the University of Southampton repository at `https://doi.org/10.5258/SOTON/D0979`

A website containing information, software tools and experimental data supporting Chapter 4 and Chapter 5 is at `http://www.powmon.ecs.soton.ac.uk`

A website containing information, software tools and experimental data supporting Chapter 6 is at `http://gemstone.ecs.soton.ac.uk`

Declaration of Authorship

I, Matthew J. Walker, declare that the thesis entitled *Hardware-Validated Performance and Power Modelling of Heterogeneous Multi-Processing Architectures* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: [305, 309, 308, 304, 310].

Signed: _____

Date: _____

Acknowledgements

I would, first and foremost, like to thank my supervisors, Bashir Al-Hashimi and Geoff Merrett, for their support, encouragement and valuable feedback throughout the many long years of my PhD studies. I would also like to thank Stephan Diestelhorst and Andreas Hansson for their invaluable guidance and technical insights both during and after my internships in Arm Ltd.

This work would not be possible without the funding and support from the Engineering and Physical Sciences Research Council (EPSRC), Arm Research, the Arm-ECS Research Centre, and the University of Southampton.

I am also grateful to my colleagues for their invaluable support, stimulating discussion, and their company when working at unsociable hours. These include, but are not limited to: Karunakar Reddy Basireddy, Sascha Bischoff, Graeme Bragg, Anup Das, Mauricio Gutierrez Alcala, Kath Kerr, Andy Kufel, Charlie Leech, Luis Maeda-Nunez, Bassem Ouni, Ghaithaa Manla, Andy Mallett, Rishad Shafik, Sheng Yang and Ilias Vougioukas.

With many long days and weekends spent in the office, I am particularly grateful to friends and family for being patient and providing support and reassurance through the most difficult times. In particular, I would like to mention Domenico, Benedetta, Ananya, my brother Rob, and my sister-in-law Liz. Finally, I would like to thank my parents, Catherine and Graham, for their love, support and encouragement, without which I would not have been able to complete this work. I would like to dedicate this thesis to the memory of my grandfather, Anthony Billson, who inspired me to pursue a career in engineering.

Nomenclature

$\|\boldsymbol{A}\|$	Determinant of Matrix A
$\boldsymbol{A}^*$	Conjugate Transpose of Matrix A
$\boldsymbol{A}^+$	Pseudo-Inverse of Matrix A
$\boldsymbol{A}^T$	Transpose of Matrix A
$\boldsymbol{A}^{-1}$	Inverse of Matrix A
ANN	Artificial Neural Network
API	Application Programming Interface
$\boldsymbol{\beta}$	Coefficient Vector
BLUE	Best Linear Unbiased Estimator
BTB	Branch Target Buffer
CMT	Chip-level Multiprocessing
CPD	Change Point Detection
CPI	Cycles Per Instruction
CPU	Central Processing Unit
CV	Coefficient of Variation
D	Cook's Distance
DPM	Dynamic Power Management
DRAM	Dynamic Random Access Memory
DVFS	Dynamic Voltage and Frequency Scaling
$\boldsymbol{e}$	Residuals Vector
E	PMC Event Rate divided by CPU Frequency
EAS	Energy-Aware Scheduling
EDP	Energy-Delay Product

EPC	Energy per Cycle
ESS	Explained Sum of Squares
f_{clk}	CPU Frequency
FPU	Floating Point Unit
FS	Full-System
GPU	Graphics Processing Unit
GTS	Global Task Scheduling
$\boldsymbol{H}$	Hat Matrix
HCA	Hierarchical Cluster Analysis
HMP	Heterogeneous Multi-Processing
HW	Hardware
$\boldsymbol{I}$	Identity Matrix
I_{btbt}	Band-to-Band Tunnelling Current
I_{leak}	Leakage Current
I_{ox}	Gate-Oxide Current
I_{sub}	Sub-Threshold Current
InO	In-Order
IoT	Internet of Things
IP	Intellectual Property
IPC	Instructions Per Cycle
ISA	Instruction Set Architecture
k-NN	k-Nearest Neighbours
L1D	Level 1 Data
L1I	Level 1 Instruction
L2	Level 2
LRU	Least Recently Used
$\boldsymbol{M}$	Annihilator Matrix
MAPE	Mean Absolute Percentage Error
ML	Maximum-Likelihood
MLE	Maximum-Likelihood Estimation

MOESI	Modified Owned Exclusive Shared Invalid
MPE	Mean Percentage Error
MSE	Mean Squared Error
MTTF	Mean Time to Failure
n	Number of observations
NN	Neural Network
NoC	Network on Chip
OLS	Ordinary Least Squares
OoO	Out-of-Order
OS	Operating System
p	Number of predictors (features)
$\boldsymbol{P}$	Projection Matrix (also known as the hat matrix, $\boldsymbol{H}$)
P_{CPU}	CPU Power Consumption
P_{dyn}	Dynamic Power Consumption
P_{sc}	Short-Circuit Power Consumption
P_{static}	Static Power Consumption
P.I.	Prediction Interval
PCA	Principle Component Analysis
PMC	Performance Monitoring Counter
PMU	Performance Monitoring Unit
PPA	Performance, Power, Area
QoE	Quality of Experience
QoS	Quality of Service
$\boldsymbol{r}$	Internally Studentised Residuals
R^2	Coefficient of Determination
$\bar{R}^2$	Adjusted Coefficient of Determination
RMSE	Root Mean Squared Error
RSS	Residual Sum of Squares
RTL	Register Transfer Level
RTM	Run-Time Management

Σ	Covariance Matrix
SER	Standard Error of Regression
SMT	Simultaneous Multithreading
SoC	System-on-Chip
SPICE	Simulation Program with Integrated Circuit Emphasis
SRAM	Static Random Access Memory
SVM	Support Vector Machine
T	Temperature
$\boldsymbol{t}$	Externally Studentised (Deleted) Residuals
TDP	Thermal Design Power
TLB	Translation Lookaside Buffer
TSS	Total Sum of Squares
V_{DD}	CPU Voltage
VIF	Variance Inflation Factor
VLSI	Very Large Scale Integration
WAPE	Weighted Absolute Percentage Error
WLS	Weighted Least Squares
$\boldsymbol{X}$	Design Matrix
$\boldsymbol{y}$	Dependent Variable Vector

Chapter 1

Introduction

The primary design goal for computer processors has historically been performance, which has reliably improved for many years with the aid of technology scaling. However, technology scaling has slowed and physical limits have caused a reduction in performance improvement and have led to challenges with heat dissipation. This has forced processor architects to look elsewhere for performance gains in order to keep up with industry expectations derived from Moore's Law [256]. Modern processors are therefore heavily optimised towards performance and have complex designs that utilise deep pipelines, memory hierarchies, speculative execution, multiple execution units (superscalar design), out-of-order execution, single instruction, multiple data (SIMD) co-processors, and multi-core designs.

In the last decade there has been a shift in emphasis from performance towards energy efficiency due to the emergence of mobile devices, such as smartphones, tablets, thinner fan-less laptops and, more recently, smart watches. Such devices have limited thermal dissipation and are typically powered from batteries with limited capacity.

By improving CPU energy-efficiency through novel design and careful online management, a mobile processor can achieve higher performance while respecting its Thermal Design Power (TDP), which enables emerging mobile applications such as augmented reality, machine learning, artificial intelligence and 3D photography. Energy-efficient processor designs, initially targeted towards mobile devices, are now also being utilised in data centres to reduce energy bills and the cost spent on cooling systems.

The following two key areas provide opportunities for improving performance and energy efficiency despite the challenges in technology scaling:

1. Run-time system optimisation that controls task scheduling and online energy management mechanisms effectively to achieve the desired level of user experience while simultaneously respecting thermal and power constraints, and maximising energy-efficiency;

2. Architectural, micro-architectural and system-level design innovations that optimise for modern workload demands and exploit new technologies.

Accurate performance, power, thermal, and energy models are required for both guiding run-time system optimisation and exploring innovative software and hardware improvements.

This thesis presents a methodology for characterising existing processors and developing accurate and stable run-time power models by employing statistical and machine learning techniques. The resulting models aid system run-time management (RTM) and energy-aware scheduling (EAS) in making efficient decisions by providing accurate run-time power estimations. System designers, processor architects and researchers rely on performance and power simulators to evaluate new ideas, new technologies, and system optimisations. The results generated by these simulators underpin processor research and development, however, there are challenges in ensuring the accuracy and reliability of such tools, which impacts the results and can ultimately change the conclusions drawn from investigations. A methodology for directly comparing processor models against a hardware reference platform and identifying sources of error is presented. Understanding, evaluating, and reducing the errors in such models provides researchers with an accurate baseline model that responds to changes in a representative way. Furthermore, by understanding the sources of error in simulator models, researchers can evaluate how these errors may impact their investigations, and whether the models are suitable for their specific use case.

While the work presented in this thesis focusses on mobile CPUs, many of the methodologies and techniques are applicable to CPUs in desktop and high-performance systems, as well as other components, such as the GPU and accelerators.

The research detailed in this thesis provides improved methodologies for developing accurate and stable hardware-validated CPU models for energy-aware run-time CPU management and design-space exploration.

1.1 Research Questions

This work answers the following research questions:

1. How robust are the current state-of-the-art power modelling methodologies and how applicable are they to mobile CPUs?

2. How effective are PMC-based power models when applied to mobile CPUs in terms of accuracy and responsiveness?

3. Which micro-architectural event statistics are most effective in estimating the power consumption across a wide range of workloads and workload phases, and how many input features are required?

4. Do the existing run-time PMC-based power models address practical considerations and how suitable are they for real-world deployment?

5. Are current performance and power simulation frameworks accurate and reliable and, if not, how can they be improved?

6. How applicable are empirical PMC-based power models to design-space exploration?

1.2 Energy Efficiency and Run-Time Management

Fundamental to improving energy-efficiency of a CPU is managing it effectively at run-time. Modern CPUs employ various power management techniques, including Dynamic Voltage and Frequency Scaling (DVFS) [235, 98, 227, 250, 274], Dynamic Power Management (DPM) [25] and Heterogeneous Multi-Processing (HMP, such as Arm big.LITTLE and DynamIQ technology) [15, 7], which need to be controlled efficiently in order to achieve the desired trade-off between performance and energy while remaining within thermal limits.

Intelligent use of DVFS can save energy without sacrificing performance by exploiting processor slack in memory-intensive tasks [274] (see Section 2.3). Additionally, it is often appropriate to trade-off device performance for improved energy-efficiency. For example, by understanding the Quality of Experience (QoE), device performance can be reduced, therefore saving energy, without significantly impacting the performance *perceived* by the user [37].

Extensive research has shown that careful run-time management of DVFS, DPM and HPM scheduling has significant benefits, such as improved energy efficiency or extended device lifetime reliability [73, 76, 268, 228, 272]. However, run-time knowledge of the power consumption of each CPU core in the system is paramount in finding the optimum power, performance, and reliability trade-off and to efficiently exploit energy-saving opportunities.

1.3 Power Modelling

Knowledge of run-time power consumption is essential for efficiently managing energy-saving techniques and maximising performance without exceeding thermal and power delivery limits. However, it is not typically practical to install power sensors in hardware due to their physical size and the cost of implementation. Furthermore, a run-time manager requires more fine-grained per-core estimations; the Power Management Integrated Circuit (PMIC) typically supplies power at a cluster level, and the power supply to individual cores can not therefore be measured.

Previous works have shown that Performance Monitoring Counters (PMCs), which are internal CPU registers that count architectural and micro-architectural events, can be utilised to estimate run-time power consumption (Section 2.6). Despite energy-efficiency being of particular importance in mobile processors, existing work has focussed on desktop and server class processors. This is largely due to technical challenges in accessing PMCs on mobile platforms. To address this, this thesis presents a method of accessing PMCs and creates open-source software tools enabling their more wide-spread use in research. Novel techniques for processing and analysing PMC event data are also presented. Existing power modelling techniques are applied to mobile platforms and significant shortcomings are identified and addressed.

Empirical power models in existing works are typically trained and tested with a limited number of workloads. When employed in run-time management, models must make predictions across a very large set of realistic workloads, which will include workloads that are very different to the synthetic ones used to train the model. Furthermore, the models will be estimating the power over small time periods, meaning that it is estimating the power consumption of workload micro-phases, which are significantly more diverse than the full workload itself. Therefore, a vital requirement of

a run-time power model is the ability to make accurate predictions across a large and diverse set of observations, even if they are not well represented in the training set.

This thesis presents a methodology that uniquely employs several techniques to improve coefficient stability and it is experimentally demonstrated how doing so results in a significantly reduced error when testing on a larger number of workloads. Furthermore, it is experimentally shown how the statistics reported to demonstrate the quality models in existing works can be optimistic and misleading without considering other metrics.

The importance of model input feature selection in ensuring stability is demonstrated and several techniques are employed to develop a methodology of choosing PMCs, including Hierarchical Cluster Analysis (HCA), Principle Component Analysis (PCA), statistical significance tests, forward stepwise selection techniques, and analysis of the variance inflation factor (VIF).

The methodology carefully specifies the models using knowledge of CMOS power consumption, employing statistical significance tests and analysing the residuals. Several existing works are identified to have misspecified models, which has lead to the conclusion that models using a single equation and set of coefficients across multiple DVFS levels achieve poor accuracy in comparison to having a separate equation for each DVFS level. This thesis shows how a carefully and correctly specified, single-equation, stable power model can achieve high accuracy across all DVFS levels. The unified model separates static and dynamic power consumption and it is experimentally shown that the coefficients can be estimated by only running the training workloads at one DVFS level and taking a single observation at each of the remaining DVFS levels. As well as having practical benefits due to a significant reduction in data collection time (40 hours, reduced to 20 minutes), this demonstrates the robustness of the formulation.

The presented methodology uniquely highlights and addresses the problem of heteroscedasticity in PMC-based power modelling and how it affects the statistics provided by regression software. It is also found that the voltage regulators exhibit non-ideal behaviour that affects the power consumption and a method of accounting for this in the power models is presented. Temperature has a significant effect on the static power consumption and several methods to account for this, both with and without the use of thermal sensors, are presented.

The overhead of the presented power models was measured at various sample frequencies to ensure their suitability for run-time power estimation. They can be implemented in the kernel or in userspace and provide per-cluster, per-core or per-thread power estimations for guiding run-time management decisions. Comparisons to existing models demonstrate how the novel techniques result in a significantly lower error when tested on a large set of diverse workloads and the coefficient stability ensures a low error variance across all test samples.

1.4 Simulation Tools

Full-system simulation tools enable researchers and system designers to evaluate new techniques, innovations and optimisations in both hardware and software. The gem5 [30] full-system performance simulator enables a custom hardware implementation to be created and runs unmodified operating systems and applications while providing performance measurements and statistics. It

supports multiple Instruction-Set Architectures (ISAs), including both ARM and x86, and contains detailed CPU models, which offer high accuracy at the cost of simulation time. While some methodologies in this thesis are applied to *gem5*, they are also applicable to other full-system simulators. To conduct power analysis, the modelled architectural and micro-architectural events from the performance simulator are fed into a power modelling framework, such as McPAT [174]. A detailed discussion of full-system performance and power simulation, including gem5 and McPAT, is provided in Section 2.7.

While such tools underpin many works of research in design-space exploration, they inherently suffer from errors that potentially affect the results and conclusion drawn. Existing works have highlighted the lack of thorough validation of many simulation tools, and the problem of many users of such tools not understanding the limitations [216]. Existing works have validated specific models of the gem5 simulation framework and identified *specification error*, which is caused by model parameters being incorrectly set due to a lack of detailed (publicly available) information about the CPU being modelled, as a key reason for large errors [117, 48].

A typical use case of simulation tools is to evaluate a proposed improvement to part of the system. To evaluate the idea, a reference model is required to generate results for a baseline case. The proposed changes are then added to the performance simulator and the new results are compared to the baseline results and evaluated. A reference model based on a typical system is therefore an important component of a simulation framework. Significant errors in the reference model prevent it from responding in a representative way to changes under test.

This thesis addresses these problems by conducting an up-to-date validation of gem5 models against a hardware platform and developing a methodology that highlights, and aids the understanding of, key sources of error in the models by employing statistical and machine learning techniques.

Typical power modelling frameworks, including McPAT, provide much flexibility at the expense of accuracy [171] and existing works have found significant errors in the results from McPAT [48]. This thesis postulates that the more accurate empirical power models developed for run-time estimations are more suitable in many situations. While the empirical power models are only valid for the device they are characterised on (the reference model), they will accurately estimate power due to the knock-on effects of a change. For example, if a new branch predictor is proposed, a user can use a hardware-validated gem5 reference model and the corresponding empirical power model for the baseline case. The gem5 model can then be modified with the new branch predictor and the performance and power evaluated again. The empirical power model will accurately model the CPU after this change, including the knock-on effects that the change has on all the components of the system, with the exception of the hardware modification to the branch predictor itself. If this is deemed non-negligible, then this component can be accurately modelled separately. This approach is more accurate than relying on a flexible power modelling framework in which there are potentially large errors in all system sub-components.

The gem5 simulation framework allows power equations to be inserted into the simulator for run-time energy analysis. However, no power equations themselves are publicly available. The methodology and corresponding open-source software tools presented in this thesis enable accurate and stable empirical power models to be developed and used within the gem5 simulation framework. Furthermore, they allow the gem5 models to be validated against hardware platforms and sources of error to be identified and analysed. The models themselves can then be improved and researchers us-

ing the them can understand how the errors present impact their specific investigation, and evaluate whether the models are suitable for their use case.

1.5 Contributions

The research presented in this thesis has resulted in the following contributions:

1. The first published implementation of a hardware PMC-based power model on a mobile CPU with high workload power variation;
2. A novel methodology for developing accurate and stable run-time power models that improves on the existing state-of-the-art and is demonstrated on a heterogeneous multi-processing (HMP) platform;
3. Significant challenges in CPU power modelling were identified for the first time and addressed, such as the impact of non-ideal voltage regulation, the presence of heteroscedasticity, the effect of PMC event interaction on the coefficient errors, misspecification of models in typical approaches and the impact of using typical benchmarking suites on coefficient errors;
4. Novel methods for modelling the effects of ambient temperature variation and switching cores offline are presented, enabling the power to be decomposed into its constituting parts;
5. A novel methodology for identifying sources of error in full-system performance simulation tools that allows simulator models to be improved, extended to other processors, validated after changes, and tested for suitability to specific use cases.
6. Techniques for optimising and integrating the empirical PMC-based power models with the calibrated performance simulator, enabling accurate hardware-validated performance, power and energy simulation for design-space exploration.

1.6 Authored Publications

1.6.1 Journals

- M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das, S. Yang, B. M. Al-Hashimi, and G. V. Merrett, "Accurate and Stable Run-Time Power Modeling for Mobile and Embedded CPUs", in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 36, no. 1, pp. 106-119, Jan. 2017 [309].

  Work from this publication is presented in Chapter 4.

1.6.2 Conferences

- M. J. Walker, S. Bischoff, S. Diestelhorst, G. V. Merrett, B. M. Al-Hashimi, "Hardware-validated CPU performance and energy modelling". IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), Belfast, United Kingdom. 02 - 04 Apr 2018 [304].

  Work from this publication is presented in Chapter 6.

1.6.3 Workshops

- M. J. Walker, A. K. Das, G. V. Merrett, and B. M. Hashimi. "Run-time power estimation for mobile and embedded asymmetric multi-core CPUs" HIPEAC Workshop on Energy Efficiency with Heterogeneous Computing (EEHCO), Netherlands. 19 - 21 Jan 2015 [305].

  Parts of the work in this publication are presented in Chapter 3.

- M. J. Walker, S. Diestelhorst, A. Hansson, D. Balsamo, G. V. Merrett and B. M. Al-Hashimi, "Thermally-aware composite run-time CPU power models," 26th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS), Bremen, 2016, pp. 17-24 [308].

  Work from this publication is presented in Chapter 5.

- M. J. Walker, Stephan Diestelhorst, Geoff V. Merrett, Bashir M. Al-Hashimi "Accurate and Stable Empirical CPU Power Modelling for Multi- and Many-Core Systems" Adaptive Many-Core Architectures and Systems Workshop, York, UK, June 2018.

1.6.4 Tutorials

- M. J. Walker, V. Spiliopoulos, S. Diestelhorst, "Building Online Power Models from Real Data", [Half-Day Tutorial] MICRO-48, Waikiki, Hawaii, December 2015

  An interactive tutorial where 10 Hardkernel ODROID-XU3 development boards were provided and participants developed run-time power models for them using the *Powmon* software tools, which implement the methodology presented in Chapter 4. Instructional presentations on the software tools were given, as well as presentations describing practical considerations, the statistical methods underpinning the processes, and results. Every participant successfully developed accurate and stable power models and a development board was presented to the team with the best accuracy and stability.

- M. J. Walker, V. Spiliopoulos, S. Diestelhorst, "Building Online Power Models from Real Data", [Full-Day Tutorial] ISPASS, Uppsala, Sweden, April 2016.

  The authors were invited to give a presentation based on the format of the previously described tutorial. This tutorial was a full-day tutorial and participants carried out more tasks using the *Powmon* software tools, including further analysis of the PMC event selection techniques. Every participant successfully selected features, developed power models, and analysed the results in the online, web-based results viewer. Power modelling results were presented, along with analysis of how temperature influences the static power consumption, and how the estimated power can be sub-divided into its constituting parts.

- S. Bischoff, M. Bonnici, M. J. Walker, ISPASS, Belfast, United Kingdom, April 2018

  A 45 minute presentation on power modelling techniques was contributed to a tutorial focussing on energy-aware scheduling and energy improvements on Arm-based systems.

1.6.5 Poster Presentations

- M. J. Walker, S. Diestelhorst, A. Hansson, B. M. Al-Hashimi and G. V. Merrett, "Building Run-Time CPU Power Models from Real Data", DATE, Dresden, Germany, March 2016.

- M. J. Walker, S. Diestelhorst, A. Hansson, K. R. Basireddy, B. M. Al-Hashimi and G. V. Merrett, "Thermally-aware Run-Time Power Modelling on Mobile CPUs", Arm Research Summit, Cambridge, September 2017.
- M. J. Walker, S. Bischoff, S. Diestelhorst, G. V. Merrett and B. M. Al-Hashimi, "GemStone: Hardware-Validated CPU Performance and Energy Modelling ", Arm Research Summit, Cambridge, September 2018.

1.7 Authored Software Tools

The work presented in this thesis has lead to the authoring and release of several open-source software tools outlined in this section.

1.7.1 Powmon

The *Powmon* software tools implement the proposed PMC-based power model building methodology (Chapters 4 and 5) and is composed of two separate tools:

***Powmon* Experimental Platform Software** Automatically runs the experiments and records the required data on a Hardkernel ODROID-XU3 hardware platform; and

***Powmon* Model Building Software** Platform-independent software tool that selects the optimum set of PMC events, and formulates and validates the power models.

The software tools, documentation and an interactive result visualisers are found at `http://powmon.ecs.soton.ac.uk`

1.7.2 GemStone

The *GemStone* suite of software tools (Figure 1.1) runs experiments on hardware platforms and the gem5 simulator, implements the methodology for identifying sources of error in full-system simulators, applies power analysis to experimental data, and evaluates performance, power and energy simulation error. The suite is made up of five separate software tools that together implement the full methodology described in Chapter 6:

GemStone Profiler-Logger Software for enabling userspace access to, and low overhead recording of, hardware PMCs on both ARMv7 and ARMv8 devices with any cluster configuration. Repository located at: `https://github.com/mattw200/gemstone-profiler-logger`.

Gemstone Profiler-Automate Automates the running of workloads at any specified set of DVFS levels or core masks while recording PMCs, power[1] and temperature[1]. Unlike the *Powmon Experimental Platform Software*, this tool is more generic, working on both ARMv7 and ARMv8 platforms and also conducts time-series analysis. In addition to logging the data, it can provide it to a program at run-time. Repository located at `https://github.com/mattw200/gemstone-profiler-automate`

[1] if supported by hardware platform

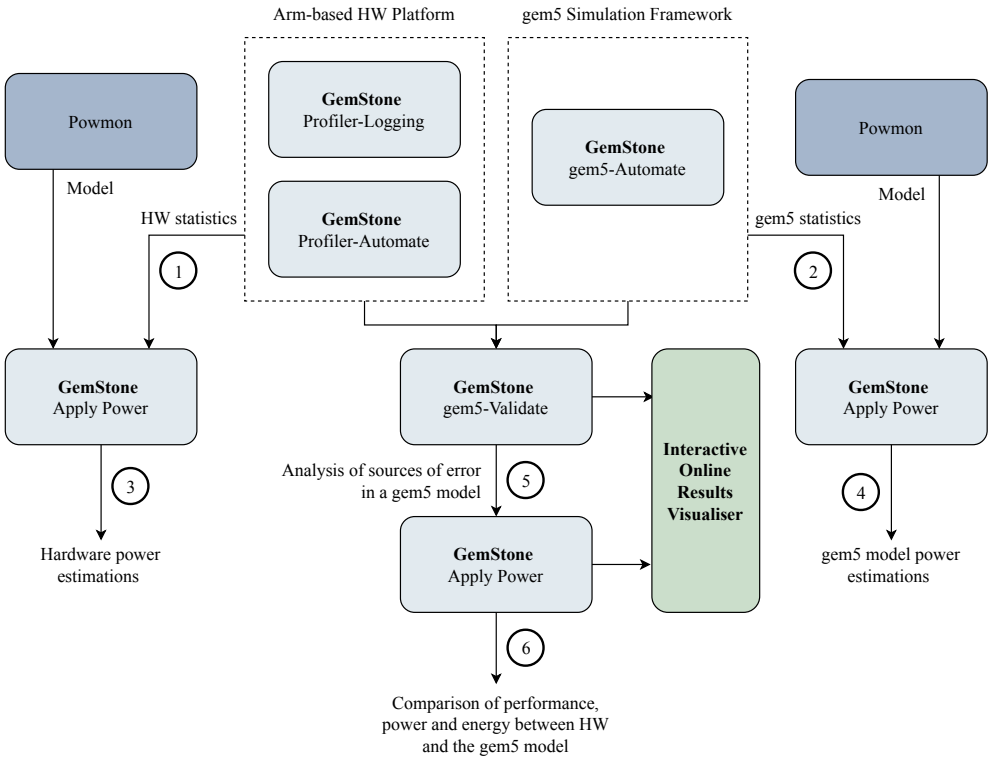


Figure 1.1: Overview of the GemStone suite of software tools

Gemstone Gem5 Auto Automates the running of experiments in gem5, matching the experiments run on hardware using the *Gemstone Profiler-Automate* tool. It allows batch running of workloads on a compute cluster and automatically converts the output data from gem5 into a database of processed results. Repository located at `https://github.com/mattw200/gemstone-gem5-auto`.

GemStone Gem5 Validate Combines the experimental data from both the hardware platform and gem5, before running analysis to identify sources of error in the gem5 model. Repository located at: `https://github.com/mattw200/gemstone-gem5-validate`.

GemStone Apply Power Applies empirical power models developed by *Powmon* to either gem5 simulation data (i.e. collected using *Gemstone Gem5 Auto*) or to recorded data from a hardware platform (i.e. collected using *Gemstone Profiler-Automate*). It can also export an equation formatted for use in the gem5 simulation framework (to provide power estimations at run-time to the simulated program). Furthermore, it conducts power, performance, and energy analysis to both hardware-collected data and simulated data and enables the error between the hardware and the simulator to be analysed, including its scaling across DVFS level and different HMP core types (e.g. *big* vs. *LITTLE*). Repository located at `https://github.com/mattw200/gemstone-applypower` (also contains power models, experimental data, map files for gem5 and hardware data, and voltage-frequency tables for power analysis).

Links to all of the repositories, documentation, step-by-step tutorials, raw results files, and an online results visualiser are available at `http://gemstone.ecs.soton.ac.uk`.

1.8 Thesis Outline

The content of this thesis is organised as follows. Chapter 2 first presents background information on CPU power consumption and CPU power-reduction techniques, before describing mobile CPUs in detail. A review of literature focussing on run-time management, power modelling techniques and full-system simulation is then provided.

Chapter 3 addresses challenges in collecting and analysing power and PMC data on a mobile hardware platform, before presenting the first PMC-based power model for a mobile platform with real data and high workload power variation. A single-core Arm Cortex-A8 system is used to illustrate the approach, which makes makes several improvements to existing approaches, including the use of a change-point detection (CPD) algorithm to extract diverse training data. The accuracy is evaluated on run-time traces and it is shown how a model trained with events averaged across workloads can be effective at predicting real-time power consumption.

An improved power modelling methodology that uniquely considers coefficient stability is presented in Chapter 4, and applied to an 8-core HMP Hardkernel ODROID-XU3 development board. It is experimentally demonstrated that the resulting models make more accurate estimations across a wide range of workloads and workload phases, even if they are not well represented in the training set, when compared to existing approaches. While the approach in Chapter 3 trains a model for each DVFS level, the robust model specification in Chapter 4 uses a single model for all DVFS levels, separates power components, considers the effect of temperature on the leakage current, and allows the data collection time to be reduced from over 40 hours to less than 20 minutes. Problems with existing approaches and practical challenges are highlighted and addressed, including non-ideal voltage regulation and heteroscedasticity.

Methods of modelling the effect of both thermal variation and switching cores offline on the power consumption are presented in Chapter 5. Furthermore, the power consumption is decomposed into its constituting parts, the energy required to move data in the memory hierarchy is analysed, and a temperature modelling methodology is presented.

Chapter 6 presents a methodology for directly comparing full-system simulator models to reference hardware platforms and identifying key sources of error using statistical and machine learning approaches. A method for developing PMC-based power models for use in the gem5 full-system simulator is presented, and the effect of gem5 event errors on the performance, power and energy error is assessed. The performance, power and energy scaling across DVFS levels and CPU types (e.g. *big* and *LITTLE*) between the hardware platform and gem5 model is evaluated. These techniques allow existing CPU performance models to be improved and extended to other CPUs, and they also provide users of simulation tools with insights into how the errors in the models affect their investigations. The integration of empirical power models enables accurate performance, power and energy modelling.

This thesis is concluded in Chapter 7 and potential avenues of future work are presented in Section 7.2.

Chapter 2

Background and Literature Review

In recent years energy-efficiency has become the primary design objective in modern CPUs. From mobile and Internet-of-Things (IoT) devices, where battery-life is limited, to servers and cloud computing, where energy bills and cooling system costs must be minimised, the key design focus is reducing power consumption while maintaining the desired level of performance. Fundamental to improving the energy-efficiency of a CPU is managing it effectively at run-time. Modern CPUs employ various energy-saving techniques, including Dynamic Voltage and Frequency Scaling (DVFS), Dynamic Power Management (DPM) and Heterogeneous Multi-Processing (HMP) cores, which need to be controlled efficiently in order to achieve the desired trade-off between performance and energy while remaining within thermal limits. Fundamental to achieving this is online access to accurate knowledge of the run-time power consumption. Furthermore, the performance of a device is limited by its thermal envelope and therefore, by carefully monitoring the power consumption and considering this thermal envelope, it is possible to achieve a greater peak performance.

In addition, the slow-down of technology scaling means that processor architects cannot rely solely on semiconductor process improvements and must look to innovative design enhancements. Accurate architecture and system-level power modelling is therefore required to evaluate the trade-offs in new designs and for design-space exploration.

Power modelling is therefore required for both run-time management, where it must provide accurate power estimates for run-time control of power management mechanisms and energy-aware scheduling, and design-space exploration, where energy trade-offs in the design and implementation of future systems must be evaluated.

This chapter first provides an overview of how power is consumed CPUs (Section 2.1) and low-level techniques for CMOS power reduction (Section 2.2), before describing system-level power management methods (Section 2.3). Mobile CPU architecture and micro-architecture, including the performance-power trade-offs in different design choices, is presented in Section 2.4 along with a description of the development platform used demonstrate the techniques and methodologies presented in Chapters 4, 5 and 6. A broad range of power modelling approaches are discussed (Section 2.5) before top-down empirical power modelling approaches (Section 2.6) and high-level bottom-up performance and energy simulation tools (Section 2.7) are described in more detail. Finally, Section 2.8 summarises this chapter and discusses the identified challenges that are addressed in the remainder of this thesis.

2.1 CPU Power Consumption

The CPU is implemented in complementary metal-oxide-semiconductor (CMOS), which is also used to implement the DRAM and other systems. The power consumption of a CPU can be broadly split into three key components: static power consumption, P_{static}; short-circuit power consumption, P_{sc}; and dynamic power consumption, P_{dyn} [148, 153]:

$$P_{CPU} = P_{static} + P_{sc} + P_{dyn} \tag{2.1}$$

Dynamic power consumption, P_{dyn}, is caused by the charging and discharging of load capacitances, C_L, when the transistors change state (2.2). It is proportional to the clock frequency, f_{clk}, and the CPU voltage squared, V_{DD}^2. Not every transistor changes state on every clock cycle and so an activity factor, α, is used to indicate the level of activity.

$$P_{dyn} = \alpha C_L V_{DD}^2 f_{clk} \tag{2.2}$$

When a gate changes state there will be a short period of time where both the PMOS and NMOS transistor will conduct due to the non-zero transistor transition time, and a short-circuit current (also known as a crowbar current) will flow, causing short-circuit power dissipation, P_{sc}, (also known as *glitching* power consumption). The short-circuit power for a CMOS inverter can be calculated using [299]:

$$P_{sc} = \frac{f_{clk}\tau}{\beta}(V_{DD} - 2V_{th})^3 \tag{2.3}$$

where V_{th} is the threshold voltage, β is the transistor gain factor, f_{clk} is the switching frequency, and τ represents the signal rise and fall times. P_{sc} is also dependent on the dynamic activity and is therefore sometimes also written as a sub-component of P_{dyn} [149]. Because the overall dynamic power consumption is dominated by P_{dyn}, P_{sc} is often not considered.

While dynamic power remains the dominant consumer, increasingly aggressive technology scaling means that static power is becoming a larger contributor. The static power consumption, P_{static}, is caused by the leakage current, I_{leak} (2.4), which has three major components that are all increasing with technology scaling [204]: sub-threshold leakage current, I_{sub}; gate-oxide leakage current, I_{gate}; and reverse bias band-to-band tunnelling leakage current, I_{btbt} [248]. Other minor components include gate-induced drain leakage (GIDL) and punchthrough current [51].

$$P_{static} = I_{leak} V_{DD} \tag{2.4}$$

$$I_{leak} = I_{sub} + I_{gate} + I_{btbt} \tag{2.5}$$

The sub-threshold leakage current, I_{sub}, is given by [248, 51, 79]:

$$I_{sub} = I_0 e^{\frac{V_{gs}-V_{th}}{mV_T}} \left(1 - e^{-\frac{V_{ds}}{V_T}}\right) \tag{2.6}$$

where V_{gs} is the gate-to-source voltage, V_{ds} is the drain-to-source voltage, m is the sub-threshold

swing coefficient (also known as the body effect coefficient) and where V_T is the thermal voltage:

$$V_T = \frac{KT}{q} \tag{2.7}$$

where q is the magnitude of the electrical charge on the electron, K is the Boltzmann constant, and T is the absolute temperature. I_0 is given by:

$$I_0 = \frac{\mu_0 C_{ox} W (m-1) V_T^2}{L} \tag{2.8}$$

where μ_0 is the zero bias mobility, W is the effective transistor width, L is the effective transistor length, and C_{ox} is the gate oxide capacitance. Note that the threshold voltage, V_{th}, is sometimes written as V_t or V_T (not to be confused with the thermal voltage, V_T).

The power dissipation of CPUs mean that they operate at elevated temperatures. The threshold voltage, V_{th}, decreases with temperature [248]. I_{sub} increases exponentially with temperature due to the increase in thermal voltage, V_T, and the decrease in V_{th} [79, 204].

The gate oxide thickness is decreased with device scaling, which results in more direct tunnelling current through the gate insulator (from the substrate to the gate, and vice-versa); this is known as gate (direct) tunnelling current, I_{gate}, which is made up of two main components: Fowler-Nordheim tunnelling and direct tunnelling. The former of which can be considered negligible under normal operating conditions and the density of the latter can be given by [248, 204]:

$$J_{gate} = A \left(\frac{T_{ox}}{V_{ox}} \right)^2 \exp \left(\frac{-B \left(1 - \left(1 - \frac{V_{ox}}{\phi_{ox}} \right)^{\frac{3}{2}} \right)}{\frac{V_{ox}}{t_{ox}}} \right) \tag{2.9}$$

where A is given by:

$$A = \frac{q^3}{16 \pi^2 \hbar \phi_{ox}} \tag{2.10}$$

and B is given by:

$$B = \frac{4 \sqrt{2 m^*} \phi_{ox}^{\frac{3}{2}}}{3 \hbar q} \tag{2.11}$$

where T_{ox} is the oxide thickness, V_{ox} is the voltage drop across the oxide [248, 257], ϕ_{ox} is the barrier height of the tunnelling electron, $\hbar$ is the reduced Plank's constant and m^* is the effective mass of an electron.

The gate direct tunnelling current, I_{gate}, exponentially increases with the supply voltage, V_{DD}, and reduction in oxide thickness, T_{ox} [249]. Newer devices use high-κ gate-oxide materials, such as aluminium oxide; hafnium oxide; and silicon nitride, instead of silicon dioxide to reduce leakage effects. I_{gate} is not significantly sensitive to changes in temperature as the electric field across the oxide is not strongly temperature-dependent [204].

The band-to-band tunnelling current, I_{btbt}, is due to an electric field across the reverse-biased pn junctions causing the tunnelling of electrons from the valence band of the p region to the conduction

band of the n region [248]. The band-to-band tunnelling current density is given by:

$$J_{btbt} = C\frac{EV_{app}}{E_g^{\frac{1}{2}}}\exp\left(-D\frac{E_g^{\frac{3}{2}}}{E}\right) \tag{2.12}$$

where V_{app} is the applied reverse bias voltage, E is the electric field at the junction, E_g is the energy band-gap, and where C is given by:

$$C = \frac{\sqrt{2m^*}q^3}{4\pi^3\hbar^2} \tag{2.13}$$

and where D is given by:

$$D = \frac{4\sqrt{2m^*}}{3q\hbar} \tag{2.14}$$

Due to the narrowing bandgap, I_{btbt} increases with temperature [204].

While the sub-threshold current, I_{sub}, is the dominant leakage current component at larger feature sizes (e.g. 90 nm and above), the gate-oxide current, I_{gate}, and band-to-band tunnelling current, I_{btbt} are the two most dominant components at 25 nm and below [204].

The higher the power consumption, the larger the amount of heat that must be dissipated. As the temperature increases, the leakage current increases causing power consumption to increase. Therefore a positive feedback effect can occur where an increase in temperature causes an increase in power consumption, which in turn increases temperature. This is known as *thermal runaway* [298, 119]. System-level thermal and power management employs several techniques to avoid thermal runaway (Section 2.3).

2.2 Techniques for CMOS Power Reduction

In recent years, the power and energy consumption of modern processors has been a major constraint, making it a primary design objective. Process and circuit-level techniques, using knowledge of CMOS power consumption (Section 2.1), are employed to reduce power consumption in modern systems.

Section 2.3 describes how some of these techniques can be controlled at run-time by the operating system to trade off performance, power, energy, temperature and reliability.

2.2.1 Dynamic Power Reduction

The most effective way to reduce power consumption is to lower the operating voltage, V_{DD}, as the dynamic power consumption, P_{dyn}, is proportional to V_{DD}^2. However, reducing V_{DD} increases the delay, τ, in the CMOS circuit. While the effect of V_{DD} on the delay is a complex relationship dependent on many variables, a simplified model for it is given by [59]:

$$\tau = \frac{C_L V_{DD}}{\mu_0 C_{ox}(\frac{W}{L})(V_{DD} - V_{th})^2} \tag{2.15}$$

And it can be therefore stated that:

$$\tau \propto \frac{V_{DD}}{(V_{DD} - V_{th})^2} \tag{2.16}$$

The delay in the circuit dictates the maximum clock frequency, f_{clk}, of the device which limits the potential performance.

Static supply voltage scaling is a method of reducing the supply voltage V_{DD} in parts of the circuit. Multiple supply voltages are utilised, with circuits on the critical path having a higher supply voltage than circuits on the non-critical path. Parallelism can be exploited in a design to shorten critical paths, allowing the supply voltage to be reduced. Level shifters are required to interface parts of the circuit operating at different voltages.

Dynamic supply voltage scaling [227, 46, 98, 250] exploits the fact that CPU load is not constant and so V_{DD} and f_{clk} can be lowered when the CPU core does not need to operate at its highest performance. This technique is known as Dynamic Voltage and Frequency Scaling and is controlled at run-time by the operating system (Section 2.3.1).

However, V_{DD} must be set to a level that ensures reliable operation at a given clock frequency in the worst-case operating environment, given:

1. process and manufacturing variation;
2. ambient temperature variation;
3. supply and signal noise variation;
4. accuracy of the voltage regulator.

Because of the quadratic relationship between P_{dyn} and V_{DD}, the voltage margin required to ensure safe operation wastes a significant amount of power. This has motivated the proposal of techniques to automatically reduce V_{DD} until failures occur by dynamically detecting and correcting them [92]. Ambient temperature variation and non-ideal voltage regulation are considered in the development of the power models in this thesis.

Techniques for reducing V_{DD} also reduce the leakage currents (as shown in Section 2.1) and hence the static power consumption, which is becoming more significant in modern devices.

The clock network contributes to a significant proportion of the dynamic power consumption and so clock gating is widely employed. Clock gating disables the clock to a part of the circuit when it is not required to prevent unnecessary switching activity. It can be applied in a fine-grained and coarse-grained manner.

2.2.2 Static Power Reduction

The contribution of the static power consumption is increasing significantly with technology scaling (Section 2.1). Furthermore, the significant popularity of mobile devices, which spend much of their time in an idle or low activity state, mean that reducing the leakage power consumption is crucial. This section outlines some key techniques for reducing leakage power consumption.

Multiple Voltage Assignment Delay paths in a circuit are not uniform and it is therefore possible to supply the gates that are not on the critical path at a lower voltage, effectively exploiting the timing slack in the circuit to save both static and dynamic power [296, 321]. Level shifters are required at the interface between parts of the circuit operating at different V_{DD} levels.

Dual Threshold CMOS (Multi-V_{th}) The leakage current can be reduced significantly by using transistors with a higher threshold voltage, V_{th}, which are often denoted *high-V_t* transistors. However using high-V_t degrades performance. Therefore, modern designs utilise low-V_t transistors in the critical path, and high-V_t transistors on the less timing critical parts of the design [67, 281]. Unlike multithreshold-voltage CMOS (MTCMOS), additional leakage control transistors are not required.

Multithreshold-Voltage CMOS (MTCMOS) Low-V_t transistors are used to implement the gates, while high-V_t *sleep transistors* are used to form virtual power rails [206, 52]. In other words, high-V_t transistors are used to cut off low-V_t circuits from the power rails during sleep mode to reduce leakage current. This is a common approach used for power gating.

Variable Threshold CMOS (VTCMOS) A body biasing technique that controls the effective threshold voltage (at runtime) by applying a substrate bias [160, 128, 321]. It enables the circuit to maintain a low V_{th} when active (maintaining performance), yet have a high V_{th} when inactive, i.e. in a sleep mode (reducing leakage current). When active, no body bias is applied; when inactive a reverse body bias is applied. Standby leakage can be improved by as much as three orders of magnitude by applying this technique [149], although the effectiveness of this reverse body-bias technique is decreasing with technology scaling [211].

Dynamic Vth Scaling (DVTS) Similar to VTCMOS but, instead of changing the V_{th} depending on the mode (e.g. active or standby), it can dynamically vary V_{th} to meet the current performance required. For example, when the maximum performance is required, the lowest V_{th} is used. However, when a lower performance level is adequate, the clock frequency, f_{clk} is decreased and V_{th} increased. When the circuit is in standby, V_{th} can be increased to its maximum level.

Stack Effect Stacking of two or more transistors that are *off* significantly reduces leakage current, though at the expense of delay. *Stack Forcing* can therefore be used to reduce leakage power on non-critical paths [209].

There are many trade-offs to be considered when implementing these techniques into components of a design, such as energy vs. delay, or coarse-grained vs. fine grained approaches. Different implementations of a functionally identical design (e.g. from the same Hardware Description Language [HDL] source files) can therefore have very different power characteristics. This is, of course, in addition to the variation caused by process technology, place and route results, and other physical properties, such as gate-oxide thickness or dielectric constant. It is therefore impossible to accurately predict the power consumption using high-level bottom-up tools that only have high-level CPU properties provided to them. This includes accurately knowing the relative power consumption between different components of the CPU.

2.3 System-Level Power Management

In order to maximise energy-efficiency and satisfy performance and power requirements for specific workloads and conditions, the operation of the CPU must be controlled effectively at run-time.

This section describes the key system-level techniques that are used by the operating system to control the performance, power and temperature, before describing the run-time algorithms presented in research that aim to optimise run-time control.

2.3.1 Dynamic Voltage and Frequency Scaling

Modern CPUs implement a number of energy-saving techniques, perhaps the most common of which is Dynamic Voltage and Frequency Scaling (DVFS), which allows the maximum performance to be traded-off for energy-efficiency. It was shown in (2.2) that the dynamic power consumption is proportional $V^2 f$; reducing the voltage alone results in a quadratic reduction in dynamic power consumption. The maximum clock frequency a CPU can operate at depends how long it takes for a signal to propagate along the critical path (this is subject to many factors, including temperature and process variation). Reducing the supply voltage causes the transistor transition time to increase, and therefore the clock frequency must be reduced to meet the timing requirements and sustain correct operating behaviour. Therefore, if an application running on a CPU does not require the maximum performance, the voltage and frequency can be reduced to save power. Moreover, many applications cannot take full advantage of a processor running at the maximum frequency due to memory *slack*, caused by the latency of reading data or instructions from memory, resulting in stalled cycles. While modern CPUs implement many techniques and features to reduce stalled cycles, including a memory hierarchy, out-of-order execution, speculative execution, Simultaneous Multi-Threading (SMT) (e.g. Intel *Hyper-Threading*), stalled cycles are inevitable. The voltage and frequency can therefore be lowered in such instances to save power without a significant loss in performance. Additionally, in many situations it may be beneficial to sacrifice performance for energy efficiency.

There is an overhead in changing the DVFS level, which must be considered when making RTM decisions. Furthermore, in most systems it is not possible to change the DVFS of each core individually as multiple cores typically share the same power domain. For example, in the case of the Exynos 5422, each core in the cluster must share the same DVFS level (Figure 2.1). This makes the DVFS level selection, and scheduling of tasks, more complex to manage.

2.3.2 Dynamic Power Management

Another common technique is to employ CPU core idle states (coarse-grained power gating), also known as Dynamic Power Management (DPM), where the CPU core is switched into a low power mode where no instructions are executed. While DVFS reduces power, it also typically reduces workload performance, meaning that static power is being consumed for longer periods of time. As technology scaling progresses, a larger proportion of power consumed is static (leakage) power, making it more viable to execute at a high DVFS level to finish the task in a smaller amount of time, and then go into an idle state. The choice of when to employ DVFS scaling or when to use

idle states (i.e., *race-to-idle*) has been the topic of much research [134, 3, 151, 74, 279, 53, 28]. In addition to this trade-off, it also needs to be considered that there is an overhead in changing DVFS level and switching a CPU core to idle and modelling this overhead has also been the subject of research [223]. Chapter 5 measures and models the impact of switching cores offline in the Arm Cortex-A15 cluster of a Samsung Exynos 5422 SoC.

2.3.3 Heterogeneous Multi-Processing Architectures

Heterogeneous Multi-Processing (HMP) architectures, also known as Single-ISA (Instruction-Set Architecture) heterogeneous architectures, such as Arm *big.LITTLE* technology [15], utilise multiple types of CPU cores optimised for different performance-power operating points. For example, Arm big.LITTLE configurations combine CPUs optimised for high-performance (e.g. Arm Cortex-A15, Cortex-A57, Cortex-A72, or Cortex-A73) with CPUs optimised for power-efficiency (e.g. Arm Cortex-A7 or Cortex-A53). This allows a high level of energy-efficiency to be achieved but also a high level of peak performance when demanded by the application. Together with DVFS and idle states, single-ISA heterogeneous architectures [159] provide a large potential for performance, power, energy and temperature optimisation through intelligent management algorithms [152, 324, 49, 105, 283, 55, 175, 242, 176, 154, 154, 158, 203, 68, 93, 138, 237, 254, 129, 268, 205, 335, 194, 115, 87].

Devices with early implementations of big.LITTLE (known as *cluster switching*) could only have a *big* cluster active, or a *LITTLE* cluster active at any one time; no *big* core could be active at the same time as a *LITTLE* core. Later implementations pair up *big* cores with *LITTLE* cores into a virtual core. In this scheme, known as *in-kernel switching* (IKS, or *CPU migration*) only one side of the pair can be simultaneously active. In the later, and more advanced, implementation, known as *Global Task Scheduling* and later as *Heterogeneous Multi-Processing* (HMP), each core, *big* or *LITTLE*, is visible to the scheduler and every core can therefore be active simultaneously, if power and thermal limits allow [140].

More recently, Arm big.LITTLE HMP has been succeeded by *Arm DynamIQ* [7], which allows a single cluster to contain both *big* and *LITTLE* cores with a maximum of eight cores in total. Furthermore, it promises lower latency in power state switching, supports dynamically adapting the size of local memory based on the application requirements, and allows cores within a cluster to potentially have different DVFS levels.

2.3.4 Energy-Aware Scheduling

The task of managing DVFS when running multiple workloads is made complex due to the fact that independent DVFS settings can usually only be applied to CPU clusters, which typically contain 2 or 4 CPU cores. The chosen DVFS operating point is therefore applied to any task running on a particular cluster. The task scheduling decisions and DVFS control must therefore be tightly coupled and support for doing so in Linux has recently been added in the form of the Energy-Aware Scheduling (EAS) enhancement to the Linux power management [8]. EAS extends the Linux scheduler, making it aware of the power/performance characteristics of the CPUs in the system and enables it to optimise energy in big.LITTLE multicore SoCs. A key component of EAS is an accurate run-time energy model to drive the scheduling decisions. With EAS, tasks can be

scheduled to allow optimum thread placement while controlling cluster DVFS levels. A motivational example for EAS is provided in Section 7.2.5.

2.3.5 Maximum Deliverable Power

Another consideration in RTM is the maximum instantaneous power deliverable to the CPU by the voltage regulator. For example, a higher DVFS level may be required by a thread, but in order to increase one cluster to this DVFS level, some cores may first need to be switched offline. This increases the latency when switching DVFS level and therefore changes the threshold at which such a decision should be made.

2.3.6 Thermal Operating Envelope

While intelligent management is important for saving energy, it is also necessary for ensuring that the CPU remains within its thermal operating envelope. The Thermal Design Power (TDP), also known as the Thermal Design Point, is used to define the maximum power a CPU can draw for a thermally significant time period while remaining within the upper limit of the thermal profile to ensure reliability [130]. This depends on the amount of heat the cooling system can dissipate. Note that this is not the maximum power (*peak power*) the CPU can dissipate; it is defined for *real* or *commercially useful* software and it is possible to exceed the TDP using a so called *power virus* (although in practice a thermal control unit should prevent this from occurring by applying thermal throttling). Due to the thermal capacitance in the package and the heatsink, the TDP can be exceeded for a short period of time (i.e., by increasing the DVFS level), therefore providing superior performance. For example, Intel's Turbo Boost technology takes advantage of any thermal headroom available to provide performance when needed [247]. Due to manufacturing variations, different devices operate at different voltages and consume different amounts of current, therefore each CPU has a unique power and temperature characteristic [247, 130]. This shows that modelling, understanding, and managing power effectively allows a high level of peak performance to be achieved.

Modern CPUs even contain dedicated microcontrollers for power management control, for example, the Intel Sandy Bridge microprocessor contains a hardware unit called the Package Control Unit (PCU) that contains state-machines and a dedicated microcontroller to handle power management features [247].

2.3.7 Lifetime Reliability

In addition to maintaining the temperature within safe operating limits, there is also the consideration of life-time reliability [278, 277]. The decreasing feature size from technology scaling and increasing power density reduces processor lifetime due to several wearout mechanisms. The mean time to failure (MTTF) of the processor can be calculated using *Ramp* (Reliability-Aware Micro-Processor) [278] models for five critical wearout mechanisms: electromigration, stress migration, time-dependent dielectric breakdown, thermal cycling, and negative bias temperature instability. Many works have managed run-time power consumption to optimise lifetime reliability [95, 81, 96, 293, 62, 75, 133, 155, 157, 294]. However, these Ramp models are difficult to validate

on real devices and are not necessarily applicable to many real devices which have relatively short lifetime requirements. Furthermore, some Ramp models are only valid in certain scenarios, for example, the model for MTTF due to thermal cycling is only stated to be valid for low frequency thermal cycling (i.e. powering up and down) and not for cycling due to changes in workload behaviour [278]. Some works have proposed alternative metrics for assessing lifetime reliability [241].

2.3.8 Energy-Delay Product

As performance and energy are both critical variables, a metric considering both the execution time and energy consumption known as the energy-delay product (EDP) is commonly used [108]. In situations where the performance is valued more than the energy consumption, variations on this metric are used, for example, ED^2P or even ED^3P. Similarly, the EDP has been adapted to put more emphasis on energy consumption, e.g. E^nDP [329].

2.4 Mobile CPU Architecture and Micro-Architecture

The emergence of smartphones and modern tablet computers was enabled through the development of advanced, energy-efficient mobile processors. Over 90% of smartphone CPUs are implemented with the ARM architecture [100, 9]. In the last few years many smartphones and tablets have shipped with CPUs implementing a 32-bit ARMv7 architecture, with more recent ones implementing a 64-bit ARMv8 architecture. Three mobile CPUs implementing 32-bit ARMv7 architectures, with varying micro-architectures and performance-power trade-off points, were used for experiments in this thesis:

- Arm Cortex-A8. An in-order (InO), dual-issue superscalar CPU with a 13-stage pipeline [10]. This CPU was used in Chapter 3.
- Arm Cortex-A7. An InO, partial dual-issue CPU with an 8-stage pipeline [14], optimised towards power-efficiency. Forms the *LITTLE* core in an Arm big.LITTLE system. This CPU was used in Chapters 4, 5, and 6.
- Arm Cortex-A15. An out-of-order (OoO) 3-way issue CPU with a 15 (integer)/17-25 (floating point) stage pipeline, optimised towards higher performance [13]. It forms the *big* core in an Arm big.LITTLE system. This CPU was also used in Chapters 4, 5, and 6.

The Hardkernel ODROID-XU3 development board [121] was used to illustrate the approaches in Chapters 4, 5, and 6. It uses a Samsung Exynos 5422 SoC, which contains a cluster of four Arm Cortex-A7 cores and a cluster of four Arm Cortex-A15 cores in an Arm big.LITTLE system.

The ODROID-XU3 board supports global task scheduling (GTS), also known as big.LITTLE MP, which allows all of the 8 cores, whether 'big' Cortex-A15s or 'LITTLE' Cortex-A7s, to be available to the operating system's scheduler simultaneously [140]. The maximum clock rate is 2 GHz and 1.4 GHz for the Cortex-A15 and Cortex-A7, respectively, and the device is implemented in a 32 nm Low-Power High-K Metal Gate (HKMG) technology. The SoC also contains an Arm Mali-T626 MP6 GPU and 2 GB LPDDR3 RAM that has a maximum bandwidth of 14.9 GB/s. The development board contains power sensors measuring the power consumed by each cluster (as well as the GPU and memory) at a default sample rate of 3.8 Hz.

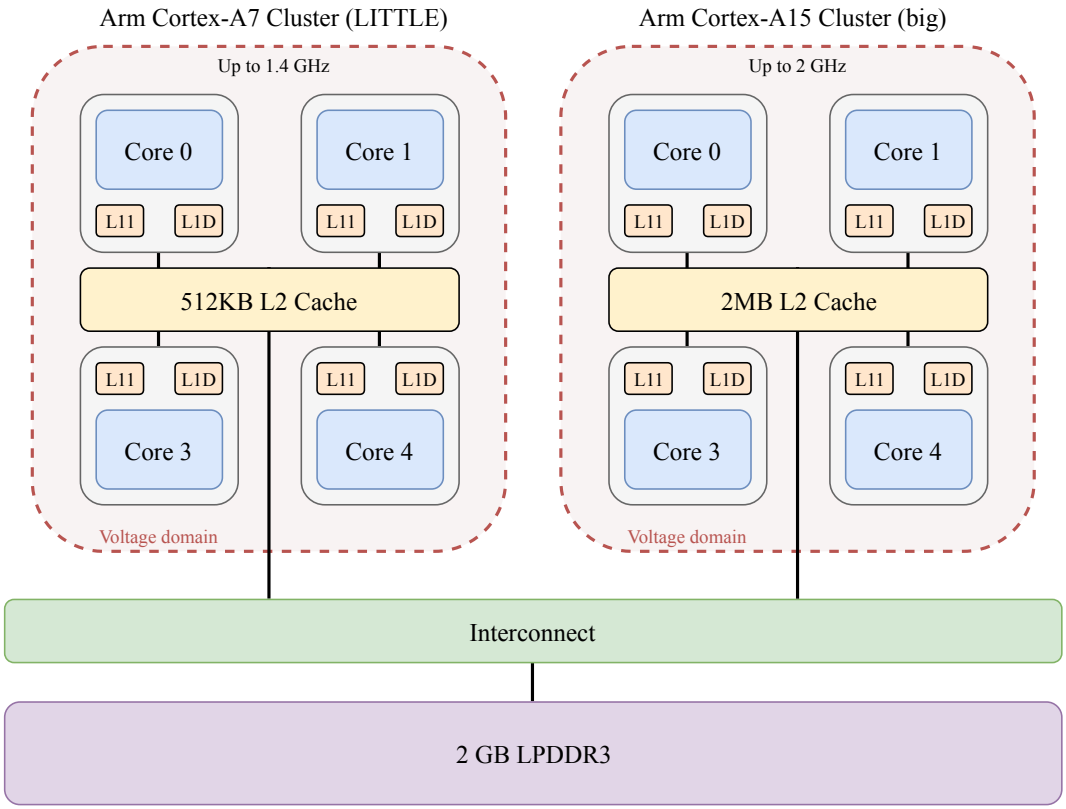


Figure 2.1: Hardkernel ODROID-XU3 CPU Block Diagram

Arm Ltd. does not sell physical chips, but instead sells CPU designs as intellectual property (IP). Companies license these designs and can configure the design to fit their requirements. The remainder of this section will discuss the design of the Arm Cortex-A7 and Cortex-A15 CPUs, as implemented in the Samsung Exynos 5422.

The Arm Cortex-A7 (LITTLE) is a simpler InO design, which means it executes instructions in the order specified by the compiler without any dynamic scheduling. If an instruction stalls due to the data needed to execute it not being immediately available, the whole pipeline stalls, which reduces performance. The core itself can be clock gated, reducing power dissipation while in this state, although significant leakage power is still being consumed and other components in the system are still running while the core has stalled. The Cortex-A15, on the other hand, exploits Instruction Level Parallelism (ILP) by dynamically scheduling instructions based on the availability of input data and can therefore reduce the number of stalls caused by memory latency. For example, if the "next" (in program order) instruction cannot be executed due to its operands not being available, another one that does have its operands available can be executed instead. However, the complexity identifying data dependencies and re-ordering instructions reduces power-efficiency. Furthermore, this complexity also makes the task of estimating the power consumption more challenging.

Both the Cortex-A7 and Cortex-A15 are superscalar designs, meaning they can exploit ILP by issuing multiple instructions per clock cycle. The Cortex-A7 is a partial dual issue (Figure 2.2), meaning that it can execute up to two instructions per cycle. The "Dual Issue" pipe is a partial ALU which allows some integer instructions to be dual-executed. The Cortex-A15 has a 3-issue pipeline, meaning it can execute up to three instructions using its eight execution units per cycle

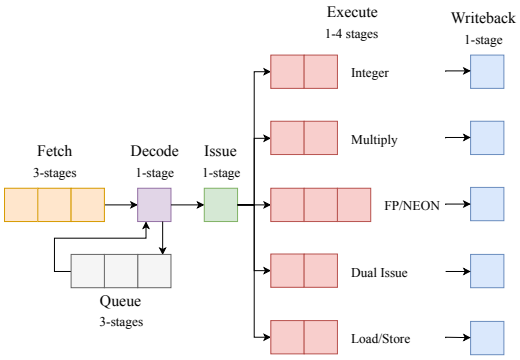


Figure 2.2: The Arm Cortex-A7 Pipeline [139]

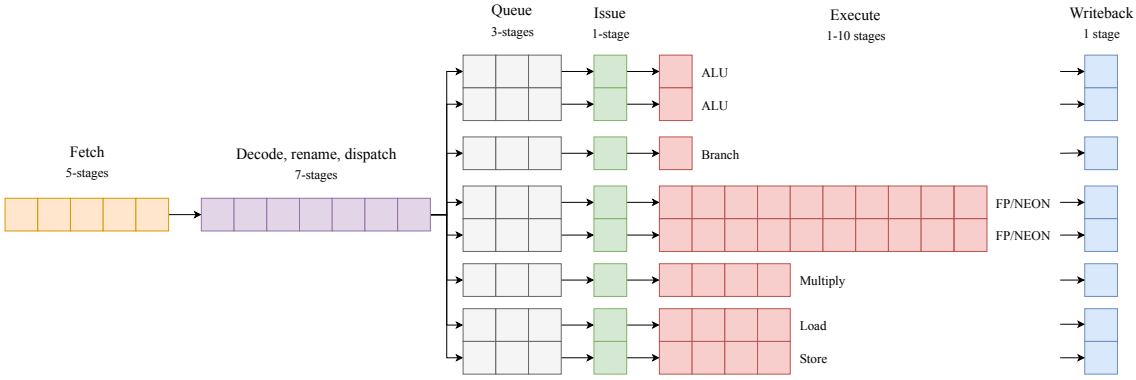


Figure 2.3: The Arm Cortex-A15 Pipeline [112]

(Figure 2.3). However, having a wider issue width adds complexity to the front-end and requires more execution units, which increases power consumption, although this can be partially alleviated by power-gating the execution units that are not in use [125].

Much of the power is consumed through moving data between levels of the memory hierarchy [150, 222]. This implementation of the Cortex-A7 core has a 32 KB two-way set-associative L1 instruction (L1I) cache with a line length of 32 bytes, and a 32 KB four-way set associative L1 data (L1D) cache with a line length of 64 bytes. Both the L1I and L1D caches use a pseudo-random cache replacement policy. There is also an L1 automatic data prefetcher. The cores in the Cortex-A15 cluster also have 32 KB L1I and L1D caches, but they are both two-way set associative, both have a 64-byte line length, and both use a least-recently used (LRU) cache replacement policy. The four Cortex-A7 cores share an eight-way set-associative 512 KB L2 cache that has a line length of 64 bytes and a pseudo-random cache replacement policy. The four Cortex-A15 cores share a 2 MB 16-way set-associative L2 cache with a hardware prefetcher. The larger L2 cache in the Cortex-A15 improves performance at the cost of power. Proposed schemes to reduce leakage consumption include cache resizing [319, 331] and some works have proposed integrating cache resizing with DVFS control [276]. Section 4.3 identifies that monitoring cache access statistics is critical in accurately estimating power consumption in both the Cortex-A7 and Cortex-A15 clusters. Furthermore, Section 5.7 measures the power consumption when accessing different levels of the memory hierarchy in a Cortex-A15 cluster.

The prefetching capabilities of the Cortex-A15 cluster are superior to the Cortex-A7 cluster. While

hardware prefetching improves performance by fetching instructions or data before they are known to be required, there is also wasted energy due to moving data around the memory hierarchy unnecessarily. Many works have focussed on modelling the power cost of prefetching [285] and evaluating the applicability of prefetching in energy-efficient mobile devices [286, 114]. The power consumption of L2 prefetching is observed in Section 5.7.

In order to handle address translation between virtual and physical addresses, the Cortex-A7 has a 2-way set-associative 256-entry unified main Translation Look-aside Buffer (TLB), 10-entry fully-associative micro instruction TLB and a 10-entry fully-associative micro data TLB. The Cortex-A15's memory management unit (MMU) contains a 32-entry fully-associative L1 instruction TLB, two separate 32-entry fully associative L1 TLBs for data load and store pipelines, and a 4-way set-associative 512-entry L2 TLB for each processor [180]. In Chapter 6, where performance models were evaluated against a hardware platform, significant modelling errors in the TLB hierarchy were identified.

The Cortex-A7 prefetch unit (PFU) contains a global branch predictor with a 256-entry pattern history table. It also contains an eight-entry deep Branch Target Address Cache (BTAC) used to predict the target address of certain indirect branches, and contains a four-entry deep Branch Target Instruction Cache (BTIC) which can store up to two instruction cache fetches per entry. The branch predictors in the Cortex-A15 cores are significantly more complex, with a two-level global history buffer with three arrays (taken, not taken and selector). Taken branch targets for indirect branches are cahced in a fully-associative 64-entry micro Branch Target Buffer (microBTB). It has a 2K-entry main BTB. The indirect predictor has a 256-entry BTB [163, 13, 117]. These large tables consume a significant area on the chip and consume large amounts of leakage power.

In ARMv7, unaligned memory accesses are supported in hardware. However, it is costly as the CPU needs to break each memory access into multiple accesses. The time (and energy) penalty is exacerbated when page boundaries are crossed and cache reloads required [193]. Chapter 4 finds that PMC events counting unaligned memory accesses should be used in the power models, and Chapter 6 finds that some errors in a performance simulator model are correlated to the number of unaligned memory accesses.

The simpler design of the Cortex-A7 results in a smaller physical footprint (0.45 mm^2 in 28 nm, incl. NEON, FPU, 32 KB L1 [139]), which reduces leakage power dissipation. Furthermore, the simpler design also means that the power consumption is more predictable.

The differences in the memory hierarchy and micro-architecture have a significant impact on performance and power characteristics. These differing trade-off points can be exploited by an RTM. However, making the right choice is not an easy task and each workload (or, more specifically each workload phase) behaves differently when being migrated from one core to another. Furthermore, the way a workload phase responds to an increase in DVFS level will also be different between the two core types.

The power models presented in this thesis count the occurrence of various micro-architectural events to estimate the power consumption. Chapter 6 proposes a method to identify misspecified micro-architectural details in models of the Arm Cortex-A7 and Arm Cortex-A15 and presents an automated way to aid highlighting and addressing them.

2.5 Power Modelling Approaches

Power modelling is required for different stages of the CPU design process, software and compiler optimisation, and system run-time management. These different use cases have varying requirements.

2.5.1 Power Measurement

The most accurate method of obtaining run-time power consumption is to measure the power consumption of the device directly using specialist test equipment. There are several approaches to obtaining CPU power measurements depending on the device and granularity of power required. For desktop and server computers, the power consumption is often measured from the power sockets, therefore collecting power for the whole system, as opposed to the CPU alone [24, 267, 26]. Some works measure the motherboard power from the ATX (Advanced Technology eXtended) connector using a hall effect current sensor [280] or the voltage drop across sense resistors [275]. However, for accurate power modelling it is important to accurately isolate the CPU power from other devices and to measure the actual voltage being supplied to the CPU. The most accurate results on desktop platforms can be obtained from accessing the CPU voltage regulator itself [274].

Experiments on mobile CPUs are usually carried out on a development board. The most accurate method of estimating power on a mobile development board is to modify the board by placing a sense resistor between the voltage regulator and the CPU, and measuring both current and voltage with a power analyser (as demonstrated in Section 3). As well as providing accurate results, it allows high frequency sampling that can capture individual workload spikes. Some mobile development boards, such as the Hardkernel ODROID-XU3 board [121], include built-in current and voltage sensors which have been utilised in many works, including [309, 48]. However, the power sensors only provide readings at 3.8 Hz (the sensors internally sample at a higher frequency and provide an average) and this must be taken into account for repeatable and accurate results, for example, by running workloads for a long time so enough samples can be collected (this approach is used in Chapters 4 and 5). While it is possible to increase the update rate of the sensors, it comes at the cost of overhead. For system-level power modelling on smartphones, smart battery interfaces have been used to analyse power [2].

While measuring the power directly is accurate, it is not a practical solution for run-time power management as the device needs to either be connected to specialist test equipment (e.g. power analysers) or contain power sensors. It is impractical for devices (particularly mobile devices) to contain such power sensors due to their cost and the large physical area required for them. Furthermore, the power sensors can only measure the power consumption of a power domain, and RTMs require more fine-grained knowledge of power consumption. In this thesis, direct power measurements are used to train empirical, top-down, power models.

2.5.2 Top-Down Modelling Approaches

Alternatively, the power consumption of a CPU can be characterised offline using external power analysis instruments and observing the behaviour of the CPU with an available online metric. A

simple metric that indicates the level of CPU activity is the utilisation, which is an Operating System (OS) statistic describing the amount of time the CPU is spent busy, as opposed to being idle. While utilisation-based power models have been proposed [334, 322, 225, 199, 69, 4, 322], it is a very simple metric and does not indicate the *type* of workload; it does not indicate which, and how many, functional units are active, the number of main memory accesses or the number of cache accesses. Therefore, utilisation-based models provide limited accuracy and are ill-suited to many applications, including estimation of modern multi-core smartphones [333].

The power consumption of a CPU is highly dependent on the operating frequency, voltage and the specific CPU activity. While the voltage and frequency can be easily determined from the OS, the CPU activity depends on the specific CPU workload. Performance Monitoring Counters (PMCs) are registers built into the CPU that can be set to count specific architectural and micro-architectural events, such as the number of integer operations executed or the number of L1 instruction cache misses, and can therefore provide detailed information on the current workload. While principally intended for performance analysis (i.e. identification of performance bottlenecks), PMCs have been widely shown to be effective in estimating the run-time CPU power consumption [24, 267, 26, 71, 34, 244, 280, 309]. Regression analysis is typically used to create a set of linear equations relating several PMC events to the CPU power consumption, however, other techniques such as workload classification [185] and neural networks [2] have also been proposed. PMC-based power models have been implemented for Intel CPUs and power estimations are available through Intel's Running Average Power Limit (RAPL) interface [141]. It uses architectural events from each core, the processor graphics and I/O (input/output) and uses coefficient weights to predict the active power consumption of the package [247]. Several software tools provide access to the power estimation results on supported platforms, including *likwid-powermeter* [291, 219]. Smejkal *et el.* [270] utilise RAPL along with information from the scheduler to provide application-level power estimations. Such approaches that measure the CPU power of an existing device to create power models using other metrics are known as *top-down* approaches, and are described in more detail in Section 2.6.

2.5.3 Circuit, Gate, and Register-Transfer Level Approaches

While carefully developed top-down power models provide excellent accuracy and are inherently validated against hardware, they can only be used for an existing device. When designing and developing a new device, the power and energy-efficiency must be analysed at various points during the design flow before a prototype or final device is available. To accurately model the CPU power consumption, the circuit-level design needs to be considered, meaning that all stages in the design-flow must be completed, including layout and routing. However, physical circuit simulation using SPICE models are unfeasible for large circuits and are therefore typically only used for small sub-components of a CPU design.

PowerMill [126] is an early transistor level current and power simulator for VLSI (very-large-scale integration) circuits using simplified device models to increase simulation speed [42, 326].

Using standard cell libraries supplied by foundries and gate-level netlists is typically several orders of magnitude faster than SPICE-level circuit simulation as only gate-level logic simulations are required. However, errors in power estimation of up to 32% can still occur due to real transistor delay effects and glitches [191]. It should also be remembered that no two CPUs have the same power and performance characteristics due to manufacturing variances.

Gate-level simulation is still too slow for many applications and higher level Register-Transfer Level (RTL) simulation is often used for faster results and to allow designers to evaluate the design and make improvements at an earlier stage in the design flow. Unlike gate-level simulation, RTL simulation does not require the design to be synthesised [45].

Circuit, transistor, gate and register-transfer level techniques require CPU intellectual property (IP) which is typically only available to commercial customers and partners. Furthermore, even RTL simulation requires much processing time even for very small workloads. For this reason, the remainder of this chapter will focus on top-down and higher-level *bottom-up* approaches.

2.5.4 Bottom-Up Modelling Approaches

For CPU and system design exploration, higher level simulation frameworks and component models, such as *gem5*, *CACTI*, and *McPAT*, are widely used in research. These high-level *bottom-up* approaches are not as accurate as top-down approaches or lower-level circuit, gate, or register transfer level approaches. However, they are faster and more accessible than lower-level approaches and are very flexible (i.e. allowing many different architectural and system-level configurations to be tested), unlike top-down approaches. Bottom-up approaches are described in detail in Section 2.7.

2.5.5 Comparison of Power Modelling Approaches

This thesis focusses on two broad applications of CPU power modelling with differing requirements: online (run-time) power estimation, where an operating system requires accurate and instant power estimations at run-time for effective CPU power management; and power simulators for use in deign-space exploration and evaluation of research ideas in both academia and industry.

Run-time power models are required to have a high level of accuracy, a low computational overhead and make use of the limited information available from the hardware and OS at run-time. For research and design-space exploration purposes, the power modelling must be flexible to allow different system configurations and CPU types to be modelled.

Two broad categories of power modelling are therefore considered from this point onwards: top-down models, which are usually created by characterising an existing device and typically implemented with regression techniques; and bottom-up models, which use theoretical principles to estimate power for a user-defined system (Figure 2.4).

Top-down power models are more accurate, as they characterise an existing device using measured data and can typically be implemented using simple equations, making them ideal for use in situations where a low performance and energy overhead is required. The models, however, are only

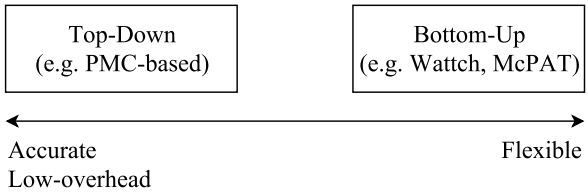


Figure 2.4: Top-down vs. Bottom-Up Power Modelling

valid for the device that have been characterised with, making them less suitable for design-space exploration, where novel architectures need to be explored.

Typical examples of bottom-up power modelling frameworks include Wattch [44], a framework for architectural power analysis; CACTI [263] a cache timing, power and area model; and McPAT [174], a multi-core and many-core system power, area, and timing modelling framework. Such tools require a system specification and an indication of CPU behaviour (e.g., activity statistics) as inputs. A system performance simulator, such as SimpleScalar [47] or *gem5* [30], is typically used to provide architectural and micro-architectural performance statistics to a power simulation tool.

This thesis postulates that empirical, top-down power models are valuable for many design-space exploration tasks and offer significant advantages over bottom-up power modelling frameworks (Chapter 6).

2.6 Empirical CPU Power Modelling

Top-down, empirical power modelling methodologies involve offline characterisation of the power consumption of an implemented device (although some works outlined in this section use simulation data) and using this to train a model. Techniques such as linear regression are typically employed and metrics that can can be obtained by an operating system at run-time are used as input features. The simplest metric to understand the load of a CPU is the utilisation. While most utilisation-based models are aimed towards desktop [21] and server [199, 332] class CPUs, several examples are aimed towards mobile devices [265, 322, 334]. While utilisation is straight-forward to obtain, it only provides the basic load of the CPU, and provides little information on the *type* of load or what functional blocks of the CPU are being utilised.

Previously, top-down regression-based models using performance monitoring counters (PMCs) as inputs have been widely shown to be effective in estimating CPU power [24, 135, 26, 34, 253, 71, 267, 33, 244, 311, 280]. Bellosa [24] presented one of the first works to use PMCs for power monitoring purposes. Isci and Martonsi [135] later presented a technique for combining PMC data with power consumption to provide run-time power estimations for an Intel Pentium 4 CPU. The model attempts to predict the power consumption of hardware units within the CPU using photos of the die layout. However, the estimated power consumption accuracy of individual components cannot be verified. A later work by Contreras and Martonosi [70] propose the use of an Ordinary Least Squares (OLS) model and using a different set of coefficients for each DVFS level. They also present a model for main memory power estimation. PMC events are selected using their correlation with power, although orthogonal events are selected to avoid redundancy. The average error across all tested benchmarks, including MiBench and SPEC2000, was 4%. Bircher [36] *et al.* also use correlation to select PMC events and an OLS estimator, achieving a reported error of 2.6% when validated using SPEC2000. These techniques are later applied to estimating the power consumption of different subsystems (such as main memory, chipset, disk) while considering the "trickle-down" effect of PMCs between different components [33]. Later works by Bertran *et. al* [26, 27] consider multi-core systems and evaluating the responsiveness of the power models on real-time power traces. Chapter 3 presents the first PMC-based power model for mobile devices that also considers the responsiveness.

Energy-efficiency and run-time power optimisation is of particular importance on mobile devices employed in smartphones, tablets and compact, fan-less laptops. However, despite this, there are relatively few works on power modelling for mobile CPUs. As identified in Chapter 3, this is primarily due to the challenges involved in accessing reliable PMC event data on such platforms. Almost all existing works on run-time mobile power modelling have therefore used CPU utilisation (see above), or relied on simulators to provide modelled PMC event data [243, 217] and to, in some cases, also provide modelled power data [244, 253]. The only known existing work (pre-dating the work from Chapter 3 [305]) that developed a PMC-based power model for a mobile Arm-based device using measured power consumption and PMC events was a work published in 2013 by Pricopi et al [238]. They used the propriety Arm Streamline software to extract PMCs from a Versatile Express development platform, which contains a prototype Arm big.LITTLE CPU with three Cortex-A7 and two Cortex-A15 cores. For estimating power, they use a separate equation (with the same input features but different coefficients) for each DVFS level, i.e. a *per-frequency* approach. They observe a narrow minimum-maximum power variation of 1.385-1.506 W for the Cortex-A7 core and 4.535-5.155 W for the Cortex-A15 core, and conclude that a power model for the in-order Cortex-A7 is therefore not required. However, significantly larger power variations were observed in this thesis by employing diverse workloads and using workload phases, as well as ensuring CPU power-saving mechanisms are active.

Many of the existing works, both targeted for mobile [217, 243] and larger systems [267, 33], use a limited set of workloads that do not exercise the CPU in a diverse fashion, which results in a low power variation. This leads to poor models with optimistic reported model accuracies. For example, in a work modelling the power consumption of a Dell PowerEdge R805 SMP server there is only a power range between 285 W and 320 W [327]. Chapter 3 presents the first mobile PMC-based power model (using measured PMC events) that has a high variation in power consumption. Techniques for increasing workload diversity, such as using Change Point Detection (CPD) algorithms to extract workload phases, testing on real-time power traces, and employing micro-benchmarks are employed in Chapters 3 and 4. Furthermore, Chapter 4 experimentally demonstrates how a stable model requires a diverse set of training observations, as well as carefully selected PMC events as input features.

The vast majority of existing works either select PMC events using intuition [243] or use the correlation with power consumption [35, 280, 244, 238], while some works use other techniques, such as Principle Component Analysis (PCA) [327, 177]. Chapter 3 analyses both the correlation and the PMC event rate of each PMC across all workloads (including workload phases) to select PMCs that both correlate with power, but without selecting events that correlate with each other, which causes multicollinearity problems. Chapter 4 demonstrates how multicollinearity significantly impacts the quality of models and presents a novel methodology, employing the variance inflation factor, stepwise regression and event transformations to reduce the multicollinearity.

Most existing works either only consider a single DVFS level or train a separate model for each DVFS level [70, 267], while some works present an equation that works across all DVFS levels [243]. The key advantage of a model that works across all DVFS levels is that it understands the effect of the voltage and frequency on the power consumption and can therefore decompose the power consumption further. However, in order for such a model to be accurate and effective, it must be correctly specified. The model equation must therefore correctly relate the relationships between the power consumption and the CPU voltage, frequency and activity, as indicated by the PMC

events. Chapter 3 develops separate models for each DVFS level (a *per-frequency* model), while Chapter 4 develops a single model that works across all DVFS levels (an *all-frequency* model) [309]. Chapter 4 underlines the importance of model specification, uses statistical significance and the power consumption equations (Section 2.1), and demonstrates the robustness by training the CPU activity components at a single DVFS level and testing on over seven DVFS levels. Earlier works by Bircher and John [32] and Spiliopoulos, Sembrant and Kaxiras [275] use well specified models for desktop and server-class systems. A recent work evaluates existing PMC-based power models for mobile systems and compares *per-frequency* models to *all-frequency* models using the same hardware platform as used in Chapter 4. However, the presented *all-frequency* models achieve very high errors of 10.5% and 20.0% for the Cortex-A7 and Cortex-A15 clusters, respectively. There are several probable reasons for this, such as not considering the sample frequency of the sensors on the platform and not ensuring experimental consistency (Section 4.2), not taking the effects of the voltage regulator into account (Section 4.6), and not thoroughly considering the thermal effects. However, a key reason is that the models have not been specified correctly and the model equations do not capture the relationships between the dependent variable (power) and the independent variables. The models presented in Chapter 4 are the first known (correctly specified) PMC-based mobile power models that use a single model equation for all DVFS levels.

The static power consumption is highly dependent on the temperature (Section 2.1). Singh, Bhadauria and McKee [267] present a PMC-based power model for an AMD Phenom CPU while analysing the effect of temperature as well as shared resources. While temperature is analysed, it is not incorporated into the power model as their system did not contain per-core sensors. A later work by Bircher and John [32] presents a PMC-based power model for a AMD Phenom II (3 GHz) that considers fine-grain temperature and voltage variation. Temperature readings from the temperature sensors are used as inputs to the static power component of the polynomial regression model. The temperature is controlled by adjusting the speed of the processor's fan. Goel and McKee [107] present another PMC-based power model of an Intel Haswell platform that compensates for temperature in the static power equation. Chapter 5 presents a novel approach to compensate for temperature effects on the static power consumption and is the first known work to create a thermally-compensated PMC-based power model for a mobile device.

A recent study by Zhang *et al.* [334] found that most multi-core smartphone models were ill-suited because CPU idle states are not taken into account. They propose a utilisation-based model that considers idle states, however, it does not consider the effect of temperature on the power consumption. As Chapter 5 demonstrates, the difference between idle online core power consumption and idle offline core power consumption is highly temperature dependent due to the leakage mechanisms. The models presented in Chapter 5 include modelling of the idle states, and are the only mobile PMC-based models to do so.

2.7 Full-System Performance and Energy Simulation

While top-down power models are profiled on a specific device, bottom-up power models use theoretical knowledge of each component to model a CPU based on a user's specification. CPU power consumption is determined by the number of transistors switching and the physical properties of the specific transistors switching. To determine this accurately, the simulator needs to know the CPU design (i.e. the size and complexity of each component); which functional units are activated;

the physical properties of transistors used in different parts of the design; and how energy-saving techniques are implemented (e.g. clock gating, power gating, current voltage and frequency, etc.). Creating a tool that is able to take a design specification from a user and to provide realistic and representative power estimations is therefore a challenging task and many abstractions and simplifications are made in practice.

There are typically two stages to bottom-up power modelling:

- an architectural performance simulator first predicts performance and event statistics;
- a power modelling framework then uses these event statistics (indicating activity), along with a CPU specification to estimate the power consumption (Figure 2.5).

The accuracy of the power models are therefore dependent on both the power modelling framework and the performance modelling framework.

2.7.1 Performance Modelling Frameworks

One of the first architectural simulation frameworks that gained widespread use was SimpleScalar [47]. It is open-source and supports several architectures, including Alpha, ARM, PowerPC, and x86, and also supports a wide range of CPU models, from simple in-order to complex out-of-order models. PTLsim [323] is a cycle-accurate full-system simulator that supports the x86 architecture, including CPUs with SMT. However, there is no support for other architectures and, as of writing, the simulator is not actively maintained. OVPsim is a full-system instruction-accurate simulator actively developed by Imperas [220]. Simics is an actively developed commercial full-system simulator that supports Alpha, ARM, MIPS, PowerPC, SPARC, and x86 architectures [313]. MARRSx86 is a widely used cycle-accurate simulator for x86 architectures only [224].

Due to its active design community and support for many ISAs (Alpha, ARM, SPARC, MIPS, POWER, RISC-V and x86), gem5 [30] has become a widely used system-level and micro-architectural simulation framework. It is open-source (BSD license) and was initially created with features from both the M5 full-system simulator [31] and GEMS [187]. The gem5 simulation framework contains both detailed and abstract models for in-order and out-of-order CPUs. For the ARM ISA, gem5 can model up to 64 (heterogeneous) cores of a Realview ARM platform, and boot unmodified Linux and Android with a combination of in-order and out-of-order CPUs [106]. Other advanced features include DRAM modelling [120], support for DVFS [273], GPU modelling [78, 236], Kernel-based Virtual Machine (KVM) support [252] and elastic traces [137]. A distributed version of gem5, called

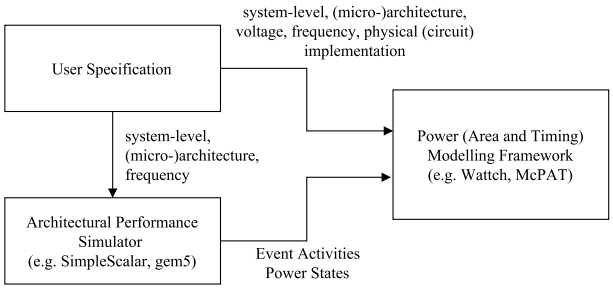


Figure 2.5: Bottom-Up Simulation Workflow

dist-gem5 [200], can model and simulate a distributed computer system using multiple simulation hosts.

While these tools are often used in research, they are not designed to be cycle-accurate detailed micro-architectural models of specific devices. Their accuracy is therefore limited and this should be considered when using them. When more accuracy and reliability is required, RTL models and detailed commercial models, such as Arm CycleModels [6] in the case of Arm products, should be used instead (see Section 2.7.3).

2.7.2 Power Modelling Frameworks

CACTI is a widely used power, area and timing modelling tool for SRAM and DRAM-based caches and memory arrays [263]. It enables researchers to rapidly estimate memory power and energy for evaluation of system trade-offs. More recently, simulators for evaluating new and emerging memory technology have also appeared, such as NVSIM [86] which estimates area, latency and energy of emerging non-volatile memory technologies, including spin-torque-transfer memory (STT-RAM), phase-change random-access memory (PCRAM) and resistive random-access memory (ReRAM). It uses the same empirical modelling methodology as CACTI but adds new features and corrects several false assumptions. The authors claim it therefore models SRAM and DRAM more accurately than CACTI. Vogelsang presents a flexible DRAM power model that can also predict the energy consumption of future DRAM devices by being combined with the DRAM roadmap [302]. It uses device-level details, technology specifications and device switching activity to estimate power. However, the user of a memory modelling tool often does not have a detailed understanding of the memory being modelled that is required by such tools, as memory vendors do not provide public access to memory specifications. A widely used DRAM power estimation technique is to use Micron's DRAM power calculator [196], which uses datasheet and high-level workload specifications. However, it has been found to be inaccurate and insufficient for many scenarios [60]. Therefore, improved DRAM power models have been presented and released as open-source tools [147].

Wattch [44] is an architectural-level CPU power modelling framework which enabled a surge in architectural research after it was released in 2000 [174]. More recently, the McPAT [174] power, area, and timing modelling framework has become widely used by architects for design-exploration of multi- and many-core systems. It models both in-order and out-of-order CPUs, caches, network-on-chips, and memory controllers. McPAT has been coupled with several performance simulators, including Multi2Sim, gem5, Sniper and ZSim, to provide activity statistics for power estimation [1]. However, with high-level simulation tools, accuracy and representativeness of the results is a concern [315, 216] and the next section focusses on the accuracy of bottom-up simulation frameworks.

2.7.3 Accuracy and Validation of Simulation Tools

While such simulation tools are invaluable to research, they inherently contain errors which can impact the conclusions drawn from research, particularly if the sources of error are not well understood. This potentially affects the quality and integrity of research that relies on the tools [304]. Recent works have focussed on these errors and their effects [315, 216, 117, 171, 304].

Three key categories of errors encountered in performance and power simulation tools include [39, 117, 48]:

Modelling Error The device being modelled is understood by the model creator, but the model is incorrectly implemented (e.g. mistakes in source code or algorithm);

Specification Error The model creator does not have access to detailed enough information on the device being modelled;

Abstraction Error The simulator abstracts or simplifies certain components of the system, resulting in a loss of accuracy.

The remainder of this section discusses errors in performance and power simulators, focussing on the widely used *gem5* and *McPAT* simulation frameworks.

Butko *et al.* [50] compare the accuracy of a gem5 model against a hardware device with a dual-core Arm Cortex-A9 CPU, find an average error of between 1.4% and 17.9% and conclude that an overly simple DRAM model is a key source of the error. Endo *et al.* [91] create gem5 models to represent Cortex-A8 and Cortex-A9 CPUs with an average error of 7%. Gutierrez *et al.* [117] analyse sources of error in full system simulation and stress the importance of simulator users understanding the limitations and considering the accuracy of the micro-architectural events as well as the execution time. They model and validate a dual-core Cortex-A15 CPU (with some complex-to-model features, such as prefetching, indirect branch predicting disabled) in gem5 and make improvements to the model to achieve an execution time MAPE of 13% and 17% for SPEC2006 and dual-core PARSEC, respectively. They identify specification error to be the dominant cause of divergence between the model and the hardware platform.

A later work by Butko *et al.* [48] presents an up-to-date gem5 model of an Exynos 5422 SoC. They solve issues with running big.LITTLE systems in gem5, configure model parameters using datasheets and educated guesses, and release their models to the gem5 community. They find an execution time MAPE of 20% using the Rodinia benchmarking suite and validate memory and operation latencies using *LMbench*. They also identify specification error (caused by a lack of detailed information about the device being modelled) as the key source of error, as well as a simplistic DRAM model. They then use McPAT to conduct energy analysis and find a MAPE of 25% when compared with the hardware platform.

Bottom-up power modelling offers more flexibility while top-down power modelling offers superior and validated accuracy. Attempts have been made to combine elements of both techniques to obtain both accuracy and flexibility. Lee *et al.* present *PowerTrain* [171], a learning-based methodology for calibrating McPAT power estimations using a reference hardware platform. A MAPE of 4.4% was achieved when validating on fifteen workloads and employing 15-fold cross-validation. However, it does not correct inaccuracies in modelling of individual components. Other works have focussed on calibrating cycle-accurate performance simulators, such as MARSSx86 [19].

2.8 Discussion

This chapter has provided a background to CPU power consumption, run-time management and modelling, and has presented an overview of the related works found in the literature. This section discusses some of the identified challenges and areas for improvement.

A key challenge identified in the literature review was the difficulty in obtaining reliable PMC event data from mobile CPUs for power modelling (Section 2.6). This is addressed in Chapter 3 through the development of a reliable methodology and experimental setup for extracting PMC events, recording CPU power consumption, and combining the two datasets together. Software tools were also authored and released to enable other researchers to extract PMC data on mobile platforms. This results in one of the first examples of a PMC-based run-time power model for a mobile device, that does not use simulated data.

Another observation in existing works was the low deviation in power consumption when using a limited number of benchmarking suites. Chapter 3 addresses this by employing an offline, unsupervised Change Point Detection (CPD) algorithm to identify workload phases for automatically generating more diverse training data. Chapter 4 improved the power variability further by specifically selecting a diverse set of workloads, manually tuning micro-benchmarks, and using a more complex OoO CPU with modern power management features enabled.

Temperature was found to have a significant effect on static power consumption (Section 2.1) but was only considered in several existing works to varying degrees, and none of these were aimed towards mobile platforms. The novel power modelling methodology presented in Chapter 4 accounts for the effects of temperature changes due to changes in the DVFS level and CPU activity. Chapter 5 presents a unique method of modelling the thermal impact on the static power consumption when using thermal sensors, and proposes a thermal model for situations where there are no on-board thermal sensors (e.g. in simulation tools).

Through the investigations and experiments conducted in Chapters 3, 4 and 5, significant omissions and shortcomings were identified in the previous works reviewed in this chapter. This included model stability, which is addressed in Chapter 4 through the use of correlation analysis, Principle Component Analysis (PCA), Hierarchical Cluster Analysis (HCA), and the Variance Inflation Factor (VIF) to develop an automated PMC event selection approach that is experimentally demonstrated to improve the accuracy of the model when tested on a larger set of more diverse workloads. Another problem identified in existing works was misspecification of models, leading to inaccurate models and incorrect conclusions. This is also addressed in Chapter 4 through a careful model specification that only has to be trained with the full set of observations at one DVFS level to work across the full range of DVFS levels. Another factor not considered in existing works was the fact that the voltage being supplied by the non-ideal voltage regulator varied with CPU load, which affected the power consumption significantly. This was also addressed in Chapter 4 by presenting a methodology for modelling the voltage fluctuations. Furthermore, the problem of heteroscedasticity was identified and addressed in Chapter 4, through the use of a heteroscedasticity consistent standard error (HCSE) estimator.

Literature reviewed in Section 2.7 identified inaccuracies in the simulation tools used by many researchers in design-space exploration and system optimisation. Many works combine a performance simulation framework (e.g. gem5) with a power modelling framework, such as McPAT. Particularly significant errors are identified in power simulation frameworks. Sections 2.1 and 2.2 described how the CPU power consumption is dependent on a vast number of physical properties and power reduction mechanisms that are applied differently to various components in the system. Section 2.4 highlighted just a few of the complex micro-architectural trade-offs that affect the performance and power characteristics. A high-level bottom-up power modelling framework, that takes only high-level CPU properties as inputs (in addition to the switching events provided by the performance

simulator), is therefore limited in its ability to estimate the power consumption, including the relative power consumption between different components within the system. Existing works identified *specification* error to be a key problem in simulation frameworks, as the specific details of the device being modelled are often not known. Chapter 6 addresses this by presenting a methodology for comparing a model directly against a reference hardware platform and identifying the sources of error. Furthermore, Chapter 6 proposes using a top-down empirical model in conjunction with a (hardware-validated) performance simulator for more accurate system modelling.

Chapter 3

Run-Time Power Modelling of Mobile CPUs

Significant energy savings can be made in mobile devices using intelligent run-time management (RTM) software that can effectively trade off CPU performance and energy by controlling energy-saving techniques. For the RTM to make informed decisions, it needs accurate and responsive run-time knowledge of the power consumption, without incurring significant performance and power overheads. Performance Monitoring Counters (PMCs) are small registers in the CPU that count various architectural and micro-architectural events, providing a detailed picture of the current CPU operation with little overhead in reading them. For desktop and server systems, hardware performance monitoring counters (PMCs) have been used for run-time power estimation [267, 71, 21, 244, 27, 33]. However, for mobile devices, where energy-efficiency is of particular importance, there are few works on empirical, PMC-based, power modelling.

There are two key reasons for this:

1. The challenges in accurately measuring run-time power consumption on mobile development boards;

2. The difficulty in extracting PMC data from mobile devices.

This has led to existing works using utilisation-based models [334, 322], combining power measurements from a real device with modelled PMC events from a performance simulator [243], or using events instrumental in RTL with Synopsys PrimeTimePX for estimating power consumption [217].

This chapter addresses these challenges by developing: 1. a reliable method of extracting PMCs from mobile devices; and 2. an experimental setup for accurately measuring run-time power consumption. A power modelling methodology, based on existing techniques, is then applied to this setup, resulting in the first empirical, PMC-based, run-time power model for an Arm-based mobile platform (the only exception known to the authors being [238] by Pricopi *et al.*). Furthermore, the limitations of existing approaches are highlighted and demonstrated, before being explored further in Chapter 4, which presents an improved methodology that is demonstrated on a big.LITTLE platform.

The contributions of this chapter are as follows:

1. A methodology and released software tools for extracting PMC events from a mobile device;
2. A robust experimental method for measuring run-time CPU power consumption and synchronising it with PMC event data;
3. The implementation and application of an offline, unsupervised change point detection method for extracting workload phases for generating diverse training observations;
4. Analysis of PMC event correlation, showing that the common approach of selecting events using their correlation with power to be flawed (this is investigated further in Chapter 4);
5. Accurate mobile power models with cross-validated MAPEs of $< 2.0\%$ that are demonstrated to be responsive when evaluated against real-time power traces (therefore showing that responsive power models can be developed using average workload data, providing the workloads are diverse).

3.1 Multivariate Linear Regression

Multivariate regression is used in this thesis, as well as many of the related works described in Section 2.6, to develop a set of linear equations relating PMC events, voltage, and frequency to the instantaneous power consumption.

3.1.1 Definition

Multivariate linear regression is a method of modelling the relationship between a dependent variable y and multiple independent variables (also known as predictors, regressors or features), $x_{i,1}...x_{i,p-1}$, using a linear predictor function:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_{p-1} x_{i,p-1} + \epsilon_i \tag{3.1}$$

$$i = 1, ..., n \tag{3.2}$$

where n is the number of observations; $p-1$ is the number of predictors, or features, excluding the intercept (there are p regression parameters); y_i is the dependent, or response, variable; $x_{i1}, x_{i2}, \dots x_{i,p-1}$ are the predictors, or features (independent variables); $\beta_0 \dots \beta_{p-1}$ are the regression coefficients (weights); and ϵ is the random error term, or disturbance term.

It is mathematically convenient to let $x_{i0} = 1$ for the intercept:

$$y_i = \beta_0 1 + \beta_1 x_{i1} + \beta_2 x_{i2} + \dots + \beta_p x_{ip} + \epsilon_i \tag{3.3}$$

It can also be written in matrix form:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{3.4}$$

Where $\boldsymbol{\epsilon}$ is the unobserved random error and $\boldsymbol{X}$ is an $n \times p$ matrix of predictors, known as the *design matrix*. The linear predictor function is also known as the *hypothesis*, $h_\beta(x)$. Note that in machine learning texts, $\boldsymbol{m}$ is often used to represent the regression weights.

$$\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}, \boldsymbol{X} = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix} = \begin{pmatrix} 1 & x_{1,1} & \dots & x_{1,p-1} \\ 1 & x_{2,1} & \dots & x_{2,p-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n,1} & \dots & x_{n,p-1} \end{pmatrix} \tag{3.5}$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_{p-1} \end{pmatrix}, \boldsymbol{\epsilon} = \begin{pmatrix} \epsilon_1 \\ \epsilon_2 \\ \vdots \\ \epsilon_n \end{pmatrix} \tag{3.6}$$

Regression analysis is also considered to be a supervised machine learning technique and terminology from the field of machine learning is often used in regression problems (e.g. *features* instead of *independent variables* or *predictors*). A common definition of machine learning is as follows (paraphrased from [251]):

> A field of study that gives computers the ability without being explicitly programmed.

A newer, and more formal, definition was provided by Tom Mitchell in 1997 [198]:

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

3.1.2 Estimating Coefficients

In the case of regression, an *estimator* is used to *learn* the values of the coefficients ($\boldsymbol{\beta}$) using training data. The hypothesis can then be used to make predictions and forecasts using these coefficients. The estimator aims to minimise the fitted line and the data points; the ordinary least squares (OLS), also known as the linear least squares, estimator minimises the sum of the squares of the difference between the observed dependent variable and the values predicted by the hypothesis (residuals). The sum of squares of the residuals is often known as the residual sum of squares (RSS), the sum of squared errors (SSE), or the sum of squared residuals (SSR).

A residual, e_i, is calculated by:

$$e_i = y_i - \hat{y}_i \tag{3.7}$$

Or, in matrix form:

$$\boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}} \tag{3.8}$$

As an aside, note that the residual, e_i is the *observed* error, which is distinct from the unknown true error, ϵ_i [162]:

$$\epsilon_i = y_i - E\{y_i\} \tag{3.9}$$

Therefore, the residual sum of squares, RSS, is calculated by:

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} e_i^2 = \boldsymbol{e}^T\boldsymbol{e} \tag{3.10}$$

These are used to derive the normal equation, which is used to calculate the coefficient estimates, $\hat{\boldsymbol{\beta}}$:

$$\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}} \tag{3.11}$$

Therefore:

$$\boldsymbol{e}^T\boldsymbol{e} = (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})^T(\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}) \tag{3.12}$$

This is then expanded:

$$\boldsymbol{e}^T\boldsymbol{e} = \boldsymbol{y}^T\boldsymbol{y} - \boldsymbol{y}^T(\boldsymbol{X}\hat{\boldsymbol{\beta}}) - (\boldsymbol{X}\hat{\boldsymbol{\beta}})^T\boldsymbol{y} + (\boldsymbol{X}\hat{\boldsymbol{\beta}})^T(\boldsymbol{X}\hat{\boldsymbol{\beta}}) \tag{3.13}$$

$$\boldsymbol{e}^T\boldsymbol{e} = \boldsymbol{y}^T\boldsymbol{y} - 2(\boldsymbol{X}\hat{\boldsymbol{\beta}})^T\boldsymbol{y} + (\boldsymbol{X}\hat{\boldsymbol{\beta}})^T(\boldsymbol{X}\hat{\boldsymbol{\beta}}) \tag{3.14}$$

$$\boldsymbol{e}^T\boldsymbol{e} = \boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{X}^T\hat{\boldsymbol{\beta}}^T\boldsymbol{y} + \boldsymbol{X}\boldsymbol{X}^T\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T \tag{3.15}$$

The objective is to minimise the RSS. The above equation is a convex function and so the derivative with respect to $\hat{\beta}$ needs to be found in order to find the global minima:

$$\frac{\partial}{\partial\hat{\beta}}\boldsymbol{e}^T\boldsymbol{e} = 0 \tag{3.16}$$

$$\frac{\partial}{\partial\hat{\beta}}\boldsymbol{e}^T\boldsymbol{e} = \frac{\partial}{\partial\hat{\beta}}\left[\boldsymbol{y}^T\boldsymbol{y} - 2\boldsymbol{X}^T\hat{\boldsymbol{\beta}}^T\boldsymbol{y} + \boldsymbol{X}\boldsymbol{X}^T\hat{\boldsymbol{\beta}}\hat{\boldsymbol{\beta}}^T\right] = 0 \tag{3.17}$$

Using the identity [229]:

$$\frac{\partial a^T b}{\partial a} = b \tag{3.18}$$

Results in the following:

$$-2\boldsymbol{X}^T\boldsymbol{y} + 2\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} = 0 \tag{3.19}$$

Which can be rearranged to give:

$$\boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{X}^T\boldsymbol{y} \tag{3.20}$$

The inverse of a matrix $\boldsymbol{A}$ can only exist if $\boldsymbol{A}$ is square and non-singular. Therefore, in inconsistent systems (such as least squares regression) a pseudo-inverse, which finds an inverse-like (and unique) matrix for any matrix, is used [182]. The matrix $(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T$ is the Moore-Penrose inverse ($\boldsymbol{A}^+$),

which is a pseudo-inverse (also known as the generalised inverse):

$$\boldsymbol{A}^{+} = \boldsymbol{A}^{T}(\boldsymbol{A}\boldsymbol{A}^{T})^{-1} \tag{3.21}$$

Multiply each side by the $(\boldsymbol{X}^{T}\boldsymbol{X})^{-1}$:

$$(\boldsymbol{X}^{T}\boldsymbol{X})^{-1}\boldsymbol{X}^{T}\boldsymbol{X}\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^{T}\boldsymbol{X})^{-1}\boldsymbol{X}^{T}\boldsymbol{y} \tag{3.22}$$

Gives the *normal equation* [162]:

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^{T}\boldsymbol{X})^{-1}\boldsymbol{X}^{T}\boldsymbol{y} \tag{3.23}$$

The normal equation allows $\hat{\boldsymbol{\beta}}$ to be solved analytically.

Assuming no correlation between the predictors, this effectively decomposes the multivariate problem into many univariate problems [97]; the pseudo-inverse is only valid if the elements of $\boldsymbol{X}$ are linearly independent (i.e. there is no multicollinearity present), as linear dependence would mean that $\boldsymbol{X}\boldsymbol{X}^{T}$ would have no inverse. The importance of this, and methods of reducing multicollinearity, are discussed later in this chapter and further in Chapter 4.

3.1.3 Fitting Values

The vector of fitted, or predicted, values, $\hat{\boldsymbol{y}}$, can be found using:

$$\hat{\boldsymbol{y}} = \boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{H}\boldsymbol{y} \tag{3.24}$$

Where:

$$\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}\boldsymbol{X}^{T})^{-1}\boldsymbol{X}^{T} \tag{3.25}$$

$\boldsymbol{H}$ is the *hat matrix* (it "puts a hat on $\boldsymbol{y}$"), also known as the *projection matrix*, $\boldsymbol{P}$. The matrix $\boldsymbol{H}$ is symmetric and has the property of *idempotency* [162]:

$$\boldsymbol{H}\boldsymbol{H} = \boldsymbol{H} \tag{3.26}$$

It can therefore be shown that the fitted values have mean $E(\hat{\boldsymbol{y}}) = \mu$ and variance-covariance matrix $var(\hat{\boldsymbol{y}}) = \boldsymbol{H}\sigma^{2}$ [245]

Remembering that the residuals, $\boldsymbol{e}$ are:

$$\boldsymbol{e} = \boldsymbol{y} - \hat{\boldsymbol{y}} = \boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}} \tag{3.27}$$

Therefore:

$$\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{H}\boldsymbol{y} = (\boldsymbol{I} - \boldsymbol{H})\boldsymbol{y} \tag{3.28}$$

Where, $\boldsymbol{I}$ is the identify matrix of $\boldsymbol{H}$. Importantly, the matrix $\boldsymbol{I} - \boldsymbol{H}$ is symmetric and idempotent. The matrix $\boldsymbol{I} - \boldsymbol{H}$ is sometimes referred to as the *annihilator*, $\boldsymbol{M}$.

Therefore, the residuals can be given by:

$$\boldsymbol{e} = \boldsymbol{M}\boldsymbol{y} \tag{3.29}$$

The residuals, $\boldsymbol{e}$, are used in Chapter 4 to validate the empirical power models and identify the problem of heteroscedasticity in PMC-based power modelling. They are also used in Chapter 5 to measure effect of temperature variation on the power modelling error.

3.1.4 Analysis of Variance

The variance-covariance matrix (also known simply as the covariance matrix) of $\boldsymbol{e}$, $\boldsymbol{\sigma}^2\{\boldsymbol{e}\}$, also denoted $var(\boldsymbol{e})$, is:

$$var(\boldsymbol{e}) = \boldsymbol{\sigma}^2(\boldsymbol{I} - \boldsymbol{H})(\boldsymbol{I} - \boldsymbol{H})^T = \boldsymbol{\sigma}^2(\boldsymbol{I} - \boldsymbol{H}) \tag{3.30}$$

(remembering that $\boldsymbol{I} - \boldsymbol{H}$ is idempotent and symmetric).

This can be estimated as:

$$\boldsymbol{s}^2\{e\} = MSE(\boldsymbol{I} - \boldsymbol{H}) = \frac{\boldsymbol{y}^T(\boldsymbol{I} - \boldsymbol{H})\boldsymbol{y}}{n - p} \tag{3.31}$$

s^2, is the OLS estimate for σ^2 and $\sqrt{s^2}$ is the standard error of the regression (SER), also known as the standard error of the estimate. It is sometimes used as a measure of *goodness-of-fit* as it is the average distance between the observed values and the regression line.

It is already known that the residual sum of squares, RSS (also known as the SSE) is given by:

$$RSS = \boldsymbol{e}^T\boldsymbol{e} = [(\boldsymbol{I} - \boldsymbol{H})\boldsymbol{y}][(\boldsymbol{I} - \boldsymbol{H})\boldsymbol{y}]^T = \boldsymbol{y}^T(\boldsymbol{I} - \boldsymbol{H})\boldsymbol{y} \tag{3.32}$$

The total sum of squares TSS is given by:

$$TSS = \boldsymbol{y}^T\left[\boldsymbol{I} - \frac{\boldsymbol{J}}{n}\right]\boldsymbol{y} \tag{3.33}$$

where $\boldsymbol{J}$ is an $n \times n$ matrix of 1s.

The explained sum of squares (ESS), also known as the sum of squares due to regression (SSR), is given by:

$$ESS = \boldsymbol{y}^T\left[\boldsymbol{H} - \frac{\boldsymbol{J}}{n}\right]\boldsymbol{y} \tag{3.34}$$

Note that:

$$TSS = ESS + RSS \tag{3.35}$$

3.1.5 Coefficient of Determination

A typical way of assessing how well the model fits the data is to use the R^2 value, which is also known as the coefficient of determination (coefficient of *multiple* determination in the case of multiple

regression). The R^2 value is always ≥ 0 and ≤ 1 and is sometimes expressed as a percentage. The closer the R^2 is to 1, the greater the degree of linear association between the independent variables $\boldsymbol{X}$ and the dependent variable $\boldsymbol{y}$.

The R^2 value can be described as the fraction of the response variation that is explained by a linear model:

$$R^2 = \frac{\text{Explained variation}}{\text{Total variation}} = \frac{ESS}{TSS} = 1 - \frac{RSS}{TSS} \tag{3.36}$$

where:

- TSS is the *total sum of squares*;
- ESS is the *explained sum of squares*, also known as the regression sum of squares (SSR); and
- RSS is the residual sum of squares, also known as the error sum of squares (SSE) or the sum of squared residuals (SSR).

While the R^2 value is widely used, it has limitations and is often misused due to its convenience and simplicity [161, 89, 104]. For example, the R^2 value is a relative quantity (it indicates how large ESS is, relative to TSS) and therefore a high R^2 may not indicate a good fit in some situations while in others, a relatively small R^2 may indicate a good fit [104]. The R^2 can also be inflated or deflated by outliers [89]. A high R^2 does not necessarily indicate good predictions can be made from the model, nor does it necessarily indicate that the estimated regression line is a good fit [161]. The R^2 value should therefore be used alongside other metrics and analysis methods to determine the suitability and quality of a model. These are discussed later in this chapter.

Furthermore, adding more independent variables to the regression model can only increase R^2 and never reduce it [161]. Therefore the R^2 value can be inflated by having too many independent variables, even if they are not statistically significant. Therefore another metric known as the *adjusted* R^2 (denoted R^2_a or $\overline{R}^2$), which compensates for the number of predictors in the model, is often used [104]:

$$\overline{R}^2 = 1 - \frac{n-1}{n-p-1}(1-R^2) \tag{3.37}$$

where n is the number of observations and p is the number of predictors (independent variables). Each time a predictor is added to the model, $\overline{R}^2$ only increases if the new predictor causes the model to improve by more than would be expected by chance alone.

3.1.6 Assumptions of OLS

Due to its perceived simplicity and ease of implementation using many standard software tools (e.g. *SPSS*, *R*, *Matlab*, *SAS*, *Microsoft Excel*, *Statistica* and *STATA*), regression modelling is widely used (and misused) [104].

While often overlooked in CPU power modelling, the estimation techniques make several assumptions that must be considered, including [104, 102]:

1. correct specification of the model;
2. the relationship between the independent variables and the dependent variable must be linear;

3. the conditional mean must be zero: $E(\epsilon|\boldsymbol{X}) = 0$, for example, independent variables must not be correlated with error.

4. the elements of β are constants;

5. normality of the error distribution: $\boldsymbol{\epsilon}|\boldsymbol{X} \sim \mathcal{N}(0, \sigma^2\boldsymbol{I})$. Note that violating this assumption does not invalidate the OLS method, however, if the error terms are not normally distributed then the standard errors become unreliable and the fit will not be BLUE (Best Linear Unbiased Estimator);

6. the independent variables in $\boldsymbol{X}$ must be linearly independent (i.e. there must be no multi-collinearity). I.e. $\boldsymbol{X}$ must be an $n \times p$ matrix of full rank. This assumption is also known as the identification condition.

7. there must be no auto-correlation, e.g. in time-series data, a value must not be dependent on a previous (historic) value;

8. constant variance (homoscedasticity): $E[\boldsymbol{\epsilon}^2|\boldsymbol{X}] = \sigma^2\boldsymbol{I}$.

These assumptions, and the consequences of violating them, are discussed in both this chapter and later in Chapter 4. For example, Section 4.3.2 discusses multicollinearity, Section 4.4.1 discusses correct model specification, and Section 4.4.2 discusses testing for non-linearity and correlation with error, normality of the error distribution, and the assumption of homoscedasticity.

3.1.7 Alternative Modelling Methods

While this thesis and many of the existing works employ simple linear models, typically using an OLS estimator (see Chapter 2), some more recent works have employed non-linear and non-parametric approaches, such as neural networks or k-nearest neighbours regression, to estimate both the CPU power consumption using PMCs [118, 65] and mobile system power estimation using higher-level statistics [84, 85].

Carvalho, Cunha and Silva-Filho [54] use k-nearest neighbours (k-NN) regression to estimate the CPU power consumption. It is a non-parametric approach, contrasting to the parametric OLS approach. While such non-parametric approaches do not have the assumptions that parametric approaches have, and are therefore useful if there is no prior knowledge of the system being modelled, they require more training data, have a higher training overhead, and are at a higher risk of overfitting. In cases where a parametric model would be suited, a non-parametric has less statistical power, meaning that conclusions can not be drawn from the model with the same degree of confidence, unless more training data is used. While the k-NN approach is simple, flexible and easily implementable, it is not well suited for run-time management because the computation cost all occurs in the prediction, and not in offline training; it uses *lazy learning*. Furthermore, the history of observations must be stored, consuming memory. They show a linear regression model to perform poorly when estimating the normalised energy for different applications when the DVFS is changed. However, while the frequency is used in the model, the voltage is not, and the authors do not provide details of precisely what features are used or the model equation. While non-linear approaches will be able to *overfit* the voltage-frequency relationship, a linear model will not, unless the correct inputs are used, i.e. either V_{DD}^2 would need to be provided as an input feature, or polynomial values of f_{clk} would need to be provided to approximate the voltage-frequency relationship.

Mair, Huang, and Eyers [184] use a k-NN approach for PMC-based power modelling of server-class systems. They overcome the expensive run-time computational cost by deploying the k-NN on a remote system and the cluster node periodically sends its PMC data across the network and receives the power estimate. However, such an approach is impractical for a mobile device, where the energy and performance cost of using the network to send and receive data is far greater than the cost of reading PMC registers and applying a simple linear equation. They also raise concerns about the common practice of training and validating with typical workloads and how this limits the applicability and extendability of the resulting power models. This problem is raised, analysed and addressed in Chapter 4.

Gutierrez, Tamir and Qasem [118] develop a neural-network to predict the power consumption of desktop-class CPUs using PMC events and compare it to a linear regression model. They highlight the challenges in training a neural network and the potential high variance but find that the accuracy of the neural network improved to be superior with more training observations while the accuracy of the linear regression stopped improving after a certain number of training observations were used. However, the five PMC events were chosen by selecting the five counters with the highest correlation with power, which is demonstrated to result in poor models in Chapter 4. The authors conclude that a linear regression model is adequate for most applications, particularly if only a small number of samples are available, while a neural network can be used to achieve a higher accuracy if a lot of training data is available. The models do not consider multiple DVFS levels, voltage or temperature.

Chen *et al.* [65] present PMC-based power models for heterogeneous cloud-edge data centres using linear regression, a Support Vector Machine (SVM), and a Neural Network (NN). It follows the previously published work from Chapter 4 ([309]) and presents a similar automated hardware performance counter selection method. They find the SVM and NN models to achieve a significantly lower error than the linear regression model, when tested with the same data used for training. However, their linear regression model for an Arm Cortex-A15 achieved an average error of over 10% despite it only considering a single DVFS level and 16 workloads, whereas the model presented in Section 4 for the same CPU achieved an average error of 2.8% across a large set of diverse workloads and with a single model for multiple DVFS levels. Furthermore, while they find that the NN and SVM models achieve better errors when tested with the training data, they also report that those models obtain higher errors than the linear regression model when tested with unseen workloads. This is an example of where an overly complex, non-linear model can easily overfit the training data. Furthermore, even the lowest errors achieved by the NN and SVM are $> 5\%$, which is larger than the errors achieved in Chapter 4 with a more rigorous validation methodology. In short, their linear model is underfitting and their non-linear models are overfitting. Model misspecification and non-optimal PMC event selection are possible causes of the linear model underfitting.

It is general good practice to attempt linear models first, and only resort to non-linear models when the relationship cannot be accurately modelled and non-linearity is observed in standard diagnostic tests, such as inspection of the residuals [161]. Non-linear models are more difficult to interpret, in particular, the effect of each feature on the response is less intuitive to understand. P-values are impossible to calculate for the features, and the R^2 is not valid. Some works claim that linear regression is not suitable because the power equation relationships are non-linear [84, 123]. This view comes from a misunderstanding of what the term "linear" means in the context of a linear estimator, such as OLS. A linear model can be used to model a non-linear equation, such

as V^2 or $\log(T)$; the estimator is linear, but the relationships defined in the model do not have to be linear. In the same way that employing a large number of high-order polynomials in a linear model will likely improve apparent model accuracy through over-fitting the relationships in the training data, employing a more complex non-linear model in a situation where a linear model is sufficient, increases the potential for overfitting. For example, it was found in Chapter 4 that it was straight-forward to over-fit the thermal compensation with higher order terms relating to V_{DD} and f_{clk}, despite there being over 1800 training observations; adding all combinations of V_{DD}, V_{DD}^2 and f_{clk} improved accuracy, but inspection of the p-values showed how many of these terms were not statistically significant. A non-linear model would likely over-fit this relationship, while linear regression allows the p-values, domain-specific knowledge and intuition of the relationship being modelled, and interpretation of the resulting coefficients to achieve a model with low variance that generalises well.

Using linear approaches, analysing the resulting statistics and applying appropriate diagnostic checks can allow new findings about the system being modelled. For example, in developing the model for Chapter 4 a large background dynamic component, that was independent of the PMC event activity (constant when the DVFS level is fixed), was identified using the model specification, $\bar{R}^2$, and p-values. Experiments where the individual cores were powered down (Chapter 5) proved the presence of this component and showed that Chapter 4 accurately modelled its magnitude and dependency with DVFS level. Another example can be found in Chapter 4 where the non-ideal voltage regulation was identified from the residual plots and individual estimated power components. Chapter 5 shows yet anther example where the effect of temperature on the residuals allowed the thermal effects to be modelled. With non-linear approaches, where the model is not specified using knowledge of the power consumption (see Section 2.1), the effect of individual components on the fit is more difficult to interpret, and the p-values cannot be calculated, these important discoveries about the system being modelled would most likely be overlooked.

In summary, while constructing a well-specified linear model can require more knowledge of the system being modelled, it can also uncover useful and interesting information and relationships about the system being modelled, that are important when the model is exposed to other scenarios or applied to other systems.

It is important to continuously test the suitability and applicability of new and different modelling approaches, providing that both the proposed approach and existing baseline approach are understood properly, implemented correctly and tested thoroughly.

Chapter 4 proposed the use of the (linear) Weighted Least Squares (WLS) method as an option for dealing with the inherent heteroscedasticity. Instead of minimising the residual sum of squares (RSS), WLS minimises the weighted sum of squares. The weights can be set to be equal to the reciprocal of the error variance:

$$W_i = \frac{1}{{\sigma_i}^2} \tag{3.38}$$

The model will then give a higher weight to observations with a lower error variance.

3.2 Extracting PMCs on Mobile Systems

Performance Monitoring Counters (PMCs), or simply *hardware performance counters*, are registers that can be set to count specific CPU architectural and micro-architectural events, such as an instruction speculatively executed, or an L1 instruction cache miss. There are typically a small number of counters (i.e. less than 10) that can be assigned to count specific events, of which there are many possibilities (i.e. 40-200). They have long been available on several architectures, including x86/x64 and ARMv7/ARMv8. PMCs on Intel x86/x64 architectures are exposed through the model-specific register (MSR) interface, which are special registers for debugging that must be accessed from a privileged software level. More information on using PMCs with Intel architectures is available in the relevant Intel Software Developers Manual (e.g. [131]). The *perf* software tool is part of Linux that allows userspace monitoring of PMCs and it supports the vast majority of modern Intel platforms, making the task of accessing PMCs on x86/x64 architectures straightforward.

On ARMv7 and ARMv8 architectures the Performance Monitoring Unit (PMU) of the CPU handles the collection of the processor and memory counters and is accessed through system registers. Most Arm Cortex-A series CPUs contain either four or six counters (Table 3.1). On recent Arm Cortex-A series processors, there is also an Activity Monitoring Unit (AMU), which has similarities with the PMU, but is specifically targeted for system power management, instead of application debugging. Note that AMU was introduced after the work of this thesis was carried out and so only the PMU is considered.

There is a large set of possible events to monitor with the counters including events related to the processor core and the memory subsystem. Examples include:

- Level 1 instruction cache refill
- Level 1 data cache access
- Instruction architecturally executed
- Predictable branch speculatively executed
- Instruction speculatively executed
- Level 1 data TLB refill, write
- Level 2 data cache refill, write
- Instruction speculatively executed, Advanced SIMD Extension

Table 3.1: A selection of Arm Cortex-A series processors and the corresponding number of PMU counters

CPU	Architecture	No. Counters[1]	Manual
Cortex-A8	ARMv7	4	[10]
Cortex-A9	ARMv7	6	[12]
Cortex-A7	ARMv7	4	[14]
Cortex-A15	ARMv7	6	[13]
Cortex-A53	ARMv8	6	[16]
Cortex-A72	ARMv8	6	[17]
Cortex-A75	ARMv8	6[2]	[18]

The *perf* tool is not as well supported on Arm platforms, causing a lack of research utilising such features on mobile CPUs. Furthermore, in many cases the PMU was not fully or correctly implemented by the vendor, making PMCs physically unavailable on some platforms. For this reason, a custom software tool was developed to read the PMC events, without relying on *perf* support or working PMU interrupts. The latest version of this software tool, which started development in 2013, was made available as part of the GemStone suite of software tools [306] (GemStone Profiler Logger).

The PMCs cannot be controlled and read from userspace without the Performance Monitor User Enable Register (PMUSERENR) first being configured (which must be done from kernel space). Instead of modifying the kernel source and recompiling the whole kernel, a Loadable Kernel Module was created to configure the PMUSERENR registers to allow userspace access to the PMC registers (`enable-pmcs/powmon-enable-pmcs.c`). The LKM source code must be compiled against the kernel source (although it is safe to force-load if the kernel source is not available) using the included Makefile (`enable-pmcs/Makefile`). The LKM can then be inserted using the `insmod` or `modprobe` commands in Linux. The software tools support enabling PMCs on all system CPUs (regardless of the number of PMCs each one has, the total number of cores, or the combination of *big* and *LITTLE* cores). The tool was extended to also support newer devices implementing the ARMv8 architecture.

A software tool written in C and using inline assembly was used to setup and record the PMCs efficiently. The source code for setting and recoding the PMCs is in the `src/pmc-helper-64.c` file. The GemStone-Profiler Logger software tool allows PMC events to be set differently for different core types (e.g. Arm Cortex-A7 and Arm Cortex-A15) included in the same device and supports both ARMv7 and ARMv8 devices. It also works when cores are switched offline at run-time. In addition to recording the PMCs, it can also collect other run-time statistics, such as the current CPU frequency, and certain platform-specific sensors. For example, if using the Hardkernel ODROID-XU3 board, which includes power sensors and thermal sensors, the tool can be compiled with a special flag to enable the recording of these.

3.3 Experimental Platform

The mobile development board used in this Chapter was the BeagleBoard-xM [23] open-source single-board computer. It has a Texas Instruments DM3730 System-on-Chip (SoC) [287] 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 also contains an Imagination Technologies SGX GPU and a Texas Instruments TMS320C64x+ DSP. For this work, only the power consumption of the Cortex-A8 core was considered. The Cortex-A8 has four PMCs in addition to the cycle counter.

The Linux kernel uses the *CPUFreq* subsystem to control the CPU frequency and voltage (i.e. the DVFS level). Part of this subsystem includes scaling *governors* which implement algorithms for estimating the CPU capacity and guiding the DVFS level selection [288]. The kernel source was modified to enable the *userspace* governor, allowing the DVFS level to be manually controlled, and to enable the 1 GHz frequency (disabled by default).

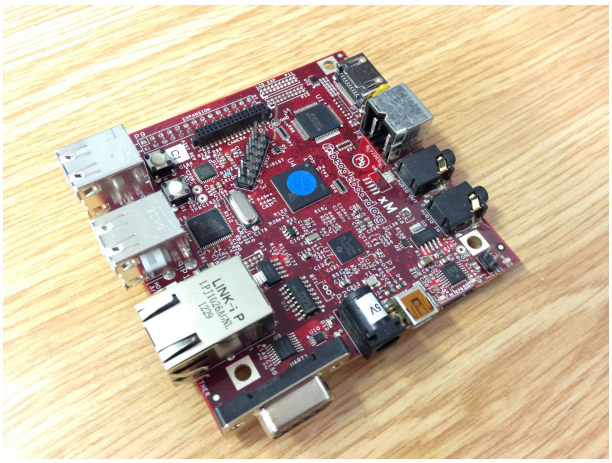
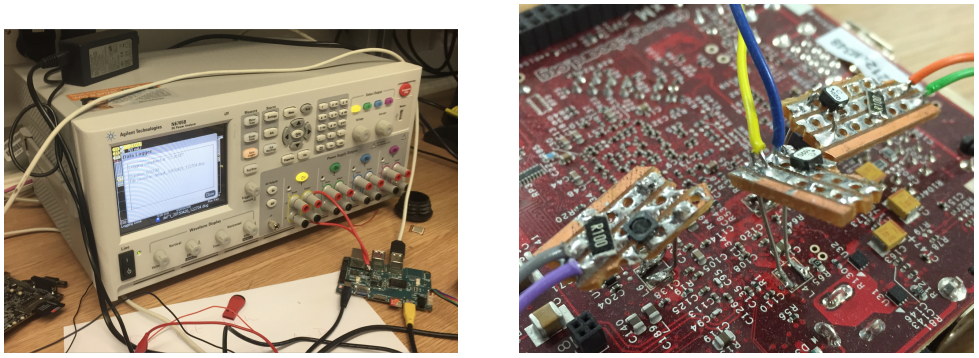


Figure 3.1: BeagleBoard-xM development board



(a) Power Analyser (b) Modified BeagleBoard

Figure 3.2: Power measurement setup

3.4 Recording Run-Time Power Measurements

Instead of measuring the power consumption of the whole board, the CPU power consumption was isolated by lifting an inductor from the board and re-routing the signal through the same inductor and an Agilent N6705 Power Analyser (Figure 3.2, the pad with blue and yellow wires, without a sense resistor). The voltage supplied to the Cortex-A8 was also measured to confirm the DVFS level and to calculate the power consumed. A sample period of 10 ms was used.

3.5 Experimental Setup

An experimental setup was devised for running experiments and recording both PMCs and power consumption (Figure 3.3). A software tool was developed to automate the running of workloads on the development board (BeagleBoard), repeating at multiple DVFS levels and to measure many PMC events (the GemStone suite of software tools contains a later version of this software tool, called GemStone Profiler-Automate). The software recorded the experimental data to file, as well as sending live experiment information to an NXP LPC1768 microcontroller (containing an Arm Cortex-M3 core) via UART over USB (Figure 3.4). Software on the microcontroller provided live

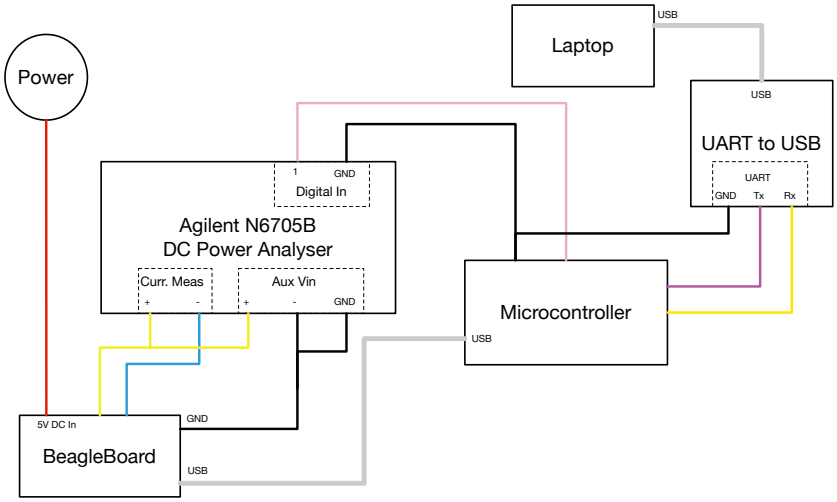


Figure 3.3: Experimental setup used for characterising and validating the PMC-based power model

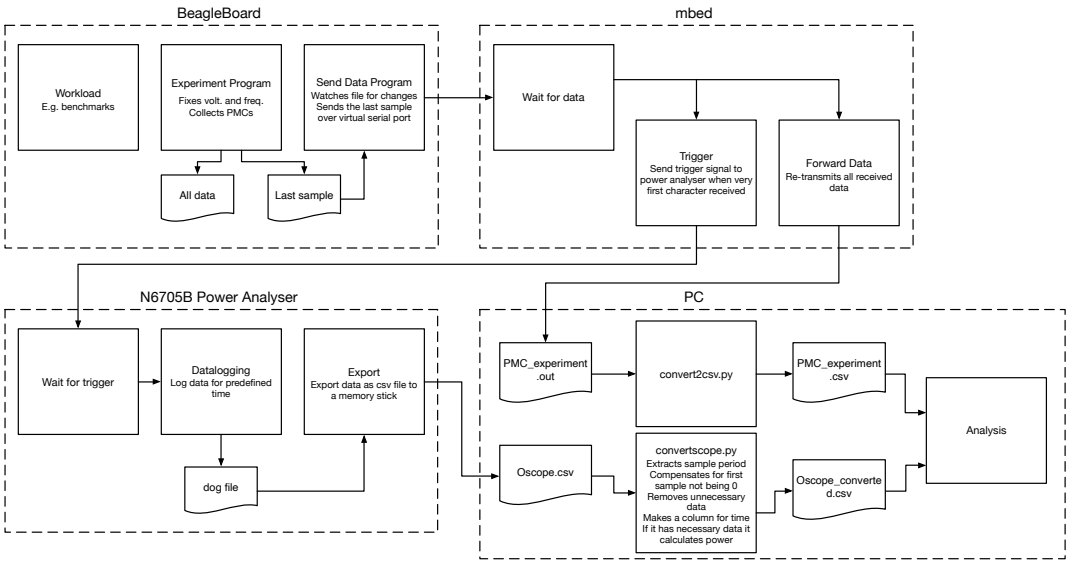


Figure 3.4: Experiment workflow

experiment updates to a laptop for online experiment monitoring (UART over USB) and also triggered the power analyser to start and stop data recording at the correct times.

3.6 Methodology Overview

As the power measurement and PMC collection were independent systems, the traces for each run of the experiment had to first be synchronised (Section 3.7).

The first stage of developing the model involved selecting the PMC events to be used as model input features. As only four events (in addition to the cycle counter) could be simultaneously counted, the experiment had to be run multiple times to capture data from the full set (and to ensure experimental consistency). The multiple runs of the experiment then had to be combined, and an

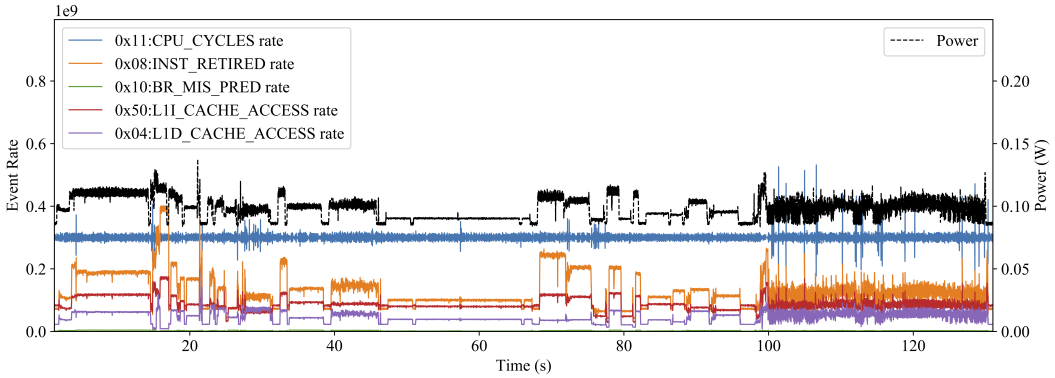


Figure 3.5: Raw PMC events and corresponding power consumption (before synchronisation). $f_{clk} = 300$ MHz.

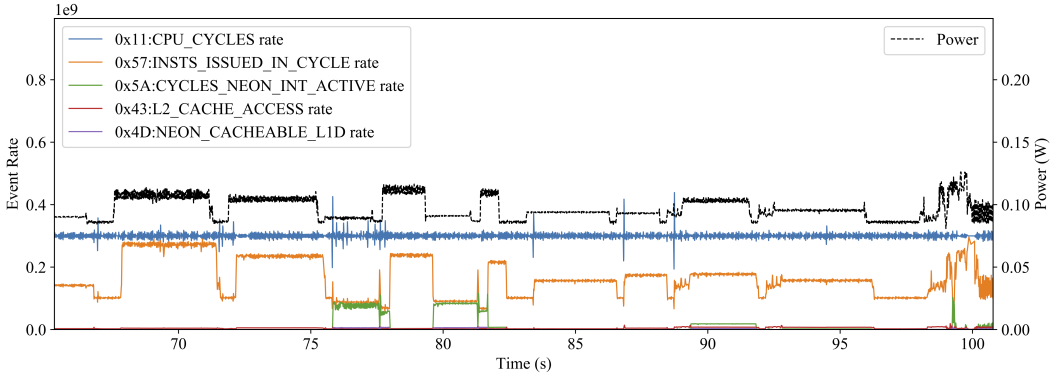


Figure 3.6: NEON SIMD and integer events (before synchronisation). $f_{clk} = 300$ MHz

offline, unsupervised phase detection technique was developed and employed to do this (Section 3.7).

The PMC events, their relationship with the power consumption, and their relationship with each other was analysed in order to select the model features (Section 3.8).

Once the features had been selected, the experiment was re-run with the chosen PMC events. The power traces and PMC event traces still had to be synchronised, but multiple runs did not have to be combined. However, phase detection was again utilised in order to identify a large number of workload phases to achieve a diverse set of training and testing observations. The model was specified and the coefficients estimated using OLS regression (see Section 3.1). Residual plots, learning curves, and cross-fold validation were employed to evaluate the model. The methodology improves on those proposed in existing works and highlights multiple potential issues that are explored further in Chapter 4.

3.7 Post-Processing and Phase Detection

The experiment was run multiple times to capture all PMC events of interest, as only four events (in addition to the cycle counter) could be counted simultaneously per experiment (Figure 3.5). While

the software and microcontroller triggered the power analyser automatically, the power (and voltage) data and the PMC data still needed to be synchronised due to small delays in communication (Figure 3.6). The data from the power analyser had a stable sample period of 10 ms while the data from the PMCs had a slightly larger period of approximately 12 ms. Both datasets where interpolated linearly and re-sampled with a period of 5 ms. The cross-correlation (3.39) between the measured power consumption and the rate of a PMC event was used to measure the lag (in number of samples) between the two datasets (Figure 3.7). This worked effectively for events that had a high correlation with power (e.g. 0x08:INST_RETIRED) but was ineffective for experiments that did not contain a PMC Event with a high correlation with power (e.g. Figure 3.8). To overcome this, it was found that using the absolute value of the derivative of the event rate (with the absolute value of the derivative of power) for cross-correlation was effective at identifying the lag (Figure 3.9).

$$(f * g)(\tau) = \int_{-\infty}^{+\infty} f^*(t)g(t+\tau)dt \tag{3.39}$$

While the previous method synchronises the power and PMC traces of the same experimental run, a technique for combining the data from the multiple experimental runs (capturing multiple PMC events and measuring experimental consistency) was required. Change Point Detection (CPD) techniques (also known as process segmentation techniques) were used to extract workload phases from the experimental data. CPD algorithms can be classified as either offline, where the entire dataset can be observed at once; or online, where the algorithm must process each data point as it arrives and detect a phase change as soon as possible after it has occurred (a review of online change detection is available at [22]). Both supervised approaches, where training data that has been labelled with change points is used, or unsupervised approaches, where phases are detected in unlabelled data, are available [5, 292, 232].

Previous works have used various approaches for detecting workload phases at run-time, such as instruction working sets, conditional branch counts, and basic block vectors (BBVs) [80, 165]. Isci and Matonosi [136] focus on detecting power oriented phases at run-time, and compare between the use of BBVs and PMC-based techniques. They find that PMCs provide a better representation of power behaviour.

This chapter utilises offline, unsupervised CPD to identify abrupt changes in the power mean in order to automatically identify workload phases for extracting a larger number of more diverse training samples and to enable multiple runs of the experiment to be combined.

A method is devised to identify:

1. the number of execution phases;

2. the location of each execution phase;

3. the value of the signals (e.g. the mean power and PMC event rates) for each execution phase.

One approach is to use Markov chain Monte Carlo (MCMC) if the number of segments are not fixed [111, 167]. However, in this case we assume the number of segments (workload phases), K, is fixed (through visual inspection of one of the traces). Maximum Likelihood (ML) estimation was used to identify abrupt changes in the mean, μ, using the following simple model [231, 168, 20]:

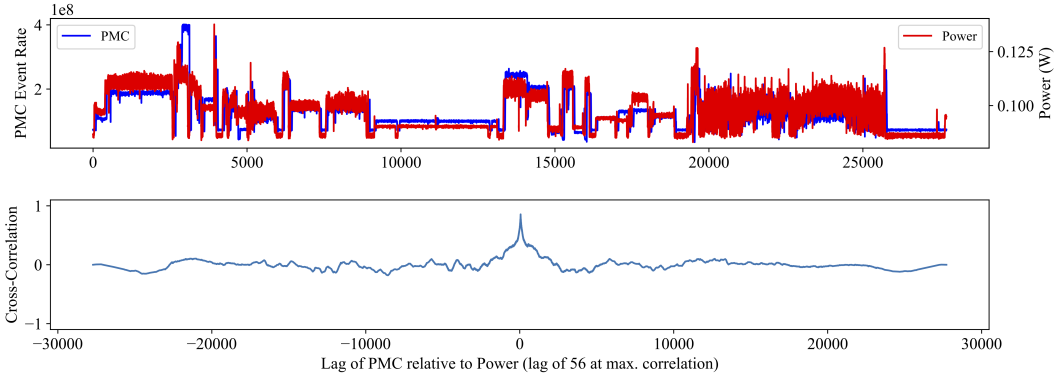


Figure 3.7: Cross-correlation of Event 0x08 (INST_RETIRED) and the power. The lag is accurately calculated.

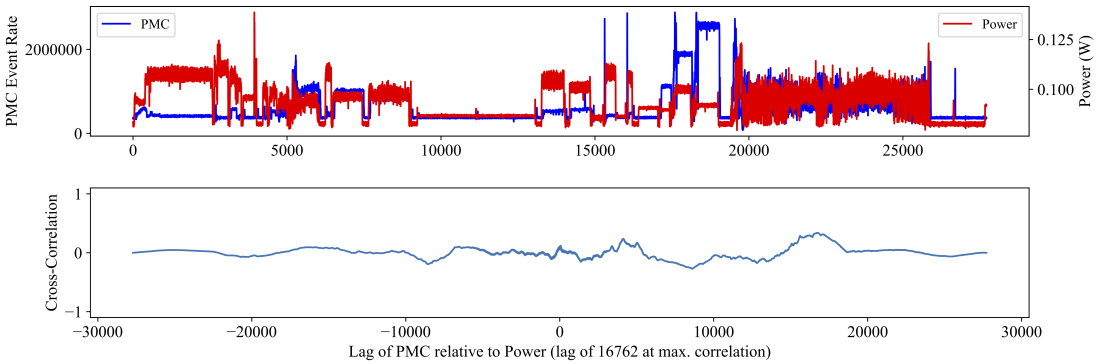


Figure 3.8: Cross-correlation of Event 0x05 (L1D_TLB_REFILL) and the power. The lag is not correctly derived.

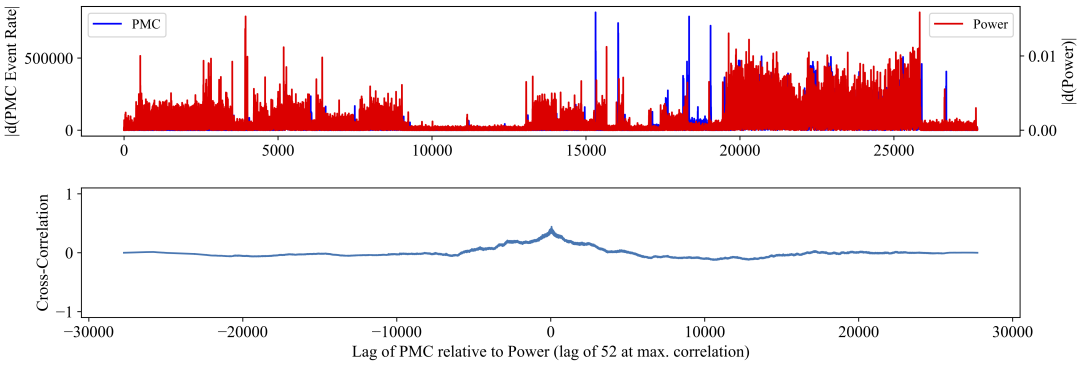


Figure 3.9: Cross-correlation of the absolute of the derivative of the Event 0x05 (L1D_TLB_REFILL) rate and the absolute of the derivative of the power measurements. The lag is accurately calculated.

$$y_i = f(t_i) + e_i \tag{3.40}$$

$$0 \leq i \leq N \tag{3.41}$$

where $f(t)$ represents an idealised version of the signal (i.e. power trace or PMC event rate trace), e_i are the residuals (see Section 3.1), and N is the number of samples.

The execution phases are denoted, $k_1, k_2, \ldots k_K$, and the boundaries of these segments are the discontinuity instants, $\tau_1, \tau_2 \ldots \tau_{K-1}$.

For a single execution phase, k, where $\tau_{k-1} \leq t \leq \tau_k$:

$$f(t) = \mu_k \tag{3.42}$$

Using this model, the three points above can be re-written as:

1. the number of execution phases, K;
2. the boundary of each execution phase, τ_{k-1} and τ_k;
3. the mean value of the signals (e.g. power and PMC event rates) for each execution phase, μ_k

The Maximum Likelihood (ML) Estimation (sometimes abbreviated to MLE) method was then used to estimate the parameters of the model. It was assumed that the residuals, e_i, have a Gaussian, or normal, distribution and a mean of zero:

$$e_i \sim \mathcal{N}(0, \sigma^2) \tag{3.43}$$

and y_i was also assumed to have a Gaussian distribution (if $\tau_{k-1} \leq i \leq \tau_k$):

$$y_i \sim \mathcal{N}(\mu_k, \sigma^2) \tag{3.44}$$

Therefore, this method becomes equivalent to the least squares minimisation (Section 3.1).

The Gaussian distribution is given by:

$$f(y) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(y-\mu)^2\right) \tag{3.45}$$

where μ is the mean:

$$\mu = \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{3.46}$$

and σ is the standard deviation:

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{3.47}$$

In other words, the probability density of observing the single observation, y, generated from a

Gaussian distribution is given by:

$$P(y;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(y-\mu)^2\right) \tag{3.48}$$

For this model, it was required to estimate the vector of parameters, $\theta = (\mu_1, \ldots, \mu_K, \sigma^2, \tau_1, \ldots, \tau_{K-1})$. The likelihood function is therefore given by:

$$\mathcal{L}(\theta; y_1 \ldots y_n) = p(y_1 \ldots y_n; \theta) \tag{3.49}$$

Considering the K execution phases:

$$\mathcal{L}(\theta; y_1 \ldots y_n) = \prod_{k=1}^{K} p(y_{\tau_{k-1}+1}, \ldots, y_{\tau_k}; \mu_k, \sigma^2) \tag{3.50}$$

Using (3.48):

$$\mathcal{L}(\theta; y_1 \ldots y_n) = \prod_{k=1}^{K} \frac{1}{2\pi\sigma^2}^{\frac{\tau_k - \tau_{k-1}}{2}} \exp\left(-\frac{1}{2\sigma^2} \sum_{i=\tau_{k-1}+1}^{\tau_k} (y_i - \mu_k)^2\right) \tag{3.51}$$

Which can be re-written as:

$$\mathcal{L}(\theta; y_1 \ldots y_n) = \frac{1}{2\pi\sigma^2}^{\frac{n}{2}} \exp\left(-\frac{1}{2\sigma^2} \sum_{k=1}^{K} \sum_{i=\tau_{k-1}+1}^{\tau_k} (y_i - \mu_k)^2\right) \tag{3.52}$$

There are then two key steps in MLE of θ:

1. estimate μ_k and τ_k using the following cost function:

$$J(\mu_1 \ldots \mu_K, \tau_1 \ldots \tau_{K-1}) = \sum_{k=1}^{K} \sum_{i=\tau_{k-1}+1}^{\tau_k} (y_i - \mu_k)^2 \tag{3.53}$$

2. estimate the variance of the estimated residuals:

$$\hat{\sigma}^2 = \frac{1}{n} \sum_{k=1}^{K} \sum_{i=\tau_{k-1}+1}^{\tau_k} (y_i - \hat{\mu}^k)^2 \tag{3.54}$$

For this purpose, a constant variance was assumed for simplicity and so only (3.53) needed to be calculated. Using only the mean of the residuals to estimate the change point removes the assumption of the residuals having a Gaussian distribution.

To provide a practical intuition of the method, a simplified example is provided for the case where there is only one clear change in the mean and $K = 1$ (Figure 3.10). The cost function is applied for two example cases, at $\tau = 160$ and $\tau = 600$ (Figure 3.11, upper plots), and resulting residuals, e are observed (lower plots).

This process is repeated for all values of τ and the corresponding residual sum of squares (RSS, see

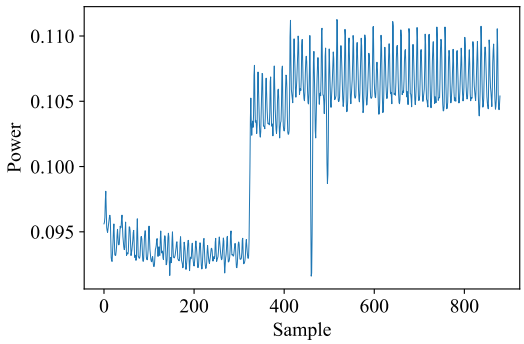
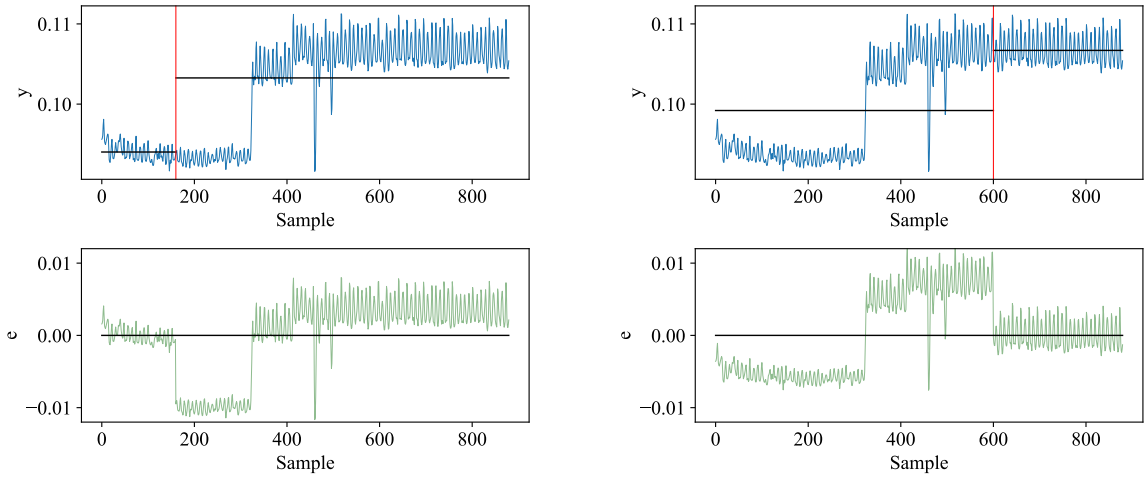


Figure 3.10: Example segment of the power trace with a clear, abrupt phase change



(a) tau=160 (b) tau=600

Figure 3.11: Upper plot shows the example power trace. τ_k (selected manually for this example) and the horizontal black lines show $\mu_k - 1$ and μ_k. The lower plot shows the resulting residuals for the chosen value of τ_k.

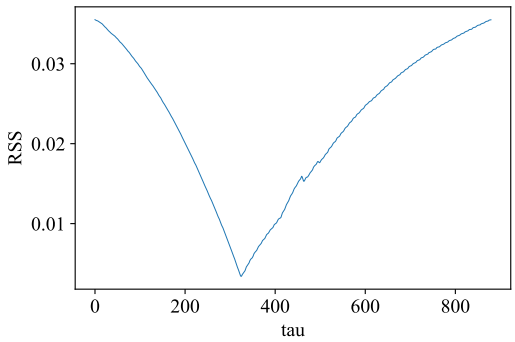


Figure 3.12: RSS plot across all values of τ (tau)

Section 3.1) is calculated (Figure 3.12). The cost function finds the value of τ (tau) that minimises the RSS, which is 325 in this example (Figure 3.13).

However, the task of finding K segments through an exhaustive search has a time complexity of $\mathcal{O}(N^k)$ and so a dynamic programming approach was employed, reducing the computational

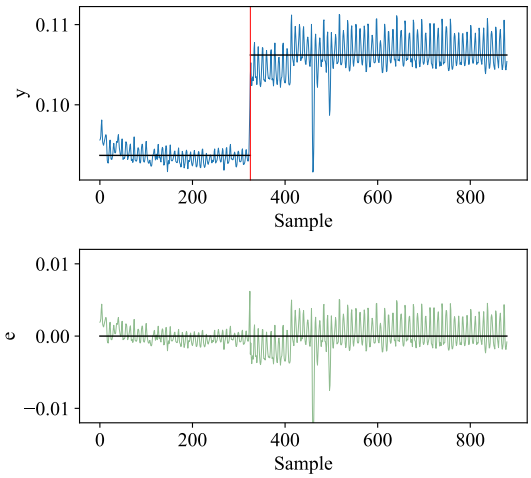


Figure 3.13: The power trace (upper plot) and residuals (lower plot) at the automatically identified change point ($\tau = 325$)

complexity to $\mathcal{O}(N^2)$ [233, 231]. The algorithm was implemented in Python, using Numpy, and based on the R implementation by Marc Lavielle [166]. The algorithm was broken into two stages: Stage A, which computes a matrix of pointers, is independent of the chosen K and is the most compute-intensive stage; and Stage B, which takes the pointer matrix as an input and outputs a matrix of τ values for various numbers of K. The τ vector for a specific choice of K can simply be extracted from this matrix.

The workloads were all run sequentially in an individual experiment, meaning that the resulting traces contained all workloads and were over 120 seconds long for the case of the clock frequency, f_{clk}, being 300 MHz. However, even with the dynamic programming implementation, the algorithm does not complete in a reasonable amount of time ($N \gg 240000$ and $K \gg 30$). Furthermore, with a long trace of many workloads, it is hard to manually identify an appropriate value for K. The workloads were therefore split into eight groups, the corresponding traces of which were called *sections*, and a 1 second idle interval was placed after each workload group, enabling the sections to be automatically identified in the post-processing. This reduced the value of N (and K) leading to a significant reduction in the execution time of the algorithm. A 0.25 second idle interval was also inserted between workloads for a clear distinction between workloads and workload phases on the traces. The value of K for each section was derived using visual inspection. To further assist the selection of an appropriate K, the software program was extended to allow saving and restoring of the output of the compute-intensive Stage A (which is independent of K). The program could be restored multiple times from this point to execute Stage B, where K was chosen.

Only larger workload phases were selected for the PMC event selection dataset. Once the value of K was found for each workload section, the algorithm was able to automatically split sections, workloads and workload phases (Figure 3.14) and data from multiple runs of the experiment (repeated to test consistency and to monitor multiple PMC event sets) could be combined.

The purpose of this method was to extract phases (which could also be seen with manual inspection) automatically and repeatedly across the different experiments. The method correctly, and consistently, identified the abrupt changes in the workload power mean to extract the phases, as confirmed with visual inspection and comparison of the means between repeated experiment iterations.

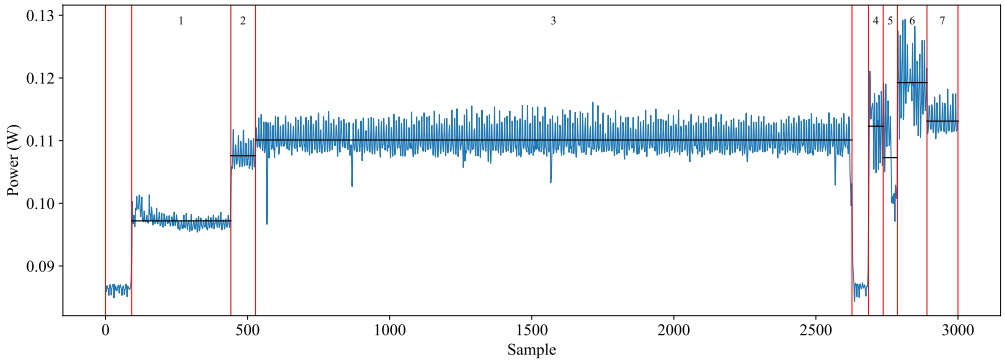


Figure 3.14: Automatic workload and workload phase detection

3.8 Feature Selection

Once the power and PMC traces had been synchronised, the phases extracted and the data from the multiple experiments combined, the PMC events to be used in the model (features) had to be selected. The Pearson product-moment correlation coefficient (PPMCC) was calculated (3.55) to give a measure of the correlation between each PMC event (rate) and the power consumption. The resulting measure of linear correlation between x and y, r_{xy}, has a value between -1 and $+1$, where a value close to $+1$ signals a strong position correlation, a value close to -1 signals a strong negative correlation, and a value close to 0 signals that there is a weak correlation or no correlation at all.

$$r_{xy} = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{\left(n\sum x_i^2 - \left(\sum x_i\right)^2\right)\left(n\sum y_i^2 - \left(\sum y_i\right)^2\right)}} \tag{3.55}$$

Event rates `0x55` (operations issued), `0x57` (instructions issued per cycle), `0x08` (instructions retired/committed) and `0x50` (L1 instruction cache accesses) had the strongest positive correlation with power (Figure 3.15). This matches intuition as they are directly related to the utilisation of the CPU. For example; the level 1 instruction (L1I) cache is accessed to fetch the next instruction; and instructions retired indicates that an instruction has been architecturally committed. The Arm Cortex-A8 is dual-issue superscalar, meaning it can execute multiple instructions per cycle.

Event rates related to the data cache and data reads and writes had a moderate positive correlation with power consumption, likely caused by the energy required to move data in the memory hierarchy. However, there is no significant relationship between the power consumption and the level 2 (L2) cache accesses (L2_CACHE_ACCESS:`0x43`) and memory accesses (L1D_CACHE_REFILL:`0x03`). These events occur less frequency and the time penalty incurred from them means that the CPU is prevented from executing. While in this experiment the NO_INST_AVAIL:`0x56` event rate has a very low correlation with power, this would likely not be the case where the CPU goes into an inactive state when no instructions are executed (it is expected that it would be negative).

It should be remembered that it is the correlation being observed, not the causation. For example, while the rate of branch mispredictions (BR_MIS_PRED:`0x10`) has a moderate positive correlation with power, it is not known whether it is the branch mispredictions themselves causing an increase

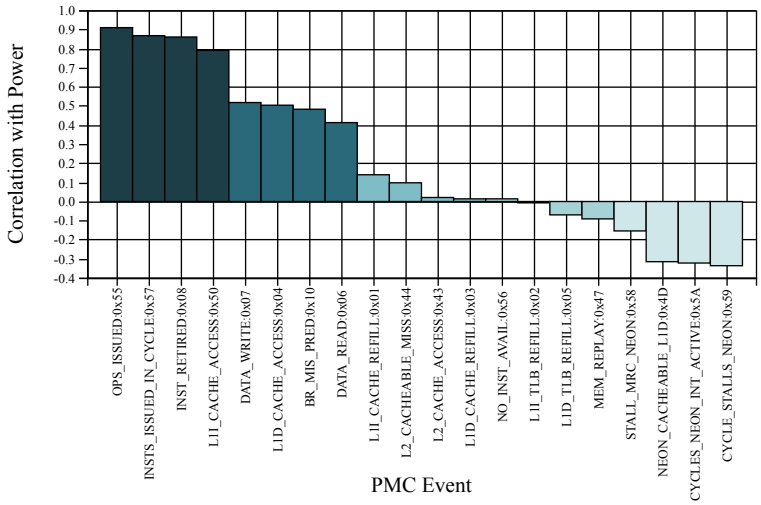


Figure 3.15: Correlation of each PMC Event with the measured power consumption

in power or whether workloads with a higher number of branch mispredictions also tend to be more power hungry (i.e. also having a large number of operations issued).

The 20 PMC events were split into five groups based on their correlation with power. Min-max normalisation (3.56) was applied to each PMC event across all of the workloads (Figure 3.16) to understand the different PMC behaviour of each workload. Zero represents the minimum value, while one represents the maximum value.

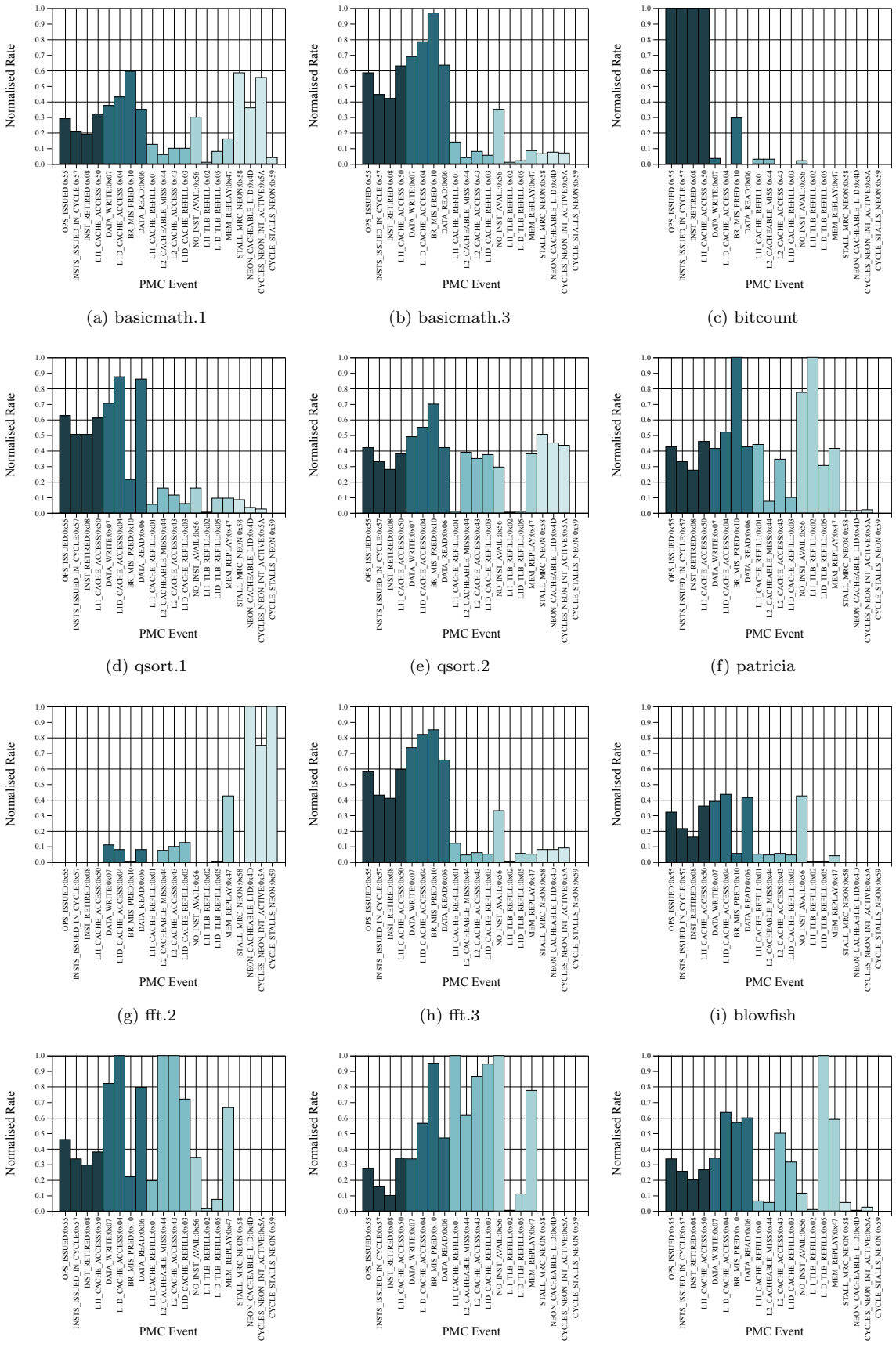
The first observation from viewing the PMC event min-max normalisation is that there is a large deviation between the workloads, including between detected phases within the same workload, such as *basicmath.1* and *basicmath.3*, *qsort.1* and *qsort.2*, *fft.2* and *fft.3*, and *tiff2rgba.1* and *tiff2rgba.2*.

$$x' = \frac{x - min(x)}{max(x) - min(x)} \tag{3.56}$$

From observing the normalised PMC events for several workloads, it is clear that PMC events in the same correlation group tend to behave in a similar manner between workloads. For example, all PMC events of the first correlation group (darkest shading) are the maximum for the *bitcount* workload (Figure 3.16c) and the minimum values for *fft.2* (Figure 3.16g). Therefore, the technique of using correlation, as employed by existing works (including [24, 36, 34, 280] and the specific mobile works, [238, 244, 214]) would not be effective as the events with the highest correlation with power also have a high correlation with each other.

A combination of computer micro-architecture knowledge (Section 2.4), the correlation of PMC events with power consumption (Figure 3.15), and the normalised PMC event plots (Figure 3.16) were used to select the following PMC Events:

1. `0x55` (85): Number of operations issued;
2. `0x50` (80): L1 instruction cache accesses;
3. `0x04` (4): L1 data cache accesses;
4. `0x4D` (77): Neon cacheable L1 data.



(a) basicmath.1 (b) basicmath.3 (c) bitcount

(d) qsort.1 (e) qsort.2 (f) patricia

(g) fft.2 (h) fft.3 (i) blowfish

(j) tiff2rgba.1 (k) tiff2rgba.2 (l) int

Figure 3.16: Min-max normalised PMC event rate for selected workloads

Table 3.2: Mean, standard deviation and coefficient of variance of all observations for each DVFS level

Frequency (MHz)	Mean	Std.	CV
300	0.0969	0.00793	8.2%
600	0.269	0.0264	9.8%
800	0.436	0.0432	9.9%
1000	0.535	0.0549	10.3%

The first two chosen events are in the group of four PMC events with the highest correlation with power. However, the `0x55` has the highest correlation with power (> 0.9), while `0x50` has the lowest in the group (< 0.8). Furthermore, it can be seen that there are both cases where the normalised value of `0x55` is greater than `0x50` (*qsort.1*, *qsort.2*, *tiff2rgba.1*, *int*) and cases where the opposite is true (*basicmath.3*, *patricia*, *blowfish*, *tigg2rgba.2*). In other words, they behave differently to each other, therefore providing unique information without excessive multicollinearity. The third event was chosen from the second group of PMC events, which is made up of four events with a correlation between 0.42 and 0.5. The third and fourth groups of PMC events had a very small absolute correlation with power (< 0.2) and so were not considered as an input feature. The final event was chosen from the last group of events which all had a negative correlation with power. Event `0x4D` was chosen over event `0x59` because event `0x59` was at its minimum for many of the workloads.

3.9 Model Specification and Validation

Once the four PMC events had been chosen, the experiment was re-run with the selected events at the four DVFS levels (300 MHz, 600 MHz, 800 MHz and 1000 MHz).

The ML phase detection technique was applied to extract 74 workload phases at each DVFS level. All phases that were classified as idle were removed, with the exception of one, to leave 48 workload phases for use as training and testing observations.

Increasing the frequency causes an increase in power consumption (Figure 3.17). The increase with frequency is lower than observed later in Chapter 4 for the Cortex-A15 CPU due to the fact that the Cortex-A8 processor does not have the out-of-order execution and is only dual issue, meaning it is not as capable at absorbing memory latency, which becomes more important as the CPU clock frequency increases.

The Coefficient of Variation (CV) was also calculated to provide a sense of relative variability:

$$CV = \frac{\sigma}{\mu} \times 100 \tag{3.57}$$

As f_{clk} increases, the coefficient of variation also increases by a small amount (Table 3.2).

The cycle counter (`0x11`) was not useful for this model as power management techniques, such as idle states, were not implemented in this operating system, and so all clock cycles were active. In Chapter 4, a system with more advanced power management techniques is considered and the cycle counter was found to be important on such a system.

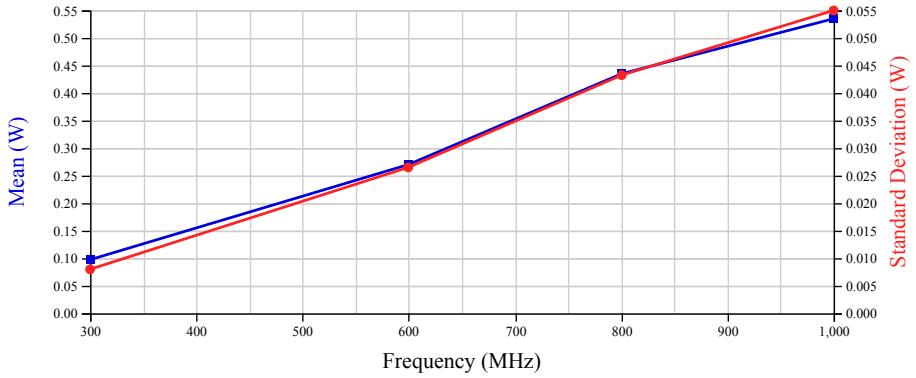


Figure 3.17: Power mean and standard deviation

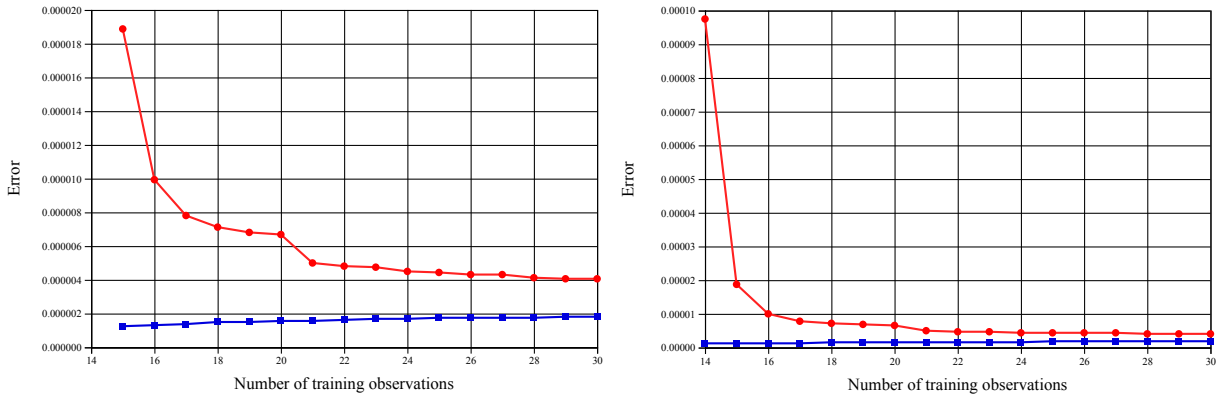
In this chapter, a separate model is developed for each individual DVFS level (3.58). The PMC events provide an indication of the dynamic power consumption while an intercept is used to estimate the static power consumption and any constant (at a particular DVFS level) dynamic power consumption (Chapter 4 employs a single equation for all DVFS levels and decomposes the power consumption into multiple components).

$$P_i = \beta_0 + \beta_1 E_{i1} + \beta_2 E_{i2} + \beta_3 E_{i3} + \beta_4 E_{i4} + \epsilon_i \tag{3.58}$$

An important step in validating the model is to test for high bias (under-fitting) and high variance (over-fitting); the trade-off between the bias and variance is an important consideration in many machine learning algorithms. A high bias is caused by the model not being correctly specified (e.g. the model is over-simplified), and therefore being unable to correctly capture the relationship between the PMC event rates and the power consumption. A high variance is caused by the model being over-fitted to the training data and it is therefore not able to generalise. The bias can be decreased by introducing more features or higher order regression polynomials while the variance can be decreased by reducing model complexity and using regularisation. A common cause of high variance is using too many features and too few training observations. The purpose of the PMC events is to indicate the activity of the CPU (α, Equation 2.2 on page 12) and polynomials were therefore not appropriate (they would likely cause over-fitting).

Eighteen of the 48 observations were randomly selected and used as the testing set, with the remaining 30 workloads being reserved for training. A model was trained 27 times, using between two and 30 training observations and the corresponding mean squared error (MSE) when tested on both the training set (round markers, Figure 3.18) and testing set was recorded (square markers). This was repeated 500 times, with the mean recorded. Note that Figure 3.18 is only showing 14 training workloads onwards. This figure is known as a *learning curve* in the field of machine learning. The small error gap between the training set and testing set error shows that there is low variance (the model is not over-fitting the training data), and the low error of the training set indicates that the model does not have high bias. It was also observed that 22 was a suitable minimum number of training observations to use. The MAPE of testing on the testing set when trained with 22 training observations was 2.17%.

For evaluating the model coefficients and residuals, the models for each DVFS level were trained with



(a) (b)

Figure 3.18: Training and testing set error (MSE) for using different numbers of training observations (number of testing observations is 18). Values shown are the mean of 500 iterations, with both the training and testing workloads being randomized on each iteration. The same plot with two different scales on the y axis are shown.

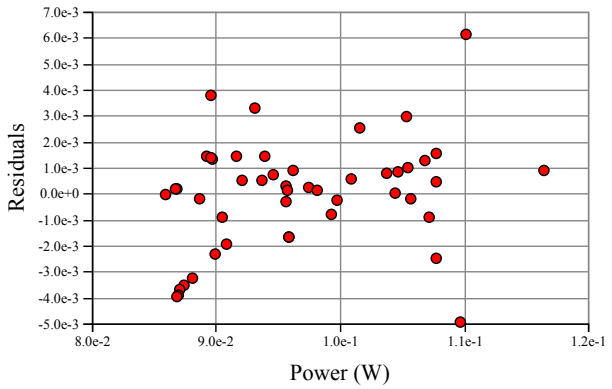
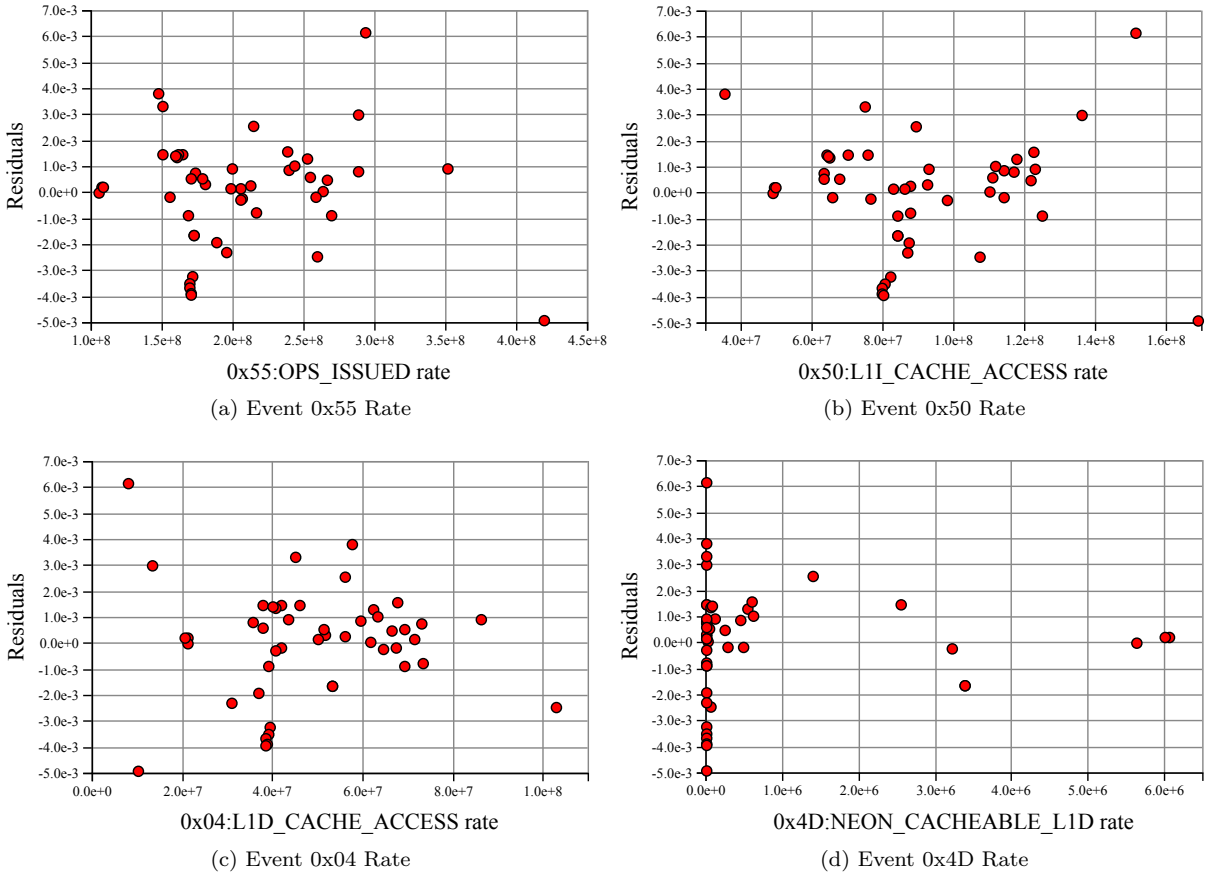


Figure 3.19: Model Power Residuals

all 48 observations. An important step of checking correct model specification is visual inspection of the residuals, e_i, (Figure 3.19). No patterns could be observed when the residuals are plotted against all of the features (Figure 3.20), showing that the model has effectively extracted the information from the features. Furthermore, it can be observed that only a handful of the workloads utilise NEON SIMD instruction (Figure 3.20d).

The model coefficients at each DVFS level are shown in Table 3.3. The unit of each feature is events per second. The MAPE and R^2 value of each model is shown in Figure 3.21a (in the case where the models are trained and validated using all of the observations). The higher f_{clk}, the better the *goodness-of-fit*. The $\bar{R}^2$ value is within 0.0008 of R^2, for example, at 1000 MHz, the R^2 value is 0.970 while $\bar{R}^2$ is 0.967. The MAPE at lower DVFS levels is higher because the same absolute power error results in a larger percentage error. The MSE increases with DVFS level (Figure 3.21b). At lower frequencies, the static power is a larger contributor to the overall power consumption. The static power consumption is highly temperature-dependent, and temperature is not taken into account in this model. In Chapter 5, further analysis of the effect of temperature on the power consumption is conducted and a temperature-compensated model is developed.



(a) Event 0x55 Rate

(b) Event 0x50 Rate

(c) Event 0x04 Rate

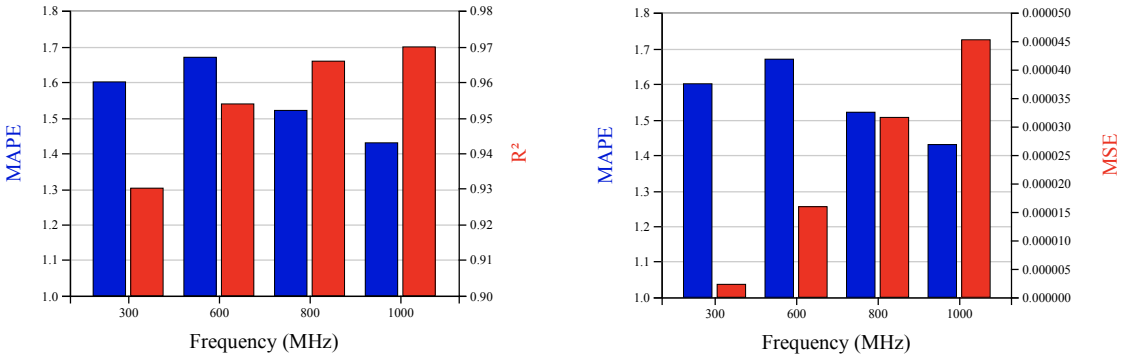
(d) Event 0x4D Rate

Figure 3.20: Residuals (shown for $f_{clk} = 300$ MHz)

In order to test the model's performance on unseen data and to test the model's ability to generalise, k-fold cross-validation was employed. The observations were split into six groups of eight observations ($k = 6$). Five of the groups were used for testing while the remaining single group was reserved for testing, and the process was repeated six times with a different testing group used each time. The highest MAPE of any fold and any DVFS level was under 2.9% (Figure 3.22), although there was a significant standard deviation between the fold MAPEs (0.71, 0.33, 0.28, 0.63 for 300 MHz, 600 MHz, 800 MHz, and 1000 MHz, respectively). The MAPE of the cross-validated models is not significantly different to the MAPE of the model trained and tested on all of the observation (Figure 3.23). The model does a fair job at estimating unseen workloads, though it must be remembered that the power variance is low (CV $\leq 10.3\%$) due to the significant idle power consumption (Figure 3.24b).

Table 3.3: Model Coefficients

Features	Coefficients			
	300 MHz	600 MHz	800 MHz	1000 MHz
Intercept	0.0627	0.1645	0.2640	0.3183
0x55:OPS_ISSUED rate	6.139e-11	7.695e-11	1.184e-10	1.123e-10
0x50:L1I_CACHE_ACCESS rate	1.451e-10	2.428e-10	2.358e-10	2.56e-10
0x04:L1D_CACHE_ACCESS rate	1.537e-10	2.542e-10	3.427e-10	3.578e-10
0x4D:NEON_CACHEABLE_L1D rate	1.124e-09	2.018e-09	2.613e-09	2.747e-09



(a) MAPE and R^2 (b) MAPE and MSE

Figure 3.21: Model performance for different DVFS levels (trained and tested with all observations)

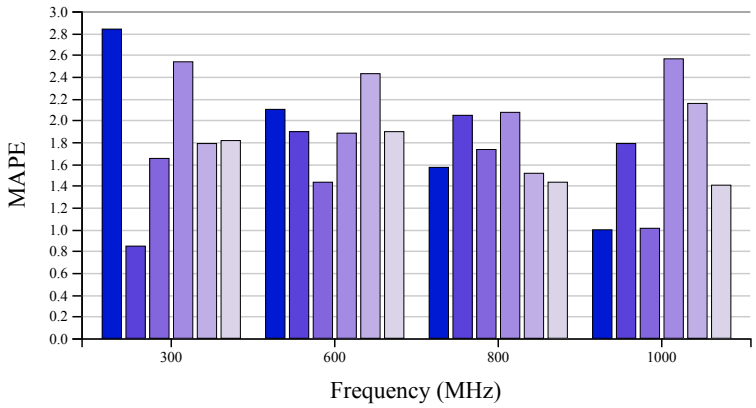


Figure 3.22: Testing set MAPE for each fold of 6-fold cross-validation

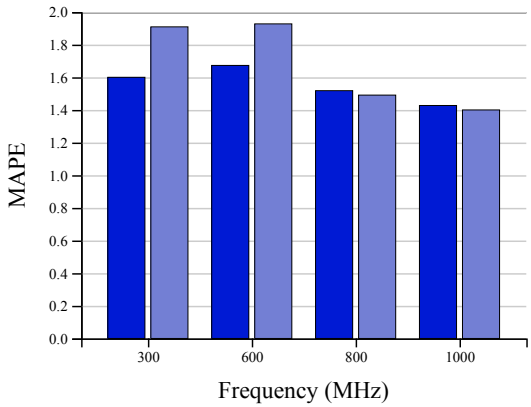
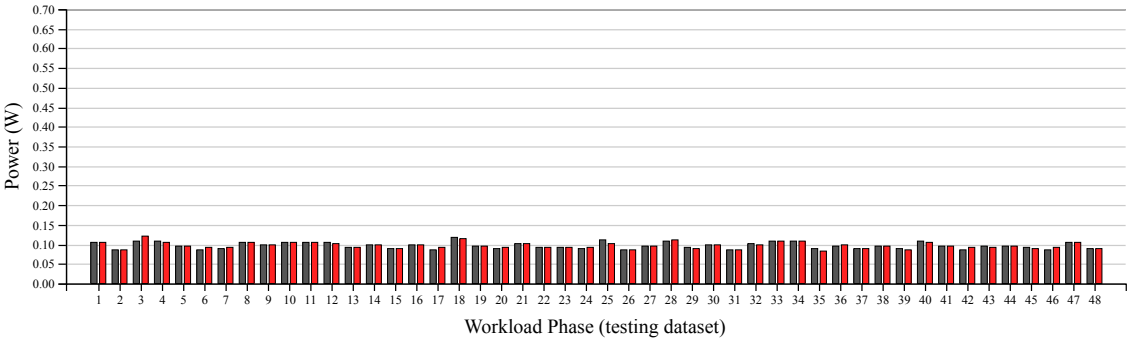
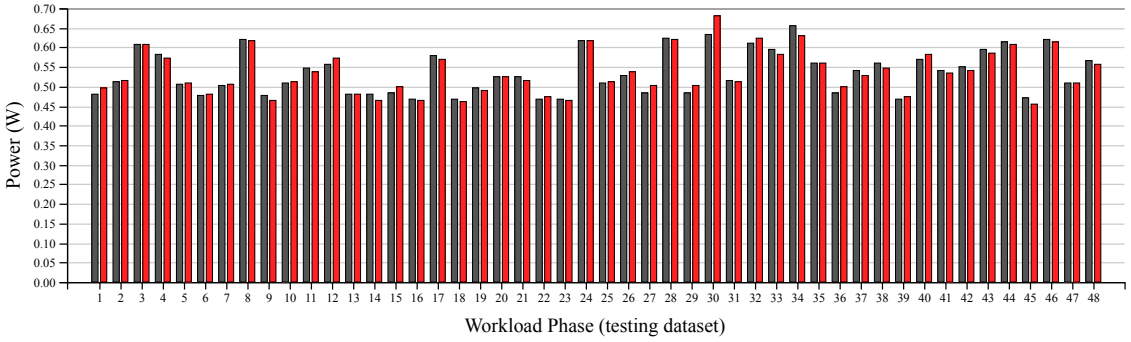


Figure 3.23: MAPE of model trained with all observations (left bars) vs. MAPE of models trained and tested with cross-validation (right bars)

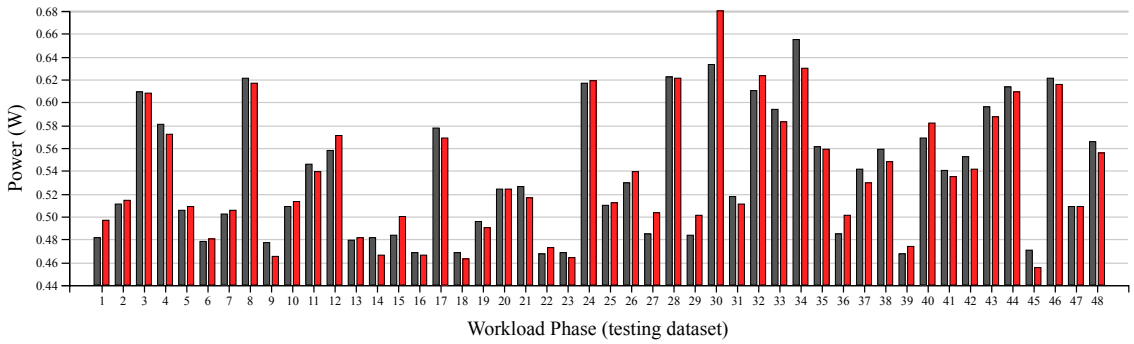
The measured and predicted power consumption for each workload is plotted (Figure 3.24). It can clearly be observed that there is a large difference in power consumption between 300 MHz and 1000 MHz (Figure 3.24a), and that the dynamic power variation is small compared to the large constant components (Figure 3.24b). However, it can also be observed that the model accurately estimates the dynamic variation across the workload phases (Figure 3.24c).



(a) f_{clk} = 300 MHz



(b) f_{clk} = 1000 MHz



(c) f_{clk} = 1000 MHz

Figure 3.24: Cross-fold validation across all testing observations (Note that the observations are not consistent between plots of different DVFS levels as the workload phase selections are randomised).

While the models have been found to be accurate across the selected observations (the power and PMC event rates of each phase were averaged across their time), this does not imply that the models work effectively on the real-time traces. The models were applied to collected data after the cross-correlation had been applied (to synchronise the power and PMC traces), but before individual phases had been extracted (Figure 3.25). The model achieves a MAPE of 1.39% across the first 35 seconds of the real-time traces (which includes all workloads with the exception of an extra video playback [*MPlayer*] workload) at 1 GHz. A MAPE of 2.14% was achieved when the model was tested on a video playback workload (Figure 3.25b). Closer inspection of the traces (Figures 3.25c, 3.25d, and 3.25e) show the responsiveness of the models (the importance of which is underlined in [26]).

The high-frequency spikes in both the PMC data and the power consumption (caused by high frequency CPU workload variation) make the task of estimating the power consumption on the

Table 3.4: Model coefficients for the different folds

Event	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
0x55 rate	1.14E-10	1.12E-10	1.46E-10	1.18E-10	1.07E-10	9.88E-11
0x50 rate	2.46E-10	2.61E-10	2.24E-10	2.44E-10	2.69E-10	2.68E-10
0x04 rate	3.63E-10	3.71E-10	2.95E-10	3.83E-10	3.55E-10	3.50E-10
0x4D rate	2.78E-09	2.75E-09	2.79E-09	2.93E-09	2.80E-09	2.53E-09
const	0.318	0.316	0.315	0.314	0.319	0.326

run-time power traces more difficult. Moreover, in most run-time management scenarios, the power consumption of high frequency events is not desired. The Simple Moving Average (SMA, also known as the rolling mean) filter was applied to remove some of the high-frequency events (it is a low-pass filter). For this example, a 20 sample (centred) window was used (Figure 3.26). The MAPEs for the equivalent traces shown in Figure 3.25 were all significantly lower, for example, the MAPE across the first 35 seconds was 1.15%.

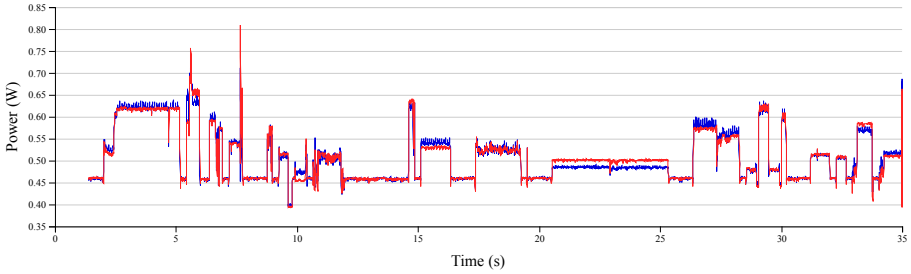
The MAPEs of the real-time traces were approximately the same as when tested on the full set of workload phase observations and were lower after using the rolling mean. This is because the workload phases are diverse and each one has an equal weight. In the real-time traces, the longer workload phases tend to be the ones that the model is able to predict more easily (despite the training set not being biased towards them), and there are therefore a higher proportion of data points that the model is better able to predict. This analysis shows that models trained with diverse workloads, and using the methodology outlined in this chapter, are accurate and responsive when applied to real-time traces.

3.10 Inspection of Coefficients

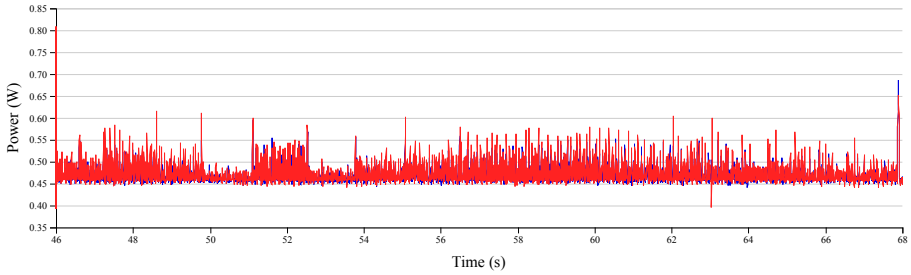
Existing works on PMC-based power modelling do not consider the stability of the model coefficients. It can be seen that the values of the coefficients change between different folds of the cross validation (Table 3.4). While it is expected that there will be variation in the resulting MAPEs, this is largely due to the fact the model is being tested on a different set of workloads. The model is still being trained with over 80% of the training observations and so the coefficients are expected to remain relatively consistent between the folds. However, there is variation between many of the coefficients, for example, the coefficient for event `0x55` varies between 9.88E-11 and 1.46E-10, while the coefficient for event `0x04` varies between 2.95E-10 and 3.83E-10. Furthermore, when an event is removed (Table 3.5), the coefficients for the existing events vary significantly (e.g. the mean coefficient value of `0x55` increases from 1.16E-10 to 2.16E-10). This is due to the presence of correlation between the input features and makes it difficult to obtain a model that makes accurate predictions across a wide range of testing observations. Correlation between input features, model stability, and their effect on model performance are analysed in Chapter 4.

3.11 Conclusion

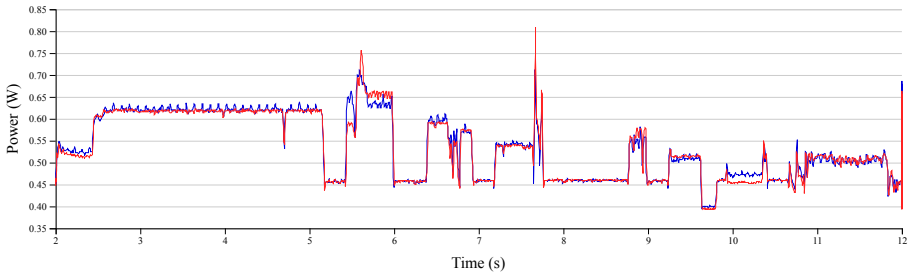
This chapter has presented a methodology for developing run-time empirical CPU power models, including techniques for extracting experimental data, feature selection, model specification, and



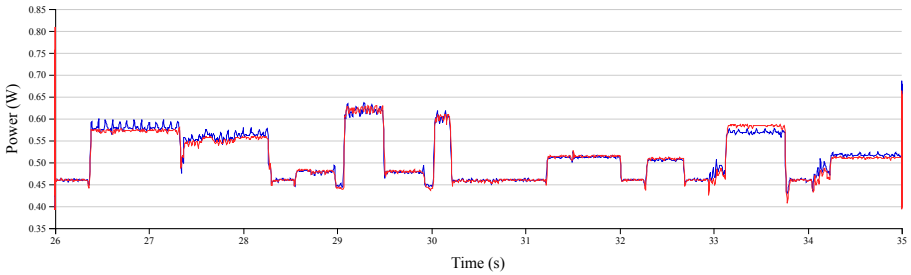
(a) MAPE=1.39%



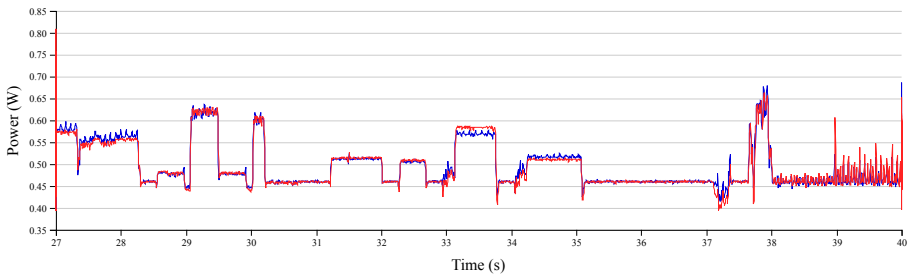
(b) MAPE=2.14%



(c) MAPE=1.37%

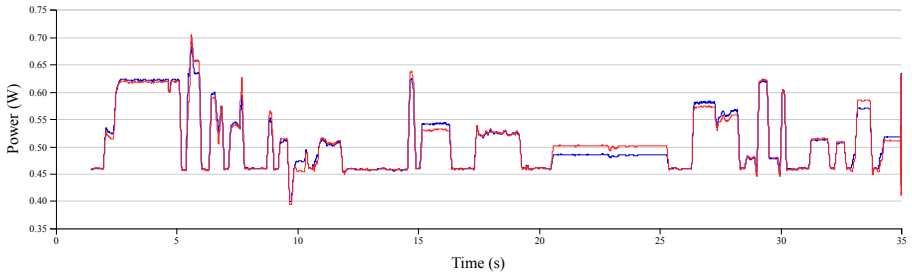


(d) MAPE=1.08%

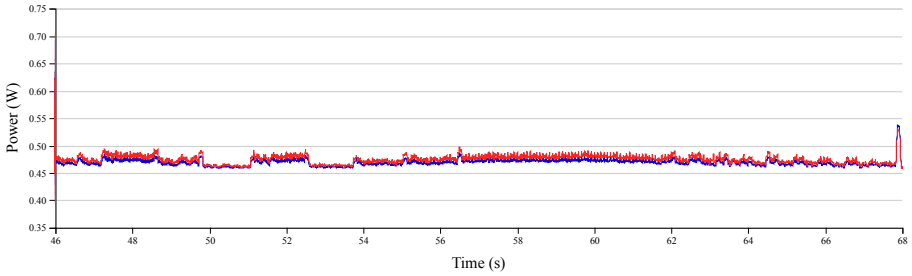


(e) MAPE=1.15%

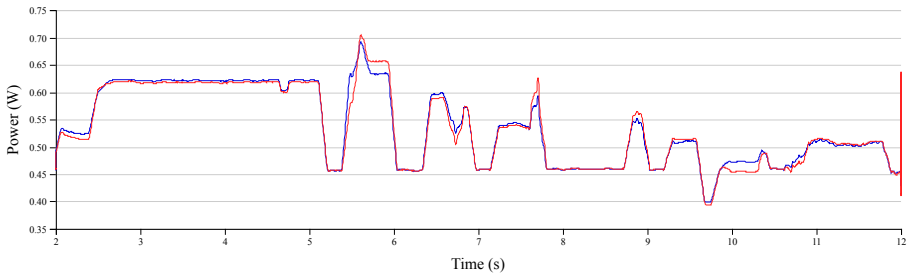
Figure 3.25: Measured power (dark, blue) and predicted power (light, red). $f_{clk} = 1000$ MHz



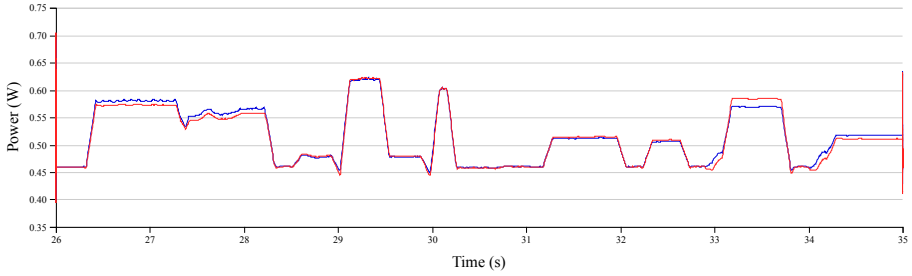
(a) MAPE=1.15%



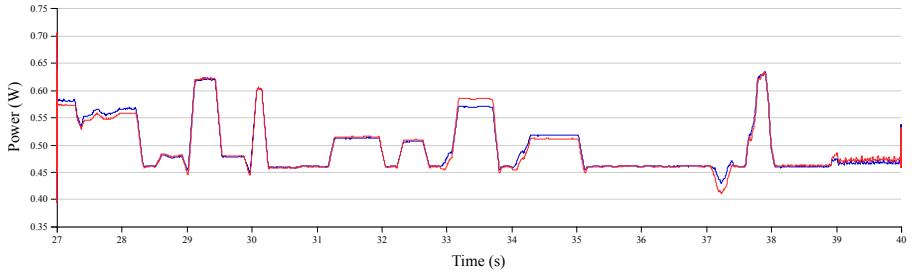
(b) MAPE=1.25%



(c) MAPE=1.04%



(d) MAPE=0.87%



(e) MAPE=0.81%

Figure 3.26: Measured power (dark, blue) and predicted power (light, red) after moving average (window size of 20 samples). $f_{clk} = 1000$ MHz

Table 3.5: Model coefficients for the different folds with a variable removed

Event	Fold 0	Fold 1	Fold 2	Fold 3	Fold 4	Fold 5
0x55 rate	2.12E-10	2.49E-10	2.07E-10	2.09E-10	2.08E-10	2.09E-10
0x04 rate	3.42E-10	2.26E-10	3.27E-10	3.42E-10	3.33E-10	3.60E-10
0x4D rate	2.71E-09	2.53E-09	2.38E-09	2.47E-09	2.35E-09	2.56E-09
const	0.330	0.325	0.335	0.333	0.334	0.329

validation.

An introduction to multivariate regression modelling and PMCs was first presented, before a robust experimental setup for both measuring run-time CPU power consumption and collection of PMCs was presented. Several processing techniques were described, including the use of an offline, unsupervised change-point detection algorithm to extract workload phases for combining data from multiple experiments and for extracting diverse training and testing observations. Correlation analysis was conducted for feature selection and it was found that existing approaches that select events using their correlation with power to be ineffective due to the level of intercorrelation between the resulting events (this is investigated and demonstrated further in Chapter 4) and the fact that the calculated coefficient is highly sensitive to the workloads used.

The variance and coefficient of variation was analysed, learning curves observed, and residuals checked to ensure appropriate training, validation and specification. The coefficients of the model are re-calculated for each DVFS level. The resulting models achieve 6-fold cross-validation MAPEs of 1.91%, 1.93%, 1.49% and 1.40%, for 300 MHz to 1000 MHz, respectively. However, the power coefficient of variation was only in the range of 8.2% to 10.3%.

It was found that some mobile systems (i.e. the system used in this chapter) did not have advanced power management techniques, which limited the variation in power consumption. Chapter 4 improves on this by using a system with such power management techniques enabled, and with a more complex out-of-order CPU (Arm Cortex-A15) which has a significantly higher power variation.

However, the CPU power consumption was found to vary significantly more than reported in existing works when diverse workload phases are utilised. For example, in [238] it was reported that the minimum and maximum observed power for the in-order Arm Cortex-A7 was 1.385 and 1.506 W (difference of 0.121 W), respectively, while the minimum and maximum power for the Arm Cortex-A15 was 4.535 and 5.155 W (difference of 0.62 W), respectively. The authors concluded that it was therefore not necessary to model the power consumption of the Arm Cortex-A7. The differences are 8.3% and 12.8% of the mean for the Cortex-A7 and Cortex-A15 respectively. However, the minimum and maximum power observed at 1 GHz for the in-order Cortex-A8 in this chapter was 0.468 and 0.655 W, respectively, which makes the difference (0.187 W), 33.3% of the mean. This is despite the more advanced power management techniques not being available on this platform. In Chapter 4, the Cortex-A7 and Cortex-A15 CPUs are used and significantly larger power consumption variation was observed.

It was also observed that the R^2 value was smaller at lower frequencies, partially due to static power (which is highly temperature-dependent) having a larger impact on the overall power consumption. Chapter 4 presents a model formula that absorbs the thermal effects and Chapter 5 uses information from online thermal sensors to compensate for the temperature. The model coefficients were inspected and found to vary between cross-validation folds and when different numbers of features

are used, indicating that the coefficients do not have high stability and that there is correlation between the input features. The topic of correlation between input features, coefficient stability, and the effects of these on model performance is investigated in Chapter 4.

The power models are then evaluated on run-time traces as well as on the averaged workload phase observations. It was shown how the techniques employed in this chapter enabled accurate and reactive (when tested on real-time data) models to be developed using average workload phase training observations.

Many of the shortcomings of existing techniques highlighted in this chapter are investigated further in Chapter 4, which uses a modern big.LITTLE system with high power variation, ensures coefficient stability, considers thermal effects on the static power consumption, models the effect of voltage and frequency on the power consumption, and uses one equation for all DVFS levels.

Chapter 4

Accurate and Stable Run-Time Power Modelling

Chapter 3 presented a run-time PMC-based mobile CPU power model and demonstrated its responsiveness when applied to real-time power traces. However, the platform considered had a relatively small variation in power consumption and used a single-core in-order CPU. Furthermore, key problems in existing approaches were highlighted, such as shortcomings related to feature selection and coefficient stability. Moreover, Chapter 3 used a different set of coefficients for each DVFS level and was not able to separate the power consumption into static and dynamic components.

This chapter presents a statistically rigorous and novel methodology for building accurate run-time power models using Performance Monitoring Counters (PMCs) for mobile and embedded heterogeneous multi-processing (HMP) devices, and demonstrates how the models make more efficient use of limited training data and better adapt to unseen scenarios by uniquely considering coefficient stability. These models can be inserted into the operating system to provide accurate, per-core, run-time CPU power estimations to the run-time management (RTM) software. They can also be used as accurate and trusted reference models in design-space exploration in conjunction with a performance simulator, such as *gem5* [30] (as presented in Chapter 6).

The robust model formulation reduces multicollinearity, allows separation of static and dynamic power, and allows a 96× reduction in experiment time while sacrificing only 0.6% accuracy. The effects of temperature changes due to f_{clk} and V_{DD} on the static power consumption are compensated for. This chapter also presents a statistically detailed evaluation of the model, highlighting and addressing the problem of heteroscedasticity in power modelling. Techniques for evaluating the influence of specific training observations on the model accuracy, as well as identifying missing information in the features are also presented. Software implementing the methodology is made available and power models for Arm Cortex-A7 and Cortex-A15 CPUs are built, with 3.8% and 2.8% average error, respectively. The behaviour of the non-ideal CPU voltage regulator under dynamic CPU activity is modelled to improve accuracy by up to 5.5% in situations where the voltage cannot be measured.

The methodology employs statistical rigour throughout each stage of the modelling process, in which novel techniques and insights are introduced and it is demonstrated how each of them improve the

model quality. The approach is illustrated on a device containing an Arm mobile CPU that is also found in the Samsung Galaxy S5 smartphone (released in 2014). It utilises Arm's big.LITTLE technology, having two CPU clusters of significantly differing microarchitectures: one optimised towards greater power-efficiency (quad-core Cortex-A7), and one optimised towards higher performance (quad-core Cortex-A15, see Section 2.4 for a detailed description of the differences between these two micro-architectures). This chapter presents models of both clusters, focusing on the latter to illustrate the approach.

Section 4.1 provides an overview of the four main steps of the modelling methodology with a brief description of the key contributions in each of them. Sections 4.2, 4.3 and 4.4 describe the first three steps in detail: data acquisition, feature selection, and model formulation and validation, respectively. Following this is a direct comparison of the presented model with related work in Section 4.5. Section 4.6 highlights how the voltage supplied to the CPU is dependent on the dynamic CPU activity and introduces a voltage model to significantly improve the power model accuracy in situations where the voltage cannot be directly measured. The key contributions of this chapter are as follows:

1. An experimental demonstration of the importance of model coefficient stability and how simple reported average error alone is ineffective and potentially misleading at indicating model quality;
2. Techniques and insights for employing machine learning to classify PMC events (Section 4.3);
3. An automated and novel PMC event selection methodology that uniquely considers model stability (Section 4.3);
4. A robust and novel model formulation that breaks down static and dynamic power, reduces multicollinearity, compensates for the effects of temperature changes due to f_{clk} and V_{DD} on the static power consumption, and reduces the experiment time by 96× (Section 4.4);
5. Novel techniques for analysing the influence of workloads on the model fit and for detecting missing information in the features;
6. Identification of the effects of non-ideal voltage regulators and a voltage model that compensates for these effects, improving model accuracy by up to 5.5% (Section 4.6);
7. Identification of, and solutions for, heteroscedasticity in CPU power modelling;
8. Two accurate and thoroughly validated run-time PMC power models for an Arm big.LITTLE mobile platform;
9. Software tools for both collecting PMC, power and voltage data (platform specific), and implementing the novel modelling methodology (platform independent), available from [307].

4.1 Proposed Methodology

This section provides an overview of the steps in the modelling methodology (Figure 4.1) and highlights the contributions in each. Step 1 (described in Section 4.2) concerns the experimental setup and method for acquiring the data used to train and test the models. PMCs, CPU power and CPU voltage are collected from an Arm big.LITTLE-based development platform while simultaneously

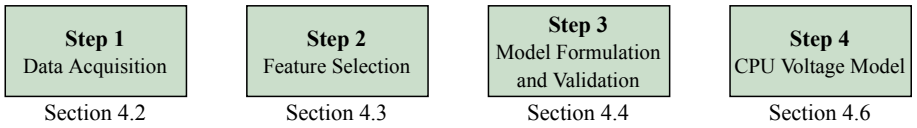


Figure 4.1: Steps of the proposed power modelling methodology

exercising the CPU with a large number of workloads, including ones that utilise parallel programming and the NEON SIMD (single instruction, multiple data) processing unit. To address the lack of existing works using PMC data from real mobile platforms, the software tools are published to aid and encourage future research. The quality of the resulting models inherently depends on the quality of data used to build them and the low overhead and precision of the experimental platform is demonstrated.

Only a limited number of the many selectable PMC events (e.g. L2 cache misses, instructions executed) can be simultaneously recorded; Step 2 (described in Section 4.3) presents the novel feature (i.e. PMC event) selection methodology that uniquely considers the *stability* of the model when choosing model inputs. Furthermore, this section demonstrates how it allows a model to better predict a wider range of workloads, even if they are not well covered in the training data. This is a crucial quality for real-world power models and, although it is not considered in previous work, it is shown that demonstrating stability of a model is perhaps more important than reporting an average error value.

Once the PMC events have been selected, the experiment is run with this selection and the results used to build a linear regression model to predict the CPU power. Step 3 (described in Section 4.4) describes the robust model formulation where knowledge of how power is consumed in CPUs is used, as opposed to adding regression coefficients directly to PMC data as is typical in existing works [244, 243, 33]. The presented formulation reduces multicollinearity, separates dynamic and static power consumption, works with any given voltage and frequency, and, when combined with the added model stability, allows the model building experiment duration to be reduced by 96× while trading off only 0.6% error. In this stage the models are also thoroughly evaluated and a large set of statistical results are provided, allowing the quality of the model to be properly assessed. This section also identifies the problem of heteroscedasticity in run-time power modelling and describes its effects and how to alleviate it.

Furthermore, it is demonstrated how the dynamic CPU activity affects the voltage being supplied to the CPU by the non-ideal voltage regulator, which in turn affects CPU power consumption (described in Section 4.6). In run-time management scenarios, the voltage supplied to the processor cannot be measured and this section therefore presents a novel voltage model, which takes the current frequency and modelled dynamic power consumption as inputs (Step 4). This section also contains a demonstration of how this improves the run-time power estimation accuracy by as much as 5.5%.

Software tools to aid research related to PMCs, power and temperature on mobile devices using real data (platform dependent) and to implement the novel methodologies described in Steps 2 and 3 (platform independent) are published at [307]; raw data, additional graphs and results, model coefficients, and software documentation are also provided.

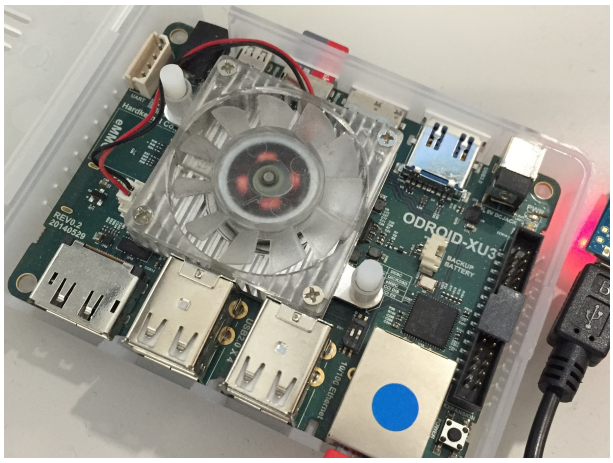


Figure 4.2: Hardkernel ODROID-XU3 Development Board

4.2 Data Acquisition

As highlighted in Chapter 3, there is very little reported research that uses real PMC data from a mobile Arm-based platform due to the lack of an established method of doing so. This section presents the experimental setup, method and accompanying software tools for collecting PMC data, running workloads and measuring CPU voltage and power on a mobile platform (Step 1 of the power modelling methodology). Its low overhead and high precision is demonstrated, which is essential for producing accurate models.

4.2.1 Experimental Platform

An ODROID-XU3 development board by Hardkernel is used to illustrate the approach (Figure 4.2). It utilises a Samsung Exynos 5422 SoC (System-on-Chip) which has an Arm big.LITTLE design containing two types of CPU core: four Arm Cortex-A15s optimised for high performance and four Cortex-A7s optimised for energy-efficiency (a detailed description of this platform is presented in Section 2.4). Each core contains a NEON SIMD processing unit, which is accounted for in the presented model.

4.2.2 Experiment Workflow

Software tools are published which streamline future research on mobile development boards. They include a customised Linux operating system image (Ubuntu 14.01.1) and programs for running experiments and capturing data from PMCs, operating system statistics and the built-in energy sensors on the ODROID-XU3 (Figure 4.3).

The kernel of the operating system (Item A in Figure 4.3) is modified to include the *userspace* frequency governor so that the frequency and voltage can be changed from userspace. A loadable kernel module (LKM) is developed to allow userspace access to the PMCs by setting the *PMUSERENR* register on each CPU core (see Section 3.2). Data-logging software is written in C and inline assembly to access and record the PMCs, power sensor data and operating system statistics (Item B) with a low overhead (Figure 4.4). There is no perceivable power overhead when running the experiment

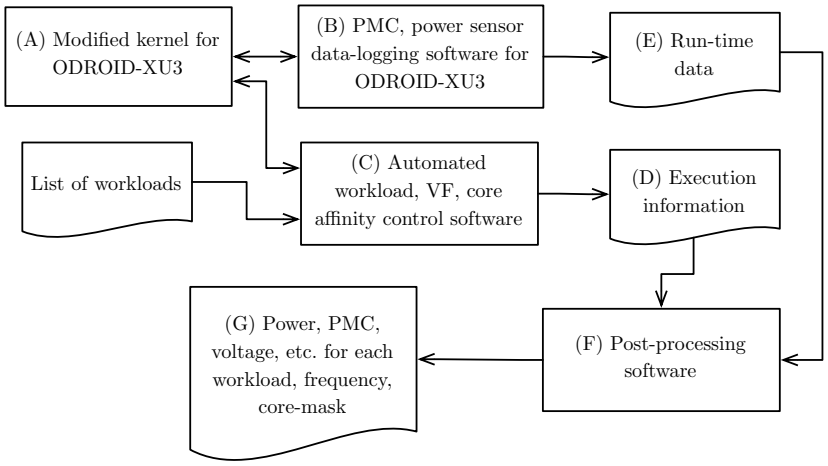
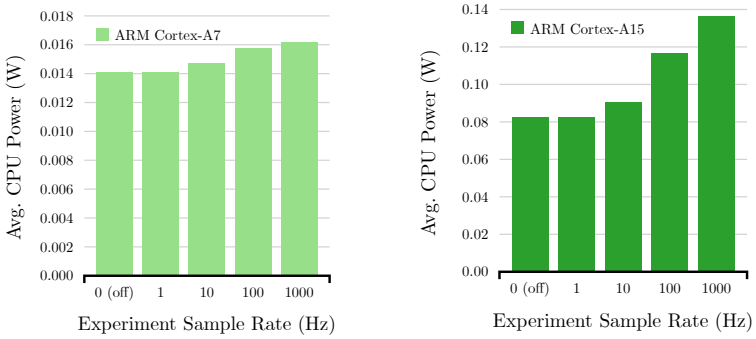


Figure 4.3: Simplified overview of the experimental platform software (corresponding to Step 1 in Figure 4.1).



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

Figure 4.4: Power overhead of experimental platform (@200 MHz, worst-case)

with a sample frequency of 1 Hz and a negligible overhead when running the experiment at 10 Hz. The overhead does not contribute to errors because the PMC data measures all of the CPU activity. However, for high-quality experimental data, the effect of the experimental software should be minimised so it does not interfere with the workloads under test. A sample rate of 5 Hz was used and the workloads were run for an extended period of time to gather a large number of data points to properly account for workload phases. To test the consistency of the experimental platform, the experiments are run 11 times and the deviation observed. The average standard deviation of the measured power (which deviates more than the PMC events) over all 60 workloads, whose average power ranges from 0.23 W to 1.8 W, is 0.0049 W. This indicates a high level of precision and repeatability in the experimental setup which is necessary for building high-quality models.

As well as recording data, the experimental software runs a specified set of workloads (Item C, Figure 4.3) over: 1) any specified list of voltage-frequency points (Cortex-A7 frequencies and Cortex-A15 frequencies can be independently specified, allowing many combinations); 2) any specified set of core masks (control of CPU affinity on both 'big' and 'little' cores); and, 3) any specified number of simultaneous executions of the workload (for exercising multiple cores simultaneously). Having the ability to automatically collect data for many combinations allows the model to be built with a wide range of scenarios.

To keep the overhead of the experiment to a minimum, the data from both Items B and C (Figure 4.3) are captured in a raw format and then combined and processed (Item F) after the data has been recorded. The result is a table of workloads (run on different numbers of cores and at different DVFS levels) against the corresponding CPU power, voltage and PMC data, all averaged over the duration of each workload (Item G).

The software for reproducing this experiment is available as part of the *Powmon* suite of software tools. Many of the features of these tools have been incorporated into the newer, more generic, GemStone suite of software tools.

4.2.3 Workloads

This work uses 60 workloads from a variety of sources, including: 14 from the MiBench [116] suite; 20 from LMbench [190]; 11 from Roy Longbottom [179]; 1 from MediaBench [169]; 5 handwritten workloads; and other workloads that make use of programs such as *MPlayer*, *tar* and *gcc*. This set covers a large range of both realistic and synthetic workloads, including ones that are CPU intensive (e.g. *bitcount*, *neon_mul*, *jpeg_dec*, *dhrystone*), memory intensive (*mp_lp_neon*, *lat_mem_rd_1_256*, *bw_mem_rd*, *cache*) and I/O intensive (*bw_mem_cp_700m*, *par_mem*, *openmp_mem_spd*). Synthetic workloads are tuned to trigger specific CPU behaviour (e.g. accessing particular levels of the memory hierarchy) and use PMC data to ensure the workloads achieve the desired effects. For example, *lat_pagefault* and *mp_lp_neon* are used to cause a large number of TLB (translation lookaside buffer) misses, *cstm_int* is used to cause a large number of mispredicted branch instructions, and *neon_add*, *cstm_fp* and *mp_neon_mflops* are used to cause a large number of unaligned accesses. In order to create a diverse set of training observations, workloads from different benchmarking suites were specifically targeted, as opposed to using workloads all from a single benchmarking suite. All workloads that were successfully compiled on the platform were included in the final results, and no workloads were left out of the error calculations.

In Section 4.3 it is shown how using a diverse subset of these workloads results in a more robust model than when using a more typical set (i.e. MiBench and MediaBench). In Section 4.4 it is shown how the components of the dynamic power prediction vary significantly between different workloads (Figure 4.32).

4.2.4 Power Variation

The experiment in Chapter 3 had a relatively low power variation, and identified existing works to have an even lower power variation. For example, Pricopi *et al.* [238] found the power of a Cortex-A15 (same micro-architecture, in a different implementation) to vary between 4.54 W and 5.16 W on their training workloads. Works on desktop and server class systems also show low power variability. For example, Singh *et al.* [267] model an AMD Phenom and validate using NAS, SPEC-OMP and SPEC 2006 benchmarking suites. Their reported graphs show a minimum measured power of approximately 19 W and a maximum measured power of approximately 27 W. The same benchmarks are also used to validate power models on an Intel Haswell Platform [107]. The development platform and workload selection used in this chapter results in a significant variability in power consumption between workloads (Figures 4.5 and 4.6).

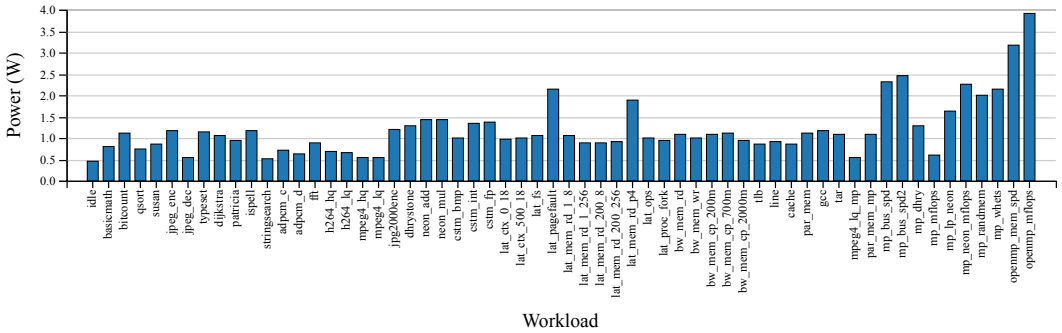


Figure 4.5: Power consumption of each workload at 1.6 GHz with one workload instance running on one core

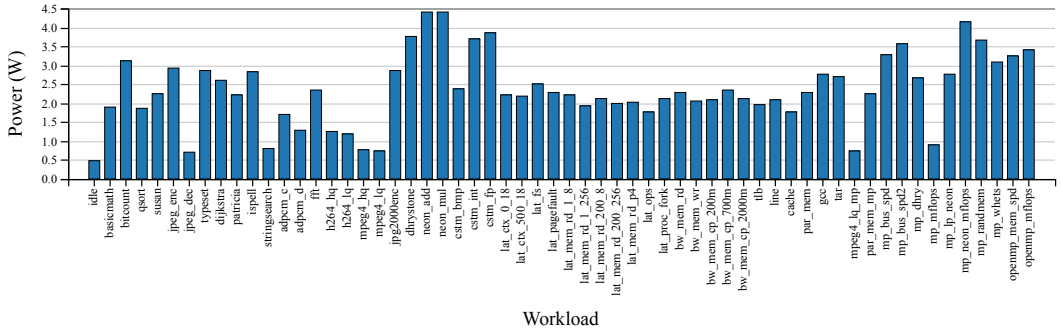


Figure 4.6: Power consumption of each workload at 1.6 GHz with four workload instances running across the four cores

4.2.5 Feature Conditioning

The raw PMC event data was converted to event rate (events-per-second). In the reviewed existing works (and in Chapter 3), the PMC event counts or the PMC event rates were used directly, without analysis of the distribution. In OLS, there is no assumption about normality on the independent variables or dependent variable (the normality assumption for linear regression applies to the errors). However, the model performance is affected by skewness [103].

The normal distribution is defined as follows:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{1}{2\sigma^2}(x-\mu)^2\right) \tag{4.1}$$

where μ is the mean:

$$\mu = \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \tag{4.2}$$

and σ is the standard deviation:

$$\sigma = S_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2} \tag{4.3}$$

In the dependent variable (Figure 4.7a) and many independent variables (Figures 4.7c, 4.7e and 4.7g) there was significant skew. While exponential transforms were effective for specific variables (Figure 4.9), it was found that the natural log transform was consistently effective across all of the

variables (Figures 4.7b, 4.7d, 4.7f, 4.7h). Often, dependent variables are transformed to change the distribution of error terms (see Section 4.4), while the independent variables are typically transformed to remedy non-linearities (transformations on the independent variables typically do not change the error distribution).

While applying transformations can improve one aspect of the model, they can easily result in a violation of one of the assumptions (Section 3.1.6), such as homoscedasticity or linearity, and so transformations must be carefully applied in an iterative manner, with every assumption being checked. Furthermore, the transformations also affect how the coefficients (and goodness-of-fit and confidence intervals) are interpreted [41, 43]. Therefore, no transformation to the dependent or independent variables were made at this point. However, the normality of the errors are evaluated in Section 4.4.2, and transformations are revisited.

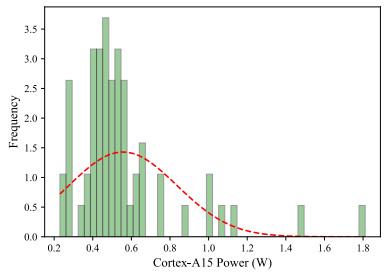
4.3 Feature Selection

While there are many PMC events, only a few of them can be monitored simultaneously on mobile CPUs. On the Arm Cortex-A7 and Cortex-A15 only four and six PMC events (in addition to the Cycle Count) can be monitored simultaneously, respectively (see Table 3.1). The decision of which PMC events to use as input features is key, as will be demonstrated. This section presents a novel methodology for choosing PMC events (Step 2, Figure 4.1) that results in an accurate and *stable* model. Furthermore, this section experimentally demonstrates the importance of stability, analyses several approaches for reducing multicollinearity, gives insight into how many PMC events should be used, and shows how the common practice of using a limited number of similar workloads for training and testing results in a poor power model with an optimistic reported error.

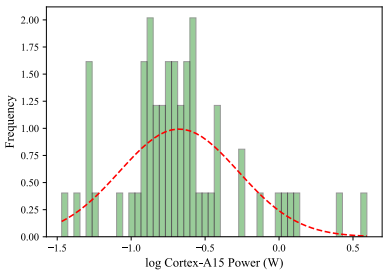
In order to analyse and compare all of the available PMC events, the experimental framework is set to keep the frequency and core-mask constant while running all of the workloads multiple times and changing the recorded PMC events on each iteration; data are collected for 39 and 66 events on the Cortex-A7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

4.3.1 Model Stability

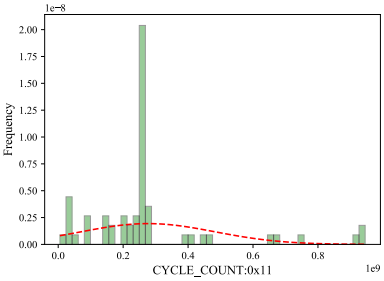
As this section will experimentally demonstrate, the coefficient stability of a CPU power model is critical in making accurate predictions across a wide range of workloads. A model can have a low error and good fit when tested with the training data. However, it still may have significant errors in the coefficients. A stable model has accurate coefficient estimates and is able to understand how each feature *individually* contributes to the overall power consumption. This means that the model is better able to make accurate predictions across a wide range of workloads, even if they are not well represented in the training data. It is impractical to cover all of the possible behaviours that real-world workloads may exhibit in the training workloads used to build the model. Furthermore, a run-time power model is required to estimate the power consumption of workload phases and micro-phases, which are much more diverse than the average behaviour across the full workload. Stability is therefore a crucial property of run-time power models and Section 4.3.7 demonstrates how an unstable model can achieve a very low average error but then perform poorly when tested



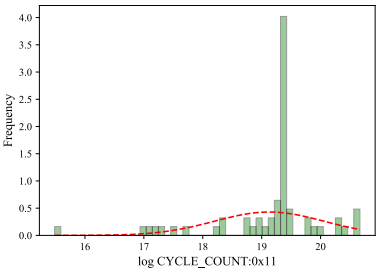
(a) Cortex-A15 Power (W)



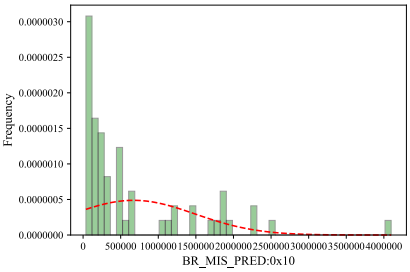
(b) Log Cortex-A15 Power (W)



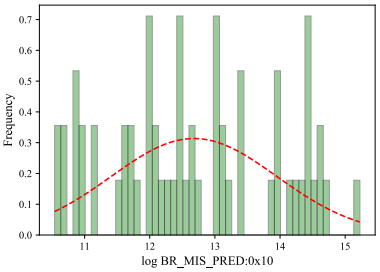
(c) CYCLE_COUNT:0x11



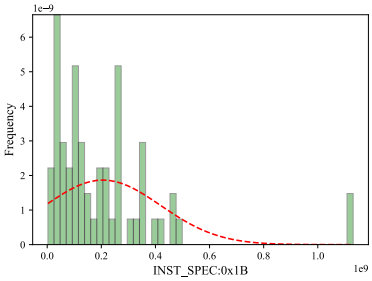
(d) Log CYCLE_COUNT:0x11



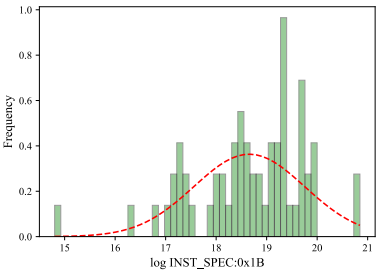
(e) BR_MIS_PRED:0x10



(f) Log BR_MIS_PRED:0x10

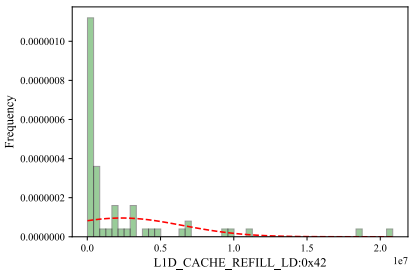


(g) INST_SPEC:0x1B

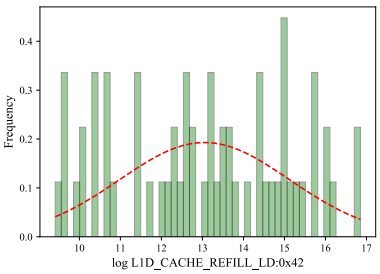


(h) Log INST_SPEC:0x1B

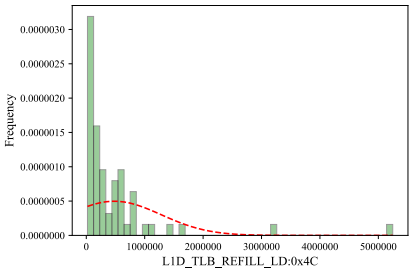
Figure 4.7: Distribution of the independent variable and PMC event rates both without any transformation and with a log transform. A normal distribution is superimposed.



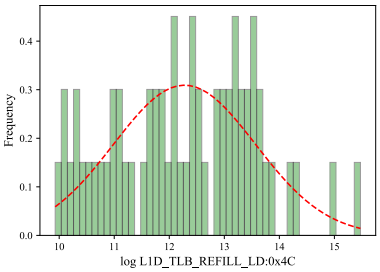
(a) L1D_CACHE_REFILL:0x42



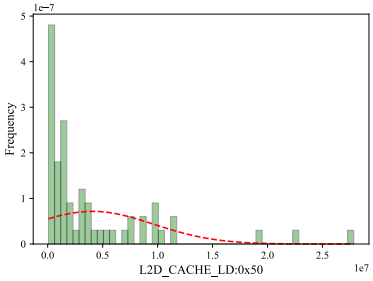
(b) Log L1D_CACHE_REFILL:0x42



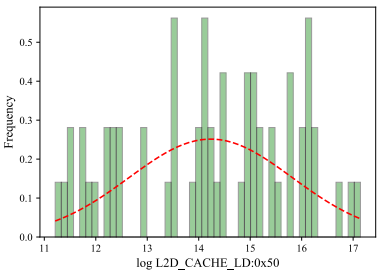
(c) L1D_TLB_REFILL:0x4C



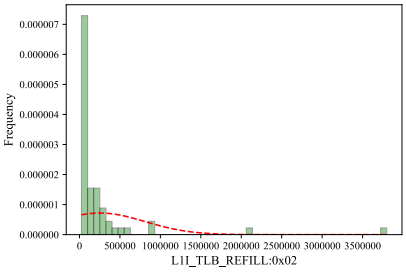
(d) Log L1D_TLB_REFILL:0x4C



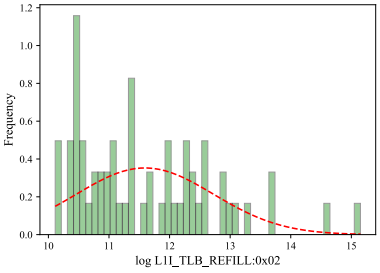
(e) L2D_CACHE_LD



(f) Log L2D_CACHE_LD

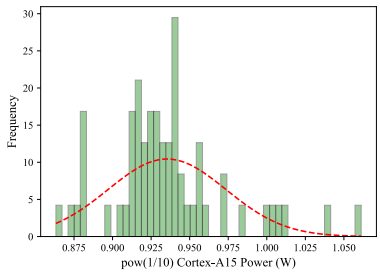


(g) L1I_TLB_REFILL

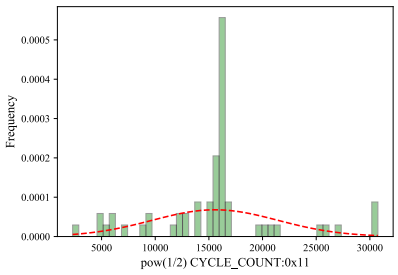


(h) Log L1I_TLB_REFILL

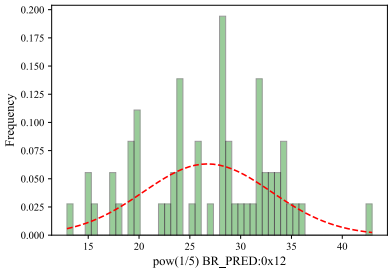
Figure 4.8: Distribution of PMC event rates both without any transformation and with a log transform. A normal distribution is superimposed.



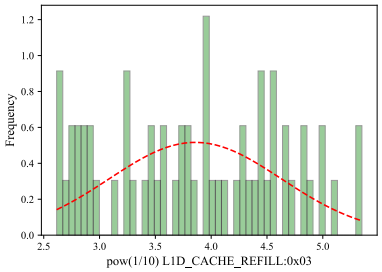
(a) Cortex-A15 Power (W) exp(1/10)



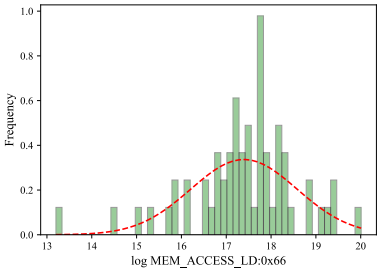
(b) CYCLE_COUNT:0x11 exp(1/2)



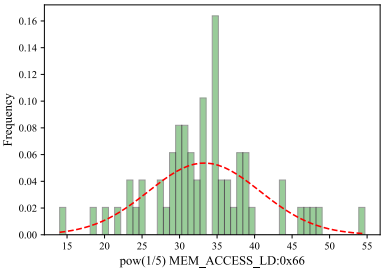
(c) BR_PRED:0x12 exp(1/5)



(d) L1D_CACHE_REFILL:0x03



(e) Log MEM_ACCESS:0x66



(f) MEM_ACCESS:0x66 exp(1/5)

Figure 4.9: Distribution of PMC event rates with various exponential transforms

with a wider range of workloads. Unstable models are typically overly sensitive to small changes in the inputs and can make wild predictions with certain input combinations.

As this section will show, the two requirements in achieving a stable model are:

1. Diverse training observations (training workloads);
2. Carefully selected and conditioned input features with little or no multicollinearity.

4.3.2 Multicollinearity

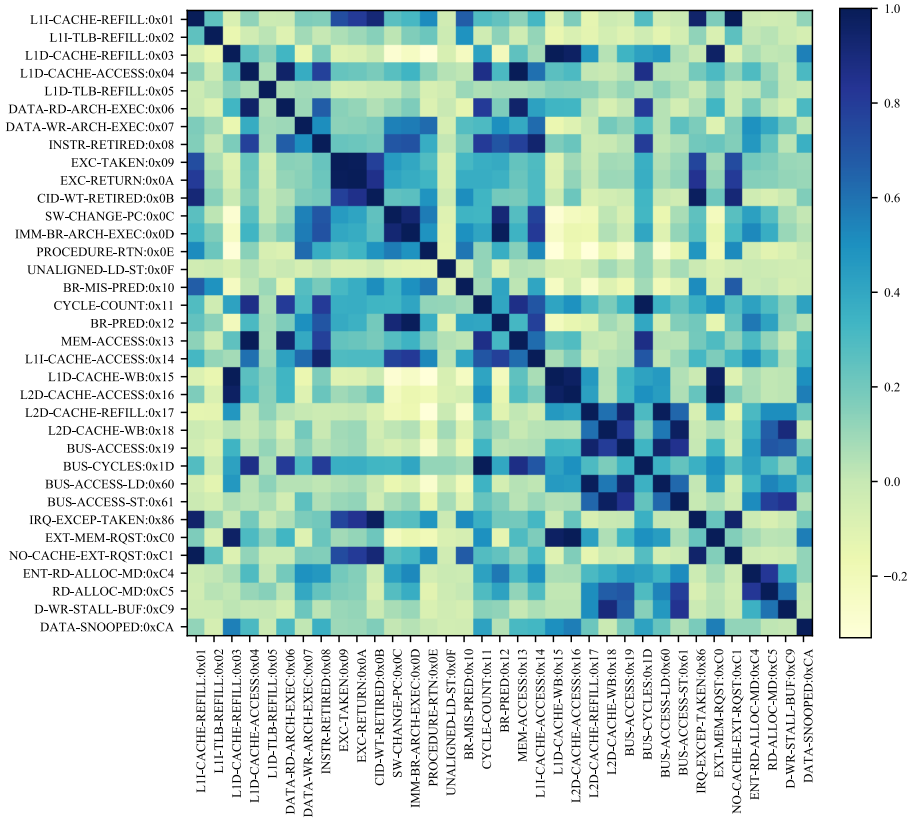
An important consideration when choosing the events is multicollinearity, which occurs when two or more independent variables in the linear regression model have inter-correlation (a relationship) between them. While this does not necessarily have a *direct* impact on the reported accuracy or overall fit, and the estimates are still Best Linear Unbiased Estimates (BLUE), it causes errors in the model coefficients and makes the model overly sensitive to changes in the inputs; they are said to be *unstable*. Furthermore, with unstable coefficients the standard errors of the estimates will be higher than they would be in the absence of multicollinearity. In the presence of multicollinearity, the coefficient estimates, $\hat{\beta}$, tend to have large sampling variability [161, 208]. As identified in Section 3.1.2, the normal equation is only valid if the elements of $\boldsymbol{X}$ are linearly independent.

The individual components of a CPU work together in executing instructions, and there is naturally very high inter-correlation between the PMC events (Figures 4.10 and 4.11). Creating a PMC-based power model with acceptable levels of inter-correlation in the input features takes great care and this has not been considered in detail in existing power modelling works. The remainder of this section analyses several techniques for reducing multicollinearity while extracting information explaining the power consumption, and evaluates their effectiveness and suitability for run-time power models. The correlation matrices in Figures 4.10a and 4.11a therefore show the starting point for these techniques. When the PMC events in the correlation matrices have been re-ordered using information extracted from unsupervised machine learning techniques, described later in this section, groups of PMC events that correlate highly with each other can be observed (Figures 4.10b and 4.11b). However, this does not provide any insight into which PMC events provide useful information for estimating power consumption.

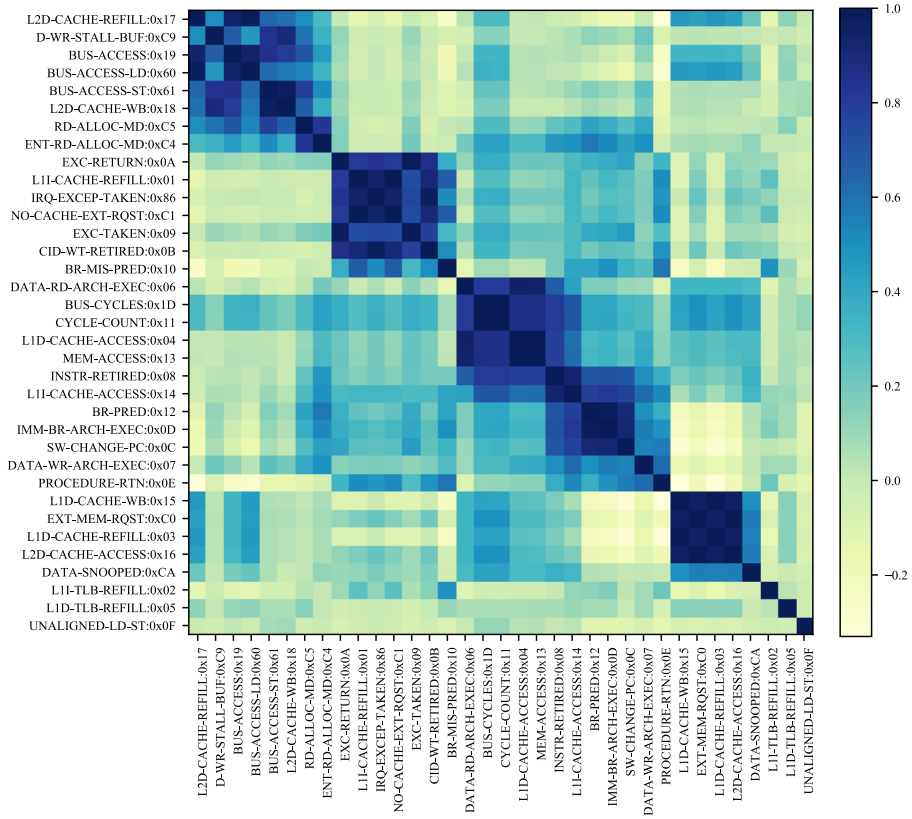
4.3.3 Principle Component Analysis

Principle Component Analysis (PCA, also known as the *Karhunen-Loèvre* transform) is a dimensionality reduction technique which is primarily used in lossy data compression, feature extraction and data visualisation [38]. PCA effectively projects the high-dimensionality data onto a lower dimensionality space (known as the *principle subspace*) such that the variance of the projected data is maximised, using a projection that best represents the data in a least-squares sense [124, 144, 90, 212].

PCA was used to reduce D dimensions to L dimensions, where $L \leq D$, maximising the variance of the projected data. Because the resulting principle components are orthogonal, there is no multicollinearity between them. In this section, PCA was used to understand how many orthogonal components make up the full set of PMC event data and whether applying PCA to the input features is effective in improving model stability.

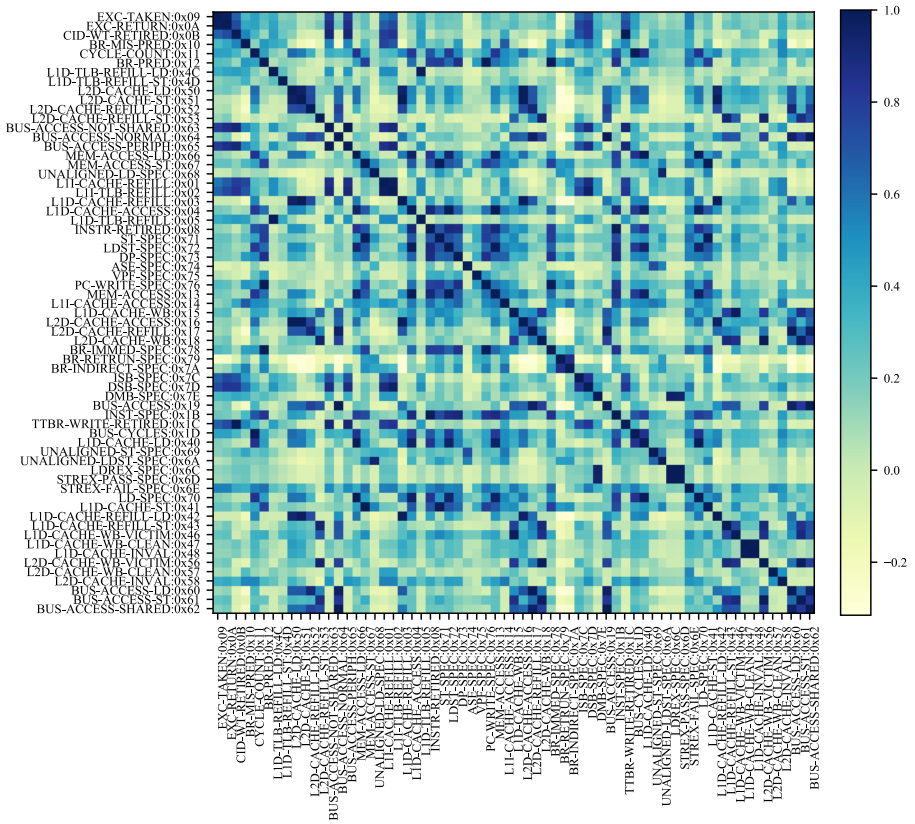


(a) Arm Cortex-A7 events ordered by event ID

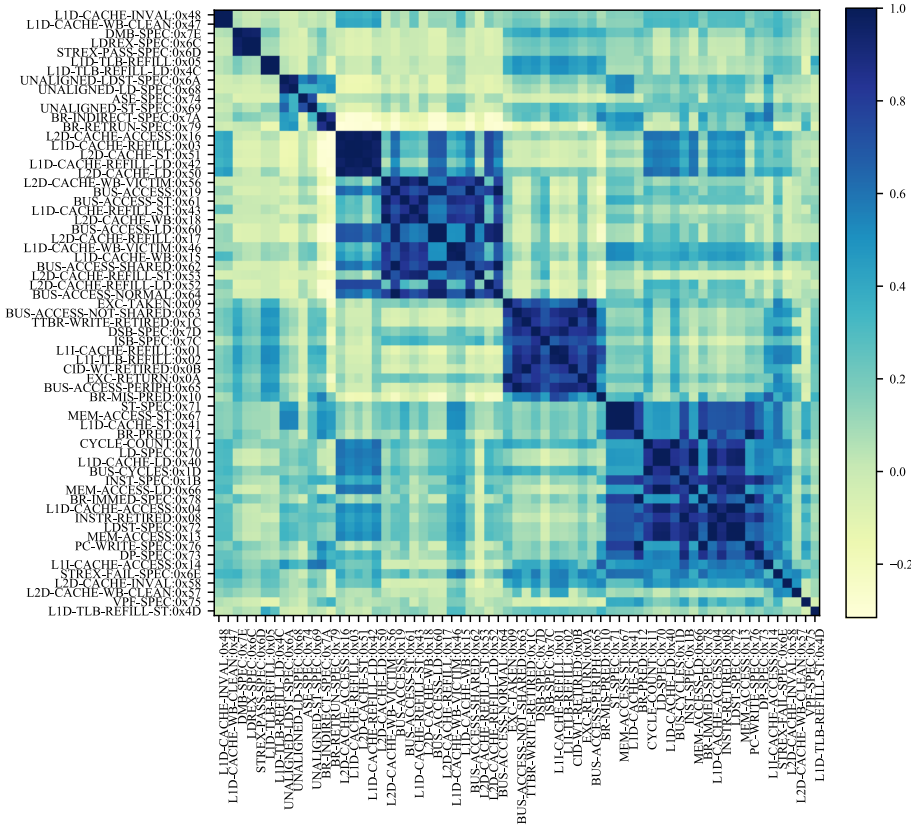


(b) Arm Cortex-A7 PMC events ordered using the results from the HCA (see Figure 4.17)

Figure 4.10: Arm Cortex-A7 PMC event correlation matrices. Shading shows the event correlation.



(a) Arm Cortex-A15 events ordered by event ID



(b) Arm Cortex-A15 PMC events ordered using the results from the HCA (see Figure 4.16)

Figure 4.11: Arm Cortex-A15 PMC event correlation matrices. Shading shows the event correlation.

The input was the matrix, $\boldsymbol{X}$ which was a $n \times p$ matrix. Each column (containing the observations for a single feature, or predictor) was denoted $\boldsymbol{x}_p$, and the mean for each column is given by:

$$\bar{x}_p = \frac{1}{n}\sum_{i=1}^{n} x_{i,p} \tag{4.4}$$

and the standard deviation is given by:

$$\sigma_p = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_{i,p} - \bar{x}_p)^2} \tag{4.5}$$

Mean normalisation and feature scaling was first applied:

$$x_{i,p} = \frac{x_{i,p} - \bar{x}_p}{\sigma_p} \tag{4.6}$$

$\boldsymbol{X}$ was then transposed and the covariance matrix, Σ, was then calculated using:

$$\Sigma = \frac{1}{n}\sum_{i=1}^{n}(x_i)(x_i)^T \tag{4.7}$$

Or, in vector form:

$$\Sigma = \frac{1}{n}(\boldsymbol{X}^*)\boldsymbol{X} \tag{4.8}$$

Where the conjugate transpose, $\boldsymbol{X}^*$, is equivalent to the regular transpose, $\boldsymbol{X}^{\boldsymbol{T}}$, in this case. The resulting Σ is an $n \times n$ matrix. The correlation matrix was also tested and produced similar results.

The eigenvalues and eigenvectors of Σ are then calculated. The eigenvectors are ordered by their eigenvalue to form $\boldsymbol{U}$, which is an $n \times n$ matrix where each column is an eigenvector ($\boldsymbol{u}$):

$$\boldsymbol{U} = \begin{pmatrix} \boldsymbol{u}_1 & \boldsymbol{u}_2 & \dots & \boldsymbol{u}_n \end{pmatrix} \tag{4.9}$$

The first L vectors are selected to reduce to L dimensions. This new matrix is called $\boldsymbol{U}_r$.

$$\boldsymbol{z} = \boldsymbol{U}_r^T \boldsymbol{X} \tag{4.10}$$

To obtain the approximation from the reduced (compressed) data:

$$\hat{\boldsymbol{X}} = \boldsymbol{U}_r \boldsymbol{z} \tag{4.11}$$

Typically, the minimum value of L that retains 90%, 95% or 99% of variance is used, depending on the application, although there is no definitive rule [297, 213, 295]. PCA was first applied to the PMC event exploration experiment data (experiment repeated to capture all PMCs at the single DVFS level of 1000 MHz) and it was found that only 21 principle components are required to capture 99% of the variation in the full set of 66 PMC events (Figure 4.12). However, while these 21 components capture $>$ 99% of the variance in the full set of PMC events, it is not known how much of this information is useful in predicting the power consumption.

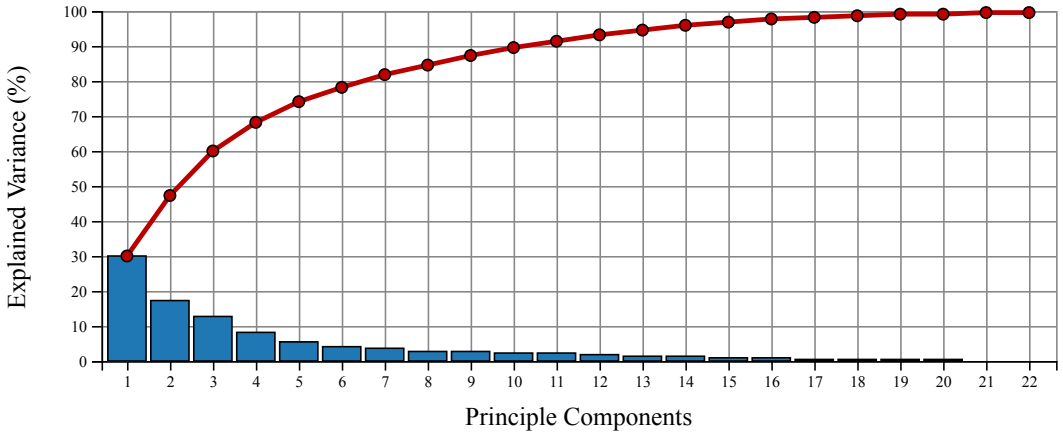


Figure 4.12: Explained variance and cumulative explained variance for each principle component

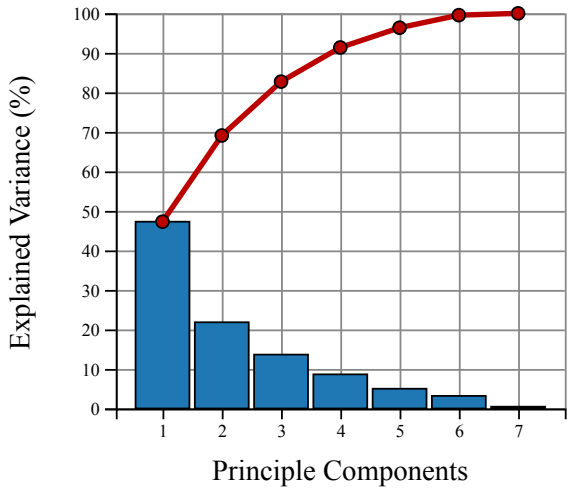


Figure 4.13: Explained variance and cumulative explained variance for each principle component

It was proposed that PCA could be applied to the seven (including the cycle counter) PMC events, identified as being useful in predicting power, to remove the multicollinearity. For example, PCA was applied to the seven PMC events chosen in the latter parts of this section. The first principle component alone captured over 45% of the variance in all seven PMC events, and five principle components captured 95% of the variance in all seven PMC events (Figure 4.13). When the principle components were used to build the model instead of the seven chosen PMC events, the cross-validated MAPE was the same (2.811% with PCA, 2.813 without PCA) while the VIF of the input features was reduced from 2.25 to 1.00. All principle components were statistically very significant ($p < 0.0001$).

While PCA was effective in removing multicollinearity, the components of the model lose their physical meaning. Furthermore, the inputs are compressed without any information from $\boldsymbol{Y}$ (i.e. valuable information regarding $\boldsymbol{Y}$ may be discarded). Moreover, PCA adds overhead to run-time power estimation. For these reasons, other methods of removing multicollinearity were explored in the remainder of this section.

Instead of using PCA for feature extraction, the resulting loadings can be used for selecting the inputs features. However, this selection method does not consider how useful each component is to predicting the power consumption specifically.

4.3.4 Event Clustering

Unsupervised machine learning techniques were used to understand the relationships between the different events within the CPU and the complex behaviour of the different workloads used to exercise them. Such techniques discover *clusters*, *sub-classes* or *groups* of similar observations in the unlabelled data, providing insights into relationships within the data. These techniques were used to classify PMC events in this chapter and are used to group workloads in Chapter 6.

The most commonly used clustering technique is the *k-means* algorithm [212, 90] which was tested first. It groups the data points into K cluster centroids using the following steps:

1. Initialising the K cluster centroids $(\boldsymbol{\mu}_0, \boldsymbol{\mu}_1, \ldots \boldsymbol{\mu}_{K-1})$ in random locations
2. Iteratively:
   (a) Assign each observation in $\boldsymbol{X}$ to the nearest centroid (distance measured using the *Euclidean distance*, see Equation 4.12);
   (b) Move the position of each centroid to the mean position of all of the observations assigned to it (a typical approach to the case of there being no observations assigned to a centroid is to remove that centroid. An alternative is to re-initialise this centroid to a random location);
   (c) The process is repeated until there is little or no change in $\boldsymbol{\mu}_i$

A typical approach to randomly initialising the cluster centroids is to randomly select K observations and setting these as the locations. The algorithm can converge to local optima and it is therefore common practice to run the algorithm multiple times and then selecting the solution that gives the lowest cost, J.

It was found that k-means was not well suited for the task of classifying PMC events for several reasons. Firstly, the algorithm had a tendency to produce equal-sized, spherical, clusters, which is not appropriate for this problem; it is expected that a large group of PMC events are related, while others belong to much smaller groups. Mahalanobis distance has been shown to be effective when the clusters are expected to be elliptical in shape, but has challenges with both performance and initialisation, and it requires some prior information about the geometric characteristics of the input data to perform well [143, 192, 188, 57, 82, 202].

Additionally, the k-means algorithm does not provide any intuition for an appropriate value of K, which must be supplied as an input. There are several statistical and visual techniques for guiding a selection of K, such as the "Elbow" method and cross-validation [110], but there is no consensus on one that works effectively across many scenarios and they make assumptions on the distribution of the underlying data [230, 290]. The selection of K is ambiguous as there must be a penalty for selecting higher numbers of clusters, as the maximum accuracy is obtained when each observation is in its own cluster. Despite the k-means algorithm being published over 50 years ago, developing an algorithm to select a suitable value of K across multiple scenarios is the topic of continuing research [172, 226, 94, 312, 325, 328]. Another disadvantage of the k-means method is its sensitivity to the initialisation of the centroids and finding new approaches to improve initialisation is also the subject of recent works [66, 261, 240, 164, 300].

Finally, the *euclidean* distance, which is the standard and most popular distance metric used for k-means clustering [202], was found to not be effective for clustering PMC event data and yielded poor

results, even after standardisation of the input features. Some works have modified the standard k-means algorithm to make it more applicable to a wider range of clustering problems by utilising different distance metrics. For example, Xing, Ng, Jordan and Russell [316] present a method for incorporating distance learning to k-means and show how it can improve clustering performance when compared to standard k-means using a fixed euclidean distance. Su and Chou [202] present a novel distance metric based point symmetry that overcomes the shortcomings of Mahalanobis distance.

Unlike the k-means algorithm, Hierarchical Cluster Analysis (HCA, also known as hierarchical clustering) is hierarchical, where clusters can have sub-clusters, which can in turn have sub-clusters, and so on. The PMC data, which counts events related to CPU components and sub-components is hierarchical in nature, as opposed to flat. Hierarchical clustering also avoids the problem of selecting the number of clusters altogether.

For clustering of PMC events and workloads using PMC event data, it was found that using Pearson's correlation (Equation 4.14) without feature standardisation was more effective than using euclidean distance with (or without) standardisation or Manhattan distance (Equation 4.13) with (or without) standardisation. This is because there is a very large disparity between event rates, for example, L2 memory events occur significantly less frequently than CPU execution events. For identifying clusters of workloads based on PMC data, it is desirable to group workloads where the pattern of events, across all events, is similar; the absolute rates are not important.

$$d_{euc}(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{4.12}$$

$$d_{man}(x, y) = \sum_{i=1}^{n} |(x_i - y_i)| \tag{4.13}$$

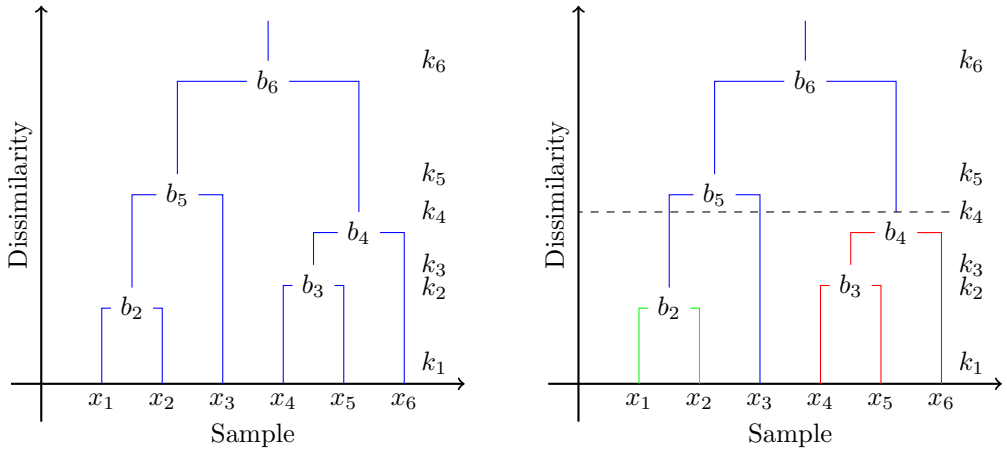
$$d_{cor}(x, y) = 1 - \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{4.14}$$

Specifically, agglomerative (bottom-up) hierarchical clustering was used and nearest neighbour algorithm, using single linkage, was employed, which uses the following distance measure:

$$d_{min}(\mathcal{D}_i, \mathcal{D}_j) = \min(\boldsymbol{x}, \boldsymbol{x}' : \boldsymbol{x} \in \mathcal{D}_i, \boldsymbol{x}' \in \mathcal{D}_j) \tag{4.15}$$

At the start of the algorithm, each element is in its own cluster and at each step two clusters (that contain the closest pair of elements that are not already in the same cluster) are combined. The end of the algorithm is when all elements are in the same cluster. For single-linkage clustering, the distance between two clusters is defined by the two elements between the clusters that are closest. In contrast, the farthest-neighbour algorithm (complete linkage clustering) uses the distance between the two pairs in each cluster that are furthest away from each other. One advantage of complete linkage is that it generally results in clusters that are compact and roughly equal in size, and avoids the *chaining effect* which can occur with single linkage clustering [90]. However, single linkage proved effective for both identifying groups of workloads and identifying groups of events.

Because it is hierarchical, the number of clusters does not have to be decided in advance, and the



(a) Without clustering

(b) With defined clusters

Figure 4.14: Dendrogram Example

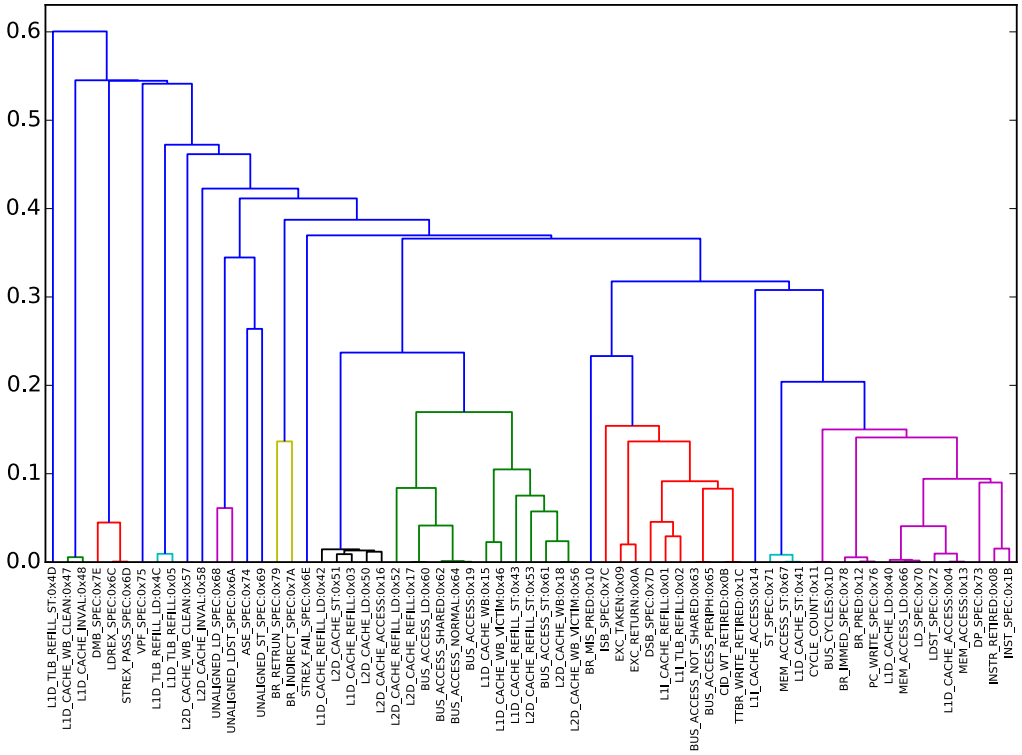


Figure 4.15: Dendrogram showing PMC Event clustering. Clusters formed by "cutting" the dendrogram at a similarity metric of 0.18.

results are viewed as a tree in the form of a *dendrogram* (Figure 4.14a), which has the dissimilarity index on the *y-axis* and the observations (or samples) on the *x-axis*. The lower a common branch of a set of samples, the more closely they are related. To split the samples into a distinct number of clusters, the dendrogram is "cut" at a desired level on the dissimilarity scale. Unique branches under this "cut" are designated as clusters. For example, Figure 4.14b shows the three clusters that result when the dendrogram is cut at $k = 4$.

The dendrogram showing the results of the HCA of the PMC events was plotted and a suitable

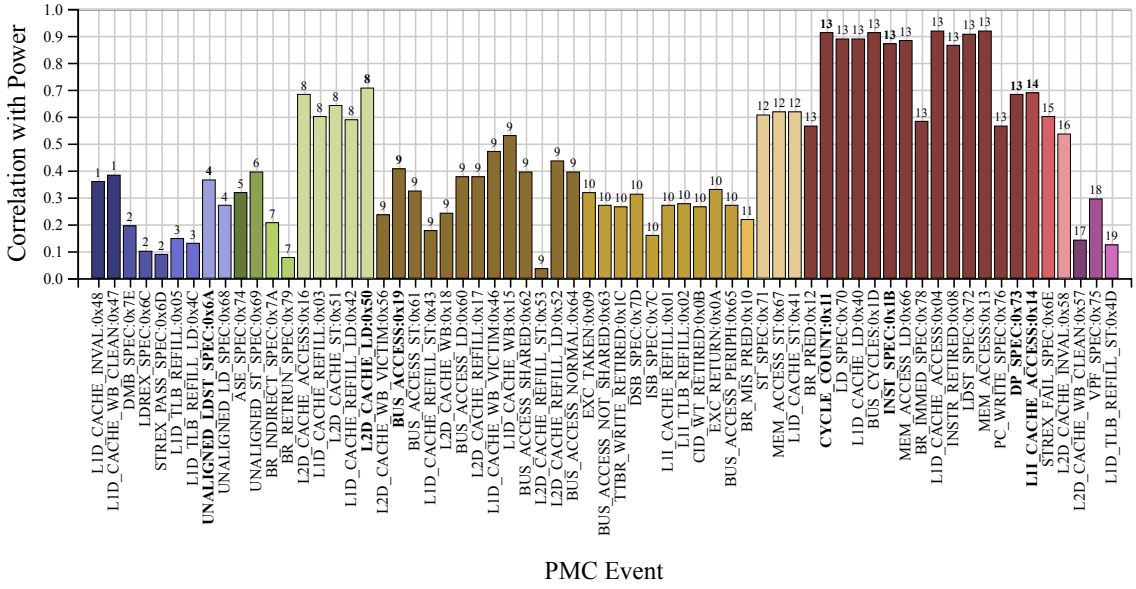


Figure 4.16: Correlation of each Arm Cortex-A15 PMC event with power, grouped, coloured and labelled by cluster number, with PMC events chosen by the automated algorithm labelled in bold font

clustering threshold was identified manually using visual inspection to obtain an appropriate number of clusters (Figure 4.15). One observation was that the L1D TLB store misses (`0x4D`) has high dissimilarity with the other PMC events. The instructions retired (committed, `0x08`) and the instructions speculatively executed (`0x1B`) are closely correlated, despite their absolute values being significantly different, as expected. The fact that L1D TLB LD refills (`0x4C`) and L1D TLB (LD or ST) refills are in the same group together suggests that there are more L1D TLB LD misses than ST misses occurring in the training set. Through applying the same logic, there are more unaligned loads (`0x68`) speculatively executed than unaligned stores. L1D cache misses and L2 cache accesses are highly correlated (as expected).

The HCA results provided insights into PMC behaviour among the training workloads and aids the feature selection. For example, based on insights from the dendrogram, including both the cycle count event (`0x11`) and bus cycles (`0x1D`) would be ineffective and problematic as they are closely related; they contain repeated information meaning that the model would not benefit from having both events instead of just one of them, and including both events would likely cause severe multicollinearity issues.

Selecting events from different clusters therefore results in diverse information being provided to the model about the current operation of the CPU.

However, the information gained from HCA does not aid in understanding whether events are useful in predicting the power consumption. The Pearson product-moment correlation coefficient was used to calculate the linear correlation of each PMC event with the CPU power consumption (see Section 3.8). The correlation was then combined with the HCA results (Figure 4.16). It was observed that PMC events with a very high correlation with power (>0.75) were all in the same cluster, highlighting how choosing PMC events on correlation alone results in similar events, which means the model will have intercorrelated inputs that only provide it with a narrow range of information.

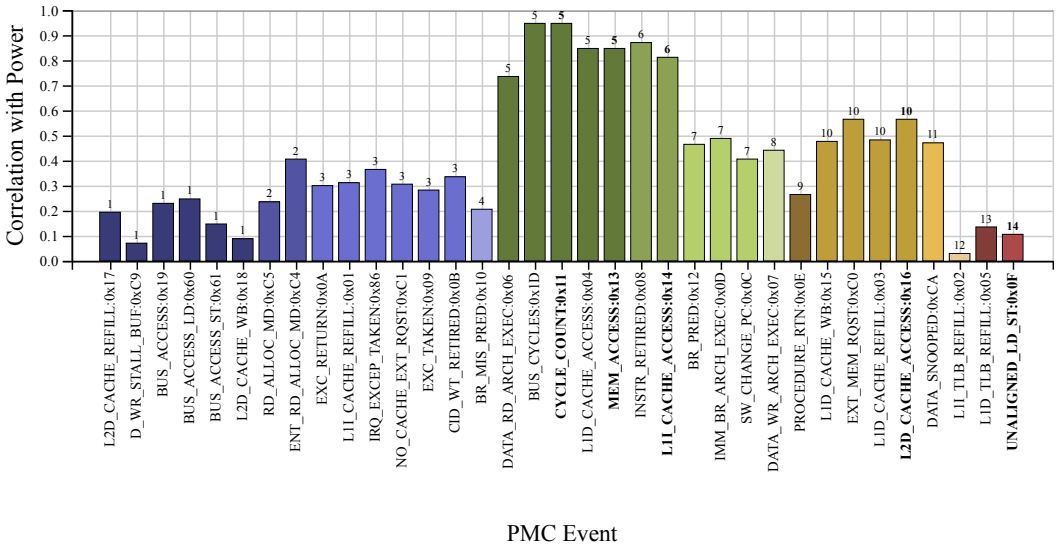


Figure 4.17: Correlation of each Arm Cortex-A7 PMC event with power, grouped, coloured and labelled by cluster number, with PMC events chosen by the automated algorithm labelled in bold font

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

Figure 4.18: Algorithm of the first stage of the PMC event selection method

While Figure 4.16 makes it straight-forward to observe the HCA groupings and the correlation with power simultaneously, it is important to refer back to the dendrogram simultaneously. For example, note that three events in cluster 13 have a significantly lower correlation (`0x12`, `0x78`, `0x76`). These events are all related to program control flow and the dendrogram shows these three events to be highly correlated and in a sub-group of cluster 13. Furthermore, the automated selection approach presenter later chooses three events from cluster 13. However, when consulting the dendrogram, there is high dissimilarity between these three events (`0x11`, `0x73` and `0x1B`) of the same group.

A good PMC selection is made by choosing PMC events that have a high correlation with power, but avoiding repeatedly selecting events from the same cluster. However, a decision of how many clusters to group the PMC events into and how much to prioritise clustering or correlation has to be made, requiring intuition and further experimentation.

4.3.5 Variance Inflation Factor

To quantify multicollinearity (and therefore give an indication of the coefficient stability) the Variance Inflation Factor (VIF) for each PMC event was calculated. To find the VIF for a particular independent variable, an ordinary least squares (OLS) linear regression model is built to predict that variable (i.e. making it the dependent variable) using the others (as independent variables). The resulting R^2 value from the model is used to calculate the VIF:

$$\text{VIF} = \frac{1}{1 - R^2} \tag{4.16}$$

A VIF of one, for example, indicates that there is no correlation between that independent variable and the others. If an independent variable has a VIF of 10, for example, it would indicate that the variance of that predictor coefficient is 10× larger (and the standard error of that predictor coefficient is therefore 3.2× [square root of 10] larger) than if there was no multicollinearity present. It is widely considered that, as a general rule of thumb, a VIF over five or ten [162, 146, 266, 77, 63] indicates that there are strong multicollinearity problems.

A set of PMC events should therefore be carefully chosen to provide the model with the largest possible amount of *unique* information useful for predicting the power, without providing duplicated information which results in multicollinearity; this is key to building an accurate and stable model. Choosing PMC events solely on their correlation with the overall CPU power, for example, results in a poor model because PMC events that correlate well with power also correlate well with each other, starving the model from valuable information contained in other events and causing multicollinearity problems.

4.3.6 Automated PMC Event Selection Method

A simple, two-stage, automated method for selecting optimum PMC events that provide the model with the largest possible amount of information for predicting power with minimum multicollinearity was developed using the knowledge from the HCA and VIF analysis.

The first stage uses the forward stepwise regression automatic search method to select features one-by-one, that add the largest insight into power consumption *given* the previously selected PMC events (Figure 4.18). At each step, a new feature is added to the model. A series of regression models were developed, testing each of the candidate features in turn. The R^2 value was the primary optimisation metric used to select the new feature, while the VIF and P-value for t^* statistic were also monitored and the chosen feature discarded if either one was unacceptably large (e.g. 20 and 0.01 respectively).

The previous analysis showed that the Cycle Count (`0x11`) should always be included in the selected set of PMCs as it contains unique information that is useful for predicting the power. Therefore, by adding the Cycle Count to the list of selected PMCs first (line 2, Figure 4.18) one can find a superior set of PMC events because the remainder of the algorithm finds the optimum events *given* the information in the Cycle Count. The algorithm then finds the next best PMC event to add to the set of model inputs by building a regression model to predict power for each additional PMC event with the existing selected PMC events also as inputs. The PMC event that results in the

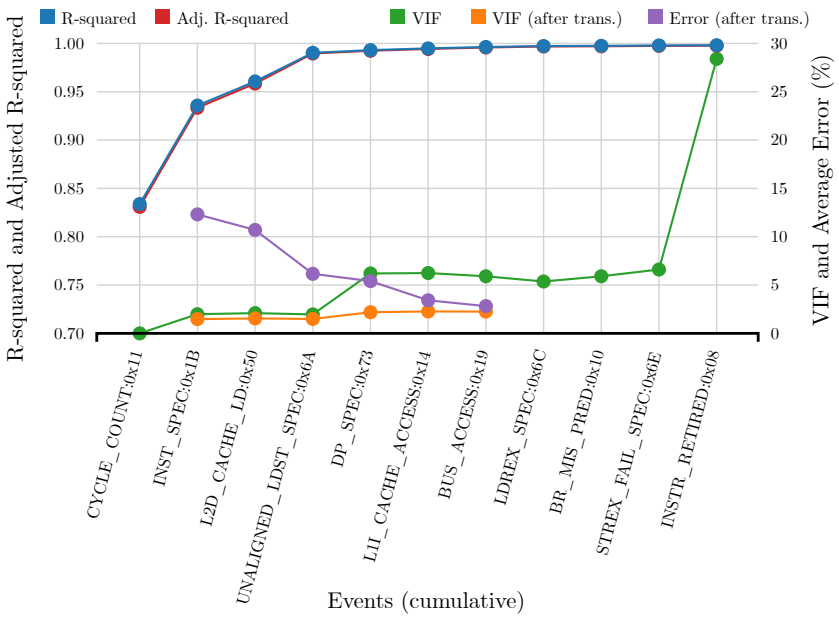


Figure 4.19: R^2, Adj. R^2, VIF of selected PMC events (cumulative) from Stage 1. Also shows error and VIF after transformation in Stage 2

most improved R^2 value is then added to the selection. For the Cortex-A15 example, the algorithm chooses the seven PMC events (highlighted in bold in Figure 4.16) from five different clusters and the chosen events do not necessarily have a high correlation with power, with one chosen event having a correlation as low as 0.36 (UNALIGNED_LDST_SPEC:0X6A).

Figure 4.19 shows how the R^2 and the Adjusted R^2 (adjusted for the number of predictors in the model) increase as each selected PMC event is added to the model (one-by-one), while the VIF (indicating the presence of multicollinearity) also increases. After choosing the fifth event (`0x73`) the VIF suddenly increases to a value over five (indicating multicollinearity problems) while the R^2 and Adjusted R^2 only marginally increase, suggesting that four PMC events (including the Cycle Count) is an optimum number to choose. However, by not utilising all seven available counters the model is not making use of all of the available information. Furthermore, from building the model (Section 4.4) with different numbers of PMC events, it was found that, despite only a marginal increase in R^2, there was a significant decrease in average error between using four and seven counters (see the *Error after trans.* line in Figure 4.19). To utilise the full seven available PMC events, the multicollinearity had to first be further reduced.

The second stage of the PMC event selection method takes the result of the first stage and further reduces multicollinearity allowing the model to use as many events as possible (therefore obtaining as much information as possible and improving accuracy) while maintaining stability. Table 4.1 shows the chosen PMC events from stage 1 for the cortex-A15 model presented in this chapter, along with the VIF and cluster number for each one, which was calculated using the full set of 60 workloads.

The method first identifies relationships between specific PMC events that contribute significantly to the multicollinearity within the chosen set of PMC events using the VIF. As each PMC event is added to the model, the collinearity between that event and the existing events can be understood by monitoring the change in the individual VIFs for each PMC event. In the example of the

Table 4.1: First seven selected events for the Cortex-A15 with their corresponding cluster number and VIF from Stage 1

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

Cortex-A15 model, Figure 4.19 shows that the VIF rises significantly when event 0x73 (DP_SPEC) is included in the model. By looking at Table 4.1 (which shows the VIF of each individual PMC event when all seven events are included in the model together) it can be seen that event 0x1B and 0x73 both have particularly high VIFs. This indicates that the increase of the average VIF among all of the selected events is caused by collinearity between these two events.

Once strong collinearity between specific events has been identified, a transformation to remove the repeated information needs to be made. In the case of the Cortex-A15 example, PMC event 0x1B counts *all* instructions speculatively executed and PMC event 0x73 counts the *integer* instructions speculatively executed, which means that event 0x73 is also counted within event 0x1B. Both events are required; one of the events cannot be derived without the other, but the repeated information leads to multicollinearity. Therefore, one should transform event 0x1B into 0x1B-0x73; no information has been lost but the duplicated information has been removed. The *VIF (after trans.)* line shows how the VIF remains low (well below five) after making this transformation, indicating that the multicollinearity has been significantly reduced. In fact, after making this transformation, it would be possible to use up to ten PMC events while maintaining acceptable VIFs (of course, at run-time only seven events can be simultaneously used).

As the eleventh PMC event (instructions retired, 0x08) is added, the VIF increases dramatically once more. This is because 0x08 is also in cluster 13 (Figure 4.16) and is closely related to event 0x1B (Figure 4.15).

Reducing the multicollinearity in this second stage has made it possible to utilise information from all six PMCs (and the cycle counter), almost halving the average error from over 5% to less than 3%, without sacrificing stability. Being able to monitor only six PMC events simultaneously is not a significant limitation; as the number of events added to the model increases, the improvement in accuracy decreases (Figure 4.19). The presented generic methodology allows an appropriate number of PMC events for a particular CPU to be found by measuring the trade-off between VIF and average error.

Table 4.2 shows the final seven chosen events for the Cortex-A15, with their corresponding VIFs after the transformations, while Table 4.3 shows the final selected events for the Cortex-A7. In the case of the Cortex-A7, no transformations were required as the VIFs after Stage 1 were acceptable. Comparing the HCA-correlation analysis between the Cortex-A15 and Cortex-A7 (Figure 4.17) shows how the fundamental differences in the micro-architecture affect the PMC event selection decisions. For example, in the case of the Cortex-A7, the L1I cache accesses are closely related to

Table 4.2: First seven selected events for the Cortex-A15 after Stage 2 transformations

	Event Hex	Event Name	Cluster	VIF
1	0x11	CYCLE_COUNT	13	2.12
2	0x1B - 0x1B	INST-DP_SPEC	13	3.96
3	0x50	L2D_CACHE_LD	8	1.87
4	0x6A	UNALIGNED_LDST_SPEC	4	1.88
5	0x73	DP_SPEC	13	2.21
6	0x14	L1I_CACHE_ACCESS	14	2.23
7	0x19	BUS_ACCESS	9	1.50

Table 4.3: First five selected events for the Cortex-A7 with their corresponding cluster number and VIF from Stage 1

	Event Hex	Event Name	Cluster	VIF
1	0x11	CYCLE_COUNT	5	3.76
2	0x14	L1I_CACHE_ACCESS	6	2.16
3	0x16	L2D_CACHE_ACCESS	10	1.50
4	0x13	MEM_ACCESS	5	2.21
5	0x0F	UNALIGNED_LD_ST	14	1.03

the instructions executed (Figure 4.17), and therefore selecting instructions architecturally executed (0x08) in addition to this event only provides redundant data and causes multicollinearity problems. However, in the case of the Cortex-A15, the L1I cache accesses (0x14) are in a separate cluster to both the instructions architecturally executed and the instructions speculatively executed (0x1B, see Figure 4.16). Both 0x14 and 0x1B are selected because they add unique information regarding the power consumption and are not highly correlated. This is because of the more sophisticated fetch and prefetching mechanisms in the out-of-order Cortex-A15 and the large power impact of accessing the instruction cache. Note that unaligned loads and stores had a significant energy impact on both the in-order Cortex-A7 and the out-of-order Cortex-A15.

4.3.7 Demonstration of Model Stability

To demonstrate the importance of stability and carefully selecting PMC events, a model is built using a different set of PMC events that do not consider variance inflation (CYCLE_COUNT:0x11, L1I_TLB_REFILL:0x02, MEM_ACCESS:0x13, L1D_CACHE_ACCESS:0x04, INSTR_RETIRED:0x08, ASE_SPEC:0x74, VPF_SPEC:0x75). These choices reflect similar counters to those proposed in works on desktop and server systems and appear to be reasonable, intuitive choices [244, 243]. However, the average VIF of these events is 1.68×10^7 and the coefficients change significantly when building the model with different sets of workloads because they are unstable (note that the last two events do not contribute to the multicollinearity problems). Figure 4.20 compares a model built with these unstable PMC events and an otherwise identical model built using the results from the proposed PMC event selection method. The errors of these two models have been calculated over a variety of training and testing workload sets, all of which use k-fold cross-validation with $k = 10$ (the building and validation process is described in detail in Section 4.4).

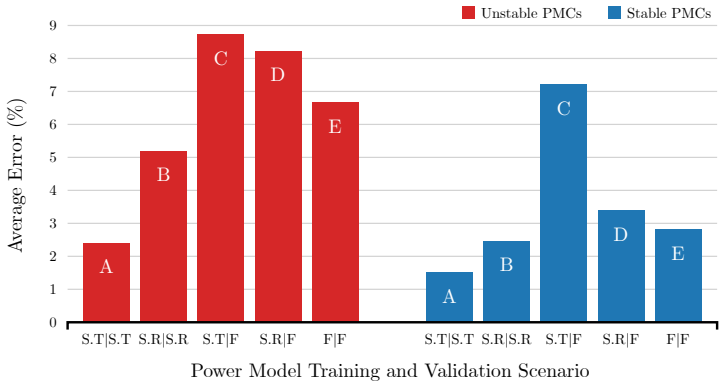


Figure 4.20: Comparison of a model with unstable PMC events and one with stable PMC events selected with the proposed methodology. Tested with different training and testing data sets. (S.T = small typical, S.R = small random, F = full set of 60 workloads)

With a limited set of 20 'typical' (MiBench and MediaBench) training and testing workloads (a typical scenario in related work [145, 243, 33]) both models obtain a small error (*A* bars). However, when 40 unseen workloads are added to the testing set, the error of both models increases significantly, from 1.5% to 7.2% in the case of the 'stable' selection (blue C bar). The small set of relatively similar workloads does not provide either model with enough information for predicting the full (and diverse) set of 60 workloads. This demonstrates how using a small, limited set of workloads for both training and validation results in a poor model which actually *appears* to be very accurate.

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

To summarise, there are three key points from Figure 4.20: 1) training and testing with a small set of workloads results in a poor model but with a very optimistic average error value; 2) training with a limited set of typical and similar workloads results in the lowest error when tested on the same set of workloads, but the highest error when tested on a large number of workloads (e.g. compared to a limited set of diverse workloads); 3) the proposed stable PMC selection is far superior at predicting a large number of diverse workloads outside of the training dataset and it makes efficient use of a limited, but diverse, training set. This benefit of stability comes from the fact that the model coefficients have lower errors and so the model better captures how each PMC event affects the overall power *individually*. A stable model is therefore able to make sensible estimation in unfamiliar situations and is less prone to wild predictions. For example, when both models are trained with 20 random (diverse) workloads and tested on 60, the blue model has a maximum error of less than 15%, while the red model has a maximum error of over 45%. Stability is a crucial quality of a run-time power model as it is impractical to represent the vast number of real-world applications in a training dataset.

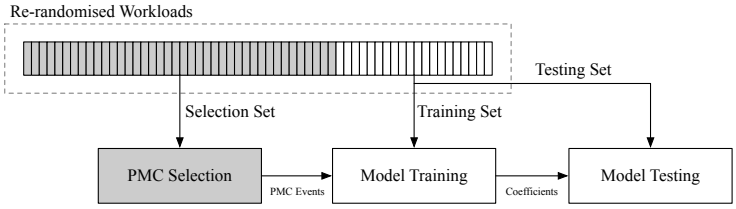


Figure 4.21: PMC event selection cross-validation procedure

4.3.8 Monte Carlo Validation

Monte Carlo Validation (also known as Repeated Random Sub-Sampling Validation) was used to test how well the PMC event selection methodology generalises to unseen observations. As data from all PMC events was required, the PMC selection dataset, which contains the PMC event data and power measurements from all 60 workloads run at a single DVFS level of 1 GHz, was used (60 observations in total). The dataset was shuffled and the first 50 observations were provided to the PMC event selection methodology (training), while the remaining 10 observations were used for building and validating the power model using the selected events (testing). The process for building the power model differs from the method described later in this chapter in that only one DVFS level is considered. This cross-validation method (Figure 4.21) was repeated 100 times and the arithmetic mean of the results calculated.

The arithmetic mean of the Mean Absolute Error (MAE) of each iteration was 0.0050 W and the standard deviation was 0.0023 W. The mean of the R^2 value was 0.998 and the standard deviation was 0.0057. The mean MAPE was 1.1%, while the median MAPE was 1.0%. Note that the MAPE is lower than the final MAPE of the model due to the fact that only one DVFS level is modelled, and that the validation technique of the coefficient values (which is not being considered in this analysis) is not as rigorous as the one used to validate the overall model.

In Chapter 6, the PMC events chosen in this chapter are used to build and validate a model where over 50% of the workloads are new, and including workloads from the PARSEC and ParMiBench benchmarking suites. A MAPE of 2.8% was achieved (the MAPE achieved in this chapter is also 2.8%), demonstrating that the chosen PMC events apply well to other workloads not considered in the PMC selection training.

4.4 Model Formulation and Validation

This section describes the model building and validating stage which is the third step of the power modelling methodology (Figure 4.1). Once the optimal events were obtained from Step 2 (Section 4.3), the experimental setup (Section 4.2) was used to run the full set of 60 workloads at many DVFS levels, with different numbers of cores being utilised, to extensively investigate as many operating conditions as possible, on both the Arm Cortex-A7 and Cortex-A15. The out-of-order Arm Cortex-A15 is used as an example to demonstrate the methodology and analysis in this section but the final results for the Arm Cortex-A7 are also reported.

4.4.1 Model Specification

The approach presented in Chapter 3, as well as many of the existing works (including [26, 244, 238, 33]), train a separate power model for each DVFS level. In this chapter, a single model is developed that works across all DVFS levels by understanding the effects of voltage and frequency, and accounts for certain thermal effects. This deeper understanding of how power in the CPU is consumed allows the power consumption to be decomposed into contributing parts and is useful for system optimisation and design-space exploration, as well as run-time power management (see Chapter 6).

An important assumption in regression analysis is correct specification of the model (the assumptions are summarised in Section 3.1.6). Therefore the equation was carefully developed considering the various components of power consumption described in Section 2.1. Many existing works insert features into OLS solvers without considering the specific relationship between them and the power consumption (examples include [243, 284, 214]). To illustrate this, the following equation has been adapted from [214]:

$$\begin{aligned} P = \beta_0 + \beta_1 V_{DD} + \beta_2 f_{clk} + \beta_3 T + \beta_4 \text{IPC} + \beta_5 \frac{\text{INT}}{N} + \beta_6 \frac{\text{VPF}}{N} + \ldots + \\ \beta_9 \frac{(\text{L2 REFILL})}{N} + \beta_{10}(\text{CPU User}) + \ldots + \beta_{15}(\text{CPU SOFT IRQ}) \end{aligned} \tag{4.17}$$

There are several serious problems with this model equation. Primarily, the relationship between the power consumption and the activity (indicated by the PMC events), frequency, voltage, and temperature has not been specified correctly. From Equation 2.2 it was seen that the dynamic power consumption is proportional to $\alpha C V_{DD}^2 f_{clk}$. However, Equation 4.17 simply inserts a separate V_{DD} term and a separate f_{clk} term. In the same fashion, a single temperature, T, term has been inserted into the model in the hope that the model equation would then capture the effect of temperature on power consumption without considering the physical relationships between the temperature, the other independent variables and the power consumption. The PMC events are inserted into the model independently, despite their individual effect on power to be dependent on the current voltage and the frequency. An OLS estimator is only able to estimate the equation coefficient estimates, $\hat{\beta}$, and is not able to automatically re-formulate the equation, combine related terms together, or to insert polynomial values in order to create an equation that models the complex physical relationships. Clearly, whatever the values of the coefficients of $\hat{\beta}$, Equation 4.17 cannot accurately estimate power across multiple DVFS levels. Furthermore, the information from the coefficients $10 \ldots 15$ is operating system data (found in the `/proc/stat` virtual file in Linux) that provides high-level information about how CPU time has been spent and does not provide information on CPU activity that is not already present in the PMC events used in the model, causing multicollinearity. This misspecification leads to the erroneous conclusion that *full-frequency* models (i.e. a single equation and set of coefficients that works at every DVFS level) cannot achieve low errors (the minimum full-frequency model error achieved by the authors for the Arm Cortex-A15 CPU was 19.95%, which is significantly higher than the error achieved in this chapter for the same platform) and that having multiple models developed at individual DVFS levels is more appropriate.

In another example, the equation below is adapted from a power model of an Arm Cortex-A8 [243]:

$$P = \beta_0 + \beta_1 f_{cpu} + \beta_2 \text{IPC} + \beta_3(\gamma_1 + \gamma_2) \tag{4.18}$$

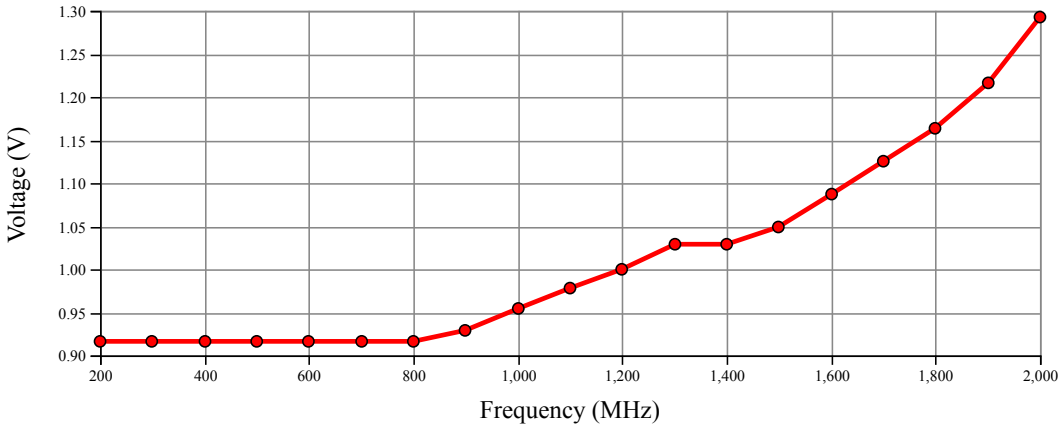


Figure 4.22: Voltage (V) at each DVFS level

where γ_1 and γ_2 are the L1 and L2 cache miss rates (it is not specified whether γ_1 includes the instruction cache and/or the data cache). The f_{cpu} has been added as an independent term, despite the power impact of the events (IPC, γ_1, γ_2) being dependent on it. Furthermore, the quadratic relationship with voltage has not been captured.

A common assumption is that f_{clk} can be used as a substitute for V_{DD} in the equations (e.g. [243, 284]). However, as shown in Section 2.2.1, optimal f_{clk} is not directly proportional to V_{DD}. For the development board considered in this chapter, the first seven DVFS levels (200 to 800 MHz, inclusively) share the same voltage, while the increase in voltage thereafter is not directly proportional to the operating frequency (Figure 4.22). Pathania *et al.* reported that only three voltages were used for nine DVFS level on the mobile platform considered in their work on power and performance modelling of mobile gaming workloads [225]. Because the dynamic power consumption has a quadratic dependence on voltage, errors in the voltage assumptions significantly impact the accuracy of the model.

The model equation presented in this chapter is based on the CPU power consumption equations and theory presented in Sections 2.1 and 2.2. Statistical significance tests and analysis of the residuals are also used to ensure correct model specification, resulting in a physically-meaningful equation that calculates static and dynamic power separately and allows the power to be further decomposed into its constituting parts (Equation 4.20). The CPU cluster power, $P_{cluster}$, was made the dependent variable and includes all four cores of the cluster, the L1 caches, TLB hierarchy, and the shared L2 cache. The M PMC event rates, $E_{0\ldots M-1}$ were used to estimate the dynamic power consumption (as explained in Section 4.2, the PMC event rates were divided by the frequency, f_{clk} in pre-processing and so $\boldsymbol{E}$ contains no frequency information). Each PMC event rate, E_m is multiplied by $V_{DD}^2 f_{clk}$. Multivariate linear regression was then performed to estimate $\hat{\boldsymbol{\beta}}$ (using an OLS estimator, see Section 3.1).

The static power consumption is proportional to the voltage, V_{DD}, and the leakage current, I_{leak}. In an overly-simplified equation, the power could be calculated using:

$$P_{cluster} = \underbrace{\left(\sum_{m=0}^{M-1} \beta_m E_m V_{DD}^2 f_{clk}\right)}_{\text{dynamic activity}} + \underbrace{\beta_a V_{DD} + \beta_b}_{\text{static}} \tag{4.19}$$

This equation assumes that all of the dynamic power is indicated by the PMC events and that the static power consumption is only dependent on V_{DD}. However, these assumptions are false and a model using this equation performs poorly.

Other component variables were added to the model and the residuals (discussed later in Section 4.4.2) and statistical significance tests were used to evaluate the impact on the model. This approach identified that a constant (at a particular DVFS level) dynamic component was present. This meant that the model required terms related to f_{clk} that were not related to PMC event inputs. This constant background (B.G.) dynamic power component is labelled as *BG dynamic*.

Section 2.1 described how the sub-threshold and band-to-band tunnelling leakage current components (I_{sub} and I_{btbt}, respectively) are temperature dependent. The models presented in this chapter assume that no thermal sensors are available to the run-time power model but must still compensate for the effects of temperature (the CPU temperature is recorded in the experimental data). Assuming a stable ambient thermal environment, V_{DD}, f_{clk} and the dynamic activity have a significant influence on the CPU temperature, T. Therefore, the static power equation must include terms related to V_{DD} and f_{clk} in order to *absorb* the effects of temperature on the static power consumption, P_{static}. The equations in Section 2.1, significance tests, and residual plots were used to identify suitable components, taking care to not over-fit the model. Computationally simpler components were selected (e.g. V^3 instead of $\log(V)$) if there was little benefit in using the more computationally complex ones. Chapter 5 contains a more detailed analysis of the thermal compensation, including an evaluation of the model error without thermal compensation, with thermal compensation, and using the measured temperature.

With the B.G. dynamic power component and the terms to absorb the effect of temperature, the model becomes:

$$P_{cluster} = \underbrace{\left(\sum_{m=0}^{M-1} \beta_m E_m V_{DD}^2 f_{clk}\right)}_{\text{dynamic activity}} + \underbrace{f(V_{DD}, f_{clk})}_{\text{static and B.G. dynamic}} \tag{4.20}$$

where $f(V_{DD}, f_{clk})$ contains functions of both V_{DD} and f_{clk}. The specific independent variables (and their estimated coefficients) for the Arm Cortex-A15 cluster are presented in Table 4.4.

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

The model equation splits the dynamic activity from the rest of the components. Therefore, if the model was correctly specified, then it would be possible to train the model by running all of the workloads at just a single DVFS level and then provide single observations (e.g. idle) at the other DVFS levels. Furthermore the enhanced model stability allows the model to be trained with fewer workloads, provided the training workloads are diverse. These two qualities were combined and the model was trained using 50% of the training workloads and running these workloads at 1000 GHz only and providing single idle (sleep) observations at the other DVFS levels. Experiment time

Table 4.4: Cortex-A15 CPU model coefficients, $d = V^2 f$

Component Type	Term	Coeff.	Low. C.I.	Upp. C.I.	p-Value
Static and B.G. Dynamic	Intercept	-7.53e+2	-8.86e+2	-6.19e+2	$p < 0.0001$
Dynamic Activity	0x11×d	5.72e-10	5.55e-10	5.90e-10	$p < 0.0001$
Dynamic Activity	(0x1b-0x73)×d	7.30e-10	6.94e-10	7.66e-10	$p < 0.0001$
Dynamic Activity	0x50×d	8.12e-9	7.40e-9	8.84e-9	$p < 0.0001$
Dynamic Activity	0x61×d	1.61e-8	1.46e-8	1.75e-8	$p < 0.0001$
Dynamic Activity	0x73×d	8.57e-11	6.27e-11	1.09e-10	$p < 0.0001$
Dynamic Activity	0x14×d	1.09e-9	9.97e-10	1.17e-9	$p < 0.0001$
Dynamic Activity	0x19×d	2.50e-9	2.22e-9	2.79e-9	$p < 0.0001$
Static and B.G. Dynamic	f	1.52e-1	1.16e-1	1.87e-1	$p < 0.0001$
Static and B.G. Dynamic	V	2.50e+3	2.07e+3	2.94e+3	$p < 0.0001$
Static and B.G. Dynamic	Vf	-6.03e-1	-7.27e-1	-4.78e-1	$p < 0.0001$
Static and B.G. Dynamic	V^2	-2.77e+3	-3.25e+3	-2.30e+3	$p < 0.0001$
Static and B.G. Dynamic	V^3	1.02e+3	8.47e+2	1.12e+3	$p < 0.0001$
Static and B.G. Dynamic	fV^3	-3.14e-1	-3.71e-1	-2.57e-1	$p < 0.0001$

Table 4.5: Model formulation experiment speedup when exploiting smart model formulation and enhanced stability

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

(time taken to run and collect the training observations) was reduced by 96× while the MAPE only increased by 0.6% (Table 4.5). It is believed that this small error increase could be further reduced by using a controlled thermal environment, as the idle observations are largely made up of static power consumption and are therefore particularly sensitive to temperature changes. Single active workload observations could also be used to help minimise the error difference further. Not only does this demonstrate the robustness of the model formulation and the advantage of stability, but the reduced experiment time (and reduced amount of resulting data) has an important practical benefit.

4.4.2 Analysis of Residuals

A fundamental and necessary part of building linear regression models is the inspection of the residuals (residuals are introduced in Section 3.1.2), which must be done in order to determine whether the model is valid and the assumptions for the linear regression model (see Section 3.1.6) have been met [201]. Yet, despite this, very few related works (the only known exceptions are [71], [314] and [284]) present or discuss the residuals, which need to be checked before other model statistics can be trusted and interpreted.

The mean of the residuals should be zero:

$$\bar{e} = \frac{\sum_{i=1}^{n} e_i}{n} = 0 \tag{4.21}$$

Furthermore, the variance of the residuals is defined as follows [162]:

$$s^2 = \frac{\sum_{i=1}^{n}(e_i - \bar{e})^2}{n-2} = \frac{\sum_{i=1}^{n} e_i^2}{n-2} = \frac{RSS}{n-2} = MSE \tag{4.22}$$

(Note that the residual sum of squares, RSS, is sometimes known denoted SSE or SSR, as described in Section 3.1.4).

Identifying Outliers and Influence

A fundamental step in regression analysis is to determine if the model is heavily influenced by one or more observations in the training set. While identifying outliers is trivial when two independent variables are used, it becomes challenging with more independent variables as univariate outliers may not be extreme in a multivariate regression model, and some multivariate outliers may not be detectable on univariate analysis [162].

The residuals can be used to identify outliers. A typical way to standardise residuals for outlier detection is to calculate the *semi-studentised* residuals:

$$e_i^* = \frac{e_i - \bar{e}}{\sqrt{MSE}} = \frac{e_i}{\sqrt{MSE}} \tag{4.23}$$

However, the variance of the residuals typically varies significantly even when the original observations all have the same variance, σ^2, because the precision of the fitted values depends on the pattern of covariate values [245]. The above equation is only considered *semi*-studentised because $\sqrt{MSE}$ is not an estimate of the standard deviation of the residual, e_i.

It is therefore useful to *studentise* each residual by dividing it by an estimate of its standard deviation. In order to calculate the studentised residuals, $\boldsymbol{r}$, the projection matrix, $\boldsymbol{H}$, (also known as the hat matrix, defined in Section 3.1.3) is used.

If the model is appropriate, the MSE is an unbiased estimator of the variance of the error terms, σ^2 (see equation 3.31 on 40).

Furthermore, it was shown in Section 3.1.4 that:

$$var(\boldsymbol{e}) = \sigma^2(\boldsymbol{I} - \boldsymbol{H}) \tag{4.24}$$

Therefore:

$$var(e_i) = \sigma^2(1 - h_{ii}) \tag{4.25}$$

where h_{ii} is the i^{th} diagonal element of the projection matrix.

The studentised residuals (also known as the internally studentised residuals), are therefore given by [162, 245]:

$$r_i^* = \frac{e_i}{\hat{\sigma}\sqrt{1 - h_{ii}}} \tag{4.26}$$

and consider the magnitude of each residual, relative to its standard deviation. This assumes that the assumption of homoscedasticity is respected (see Section 4.4.2).

A refinement to the studentised residuals, $\boldsymbol{r}$, is the studentised *deleted* residuals, $\boldsymbol{t}$ (also known as the externally studentised residuals). However, the (internal) studentised residuals were adequate for this purpose.

A common rule of thumb is that studentised residuals with absolute values over two or three are considered outliers and have significant influence on the regression fit.

The workloads *cstm_bmp* and *openmp_mem_spd* result in the largest positive residuals and *cstm_int*, *cstm_fp* and *cache* (from LMbench) had the largest negative residuals. Furthermore, the studentised residuals tended to be larger at higher frequencies and when larger numbers of cores were utilised (which is addressed in Section 4.4.2).

While the outliers are unusual dependent variable values, the *leverage* measures unusual influence of predictor values. The diagonal elements, h_{ii}, of the projection matrix, $\boldsymbol{H}$, are the leverage values. All values of h_{ii} have values between 0 and 1, where a value close to zero indicates low influence and a value close to one indicates high influence. As a rule of thumb, if the leverage value, h_{ii}, is more than twice as large as the mean leverage value, $\bar{h}$, it is considered large. The mean leverage value is given by [161]:

$$\bar{h} = \frac{\sum_{i=1}^{n} h_{ii}}{n} = \frac{p}{n} \tag{4.27}$$

Another rule of thumb suggests that a h_{ii} between 0.2 and 0.5 indicates moderate leverage while a value greater than 0.5 indicates very high leverage.

After detecting outliers, the next step was to understand whether these outliers are influential. Cook's distance was used to understand how far the predicted values would move if the model were to be fit without the detected outliers. Cook's distance, D_i, for the effect of the i^{th} case on all fitted values, n, is given by [161]:

$$D_i = \frac{\sum_{j=1}^{n} (\hat{y}_j - \hat{y}_{j(i)})^2}{ps^2} \tag{4.28}$$

where $y_{j(i)}$ is the fitted dependent variable when excluding i and s^2 is the mean squared error (MSE) of the regression model. Alternatively, in matrix form:

$$D_i = \frac{(\hat{\boldsymbol{y}} - \hat{\boldsymbol{y}}_{(i)})^T (\hat{\boldsymbol{y}} - \hat{\boldsymbol{y}}_{(i)})}{ps^2} \tag{4.29}$$

In order to calculate the Cook's distance without having to fit the model for each i, the following expression was used:

$$D_i = \frac{e_i^2}{ps^2} \left[\frac{h_{ii}}{(1 - h_{ii}^2)} \right] \tag{4.30}$$

Observations with the largest studentised residuals, leverage and Cook's distance were at the highest DVFS level (again, this is addressed in Section 4.4.2). The workloads with the largest influence on the fit were (Figure 4.23):

- 1403: cstm_bmp
- 1404: cstm_int
- 1402: neon_mul
- 1382: bitcount

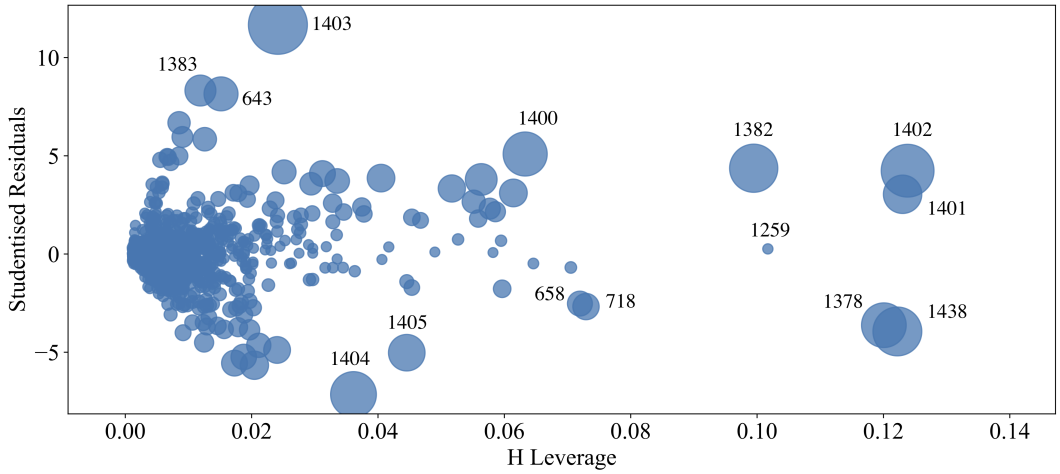


Figure 4.23: Influence plot. Point diameter is proportional to Cook's distance

- 1438: openmp_mem_spd

Clearly, there are observations that are influential due to their leverage (e.g. 1402, 1382) and others due to large studentised residuals (e.g. 1403). Some observations have high leverage but low influence (e.g. 1259, openmp_mflops). Note that the leverage values are all below the value generally considered to indicate moderate leverage (0.2). However, many of the studentised residuals are greater than three (indicating outliers).

Analysis of outlying and influential observations is a crucial step in regression analysis that is overlooked in the existing power modelling works. However, this step is not automatic and requires intuition.

Removing the five workloads detected as being influential increases the R^2 value from 0.997to 0.998 and reduces the cross-fold validated MAPE from 2.8% to 2.4%. However, there is no objective reason for removing these workloads from the training set and they are not caused by measurement error. Three of the five workloads were created specifically to exercise the CPU in a diverse fashion for improving coefficient stability. Reducing the outliers and their influence in this situation would require collecting more PMC events to better capture the complex workload behaviour's influence on power consumption.

In other words, analysis of studentised residuals, leverage and Cook's distance has identified workloads that are not well captured by the selected PMC events.

Normality of Error Terms

In OLS it is typically assumed that the errors have a Gaussian (normal) distribution (Section 3.1.6). A histogram plot of the (internal) studentised residuals was plotted with a Gaussian distribution line superimposed using the mean and variance of the residuals (Figure 4.24). It can be observed that the histogram of the residuals does not match this Gaussian curve; there is a higher proportion of smaller residuals than expected if the distribution were Gaussian.

A normal probability, also known as a normal quantile-quantile (Q-Q) plot, was created by plotting each studentised residual against its expected value according to the normal distribution (Fig-

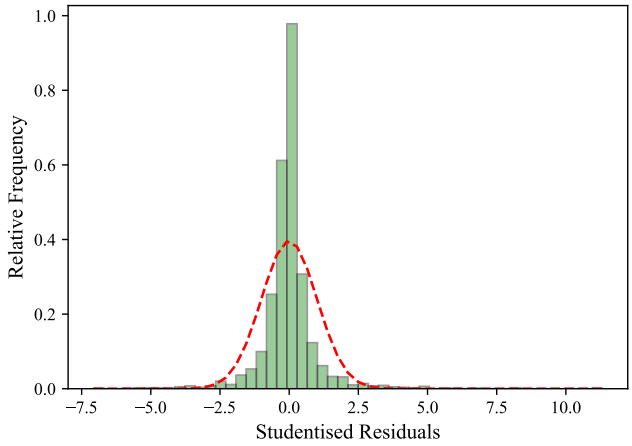


Figure 4.24: A histogram plot of the (internal) studentised residuals with a modelled Gaussian distribution superimposed

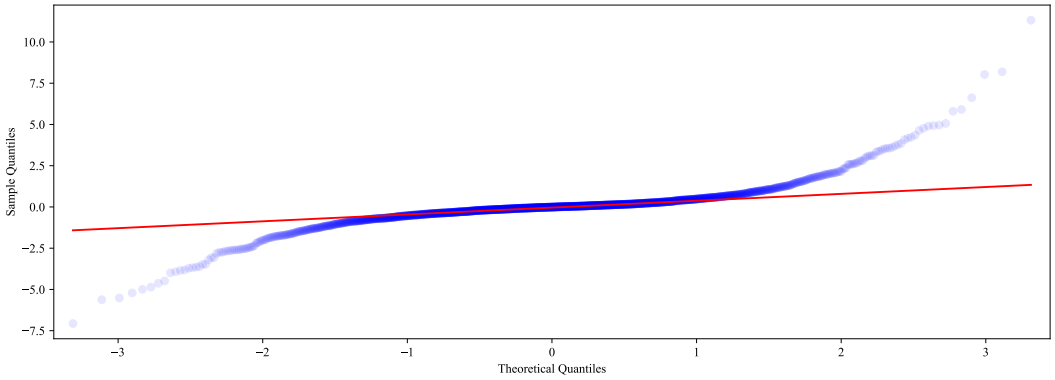


Figure 4.25: Normal probability plot of the studentised residuals

ure 4.25). The two sets of quantiles were plotted against each other and so a straight line plot would suggest an agreement with normality (though the ends of the line deviating from the line slightly is typical) [161, 101].

Both the histogram and the inverted s-shape of the normal probability plot show that the distribution is *over-dispersed* compared to the normal distribution, having heavy tails (positive excess kurtosis). This is also known as a leptokurtic distribution.

The assumption of normal distribution of the residuals is not necessary if the model equation is correct and its only use is to estimate its coefficients and generate predictions, as is the case for this model. However, the calculation of the confidence intervals and the significance tests assume normally distributed errors [210]. Furthermore, the residuals can appear to not be normally distributed because of a misspecified model or non-constant error variance (heteroscedasticity, identified and addressed later in this chapter).

Checking the Residual Plots

The model:

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon} \tag{4.31}$$

can be written as:

$$\text{response} = \text{deterministic} + \text{stochastic} \tag{4.32}$$

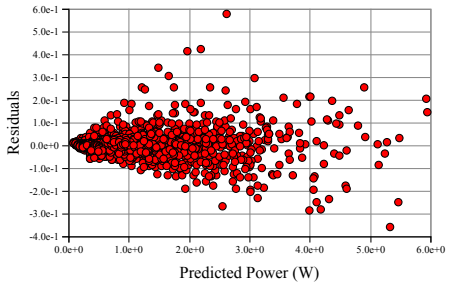
where the deterministic part is explained by the independent variables in the model and the stochastic part is unpredictable.

If the model was correctly specified and the independent variables contained all the required information, then the error would only be made up of inherent randomness of a real-world phenomenon [197]. As only six PMC events and the cycle count can be monitored simultaneously, there will be error due to missing information regarding the CPU operation. Furthermore, it is not possible to capture the full complexity of the thermal behaviour (particularly as ambient temperature is not known to the model).

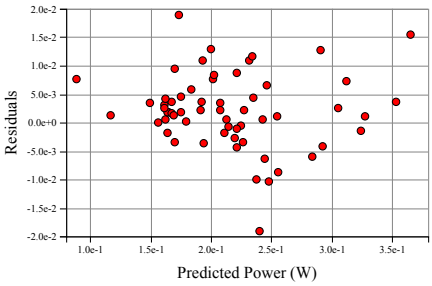
Plots of the residuals were used to check that the error is stochastic. A plot of the residuals against the fitted values (predicted power) showed no clear patterns (Figure 4.26a). Note that the non-constant variance is addressed in the next section. The residuals were plotted against each feature (Figures 4.27 and 4.28). No patterns exist in any feature, meaning that the model captures the necessary information from them. The right-side plots show the PMC event rates (first divided by the f) multiplied by $V^2 f$ (which is how they appear as inputs to the model). There are clearly some sub-paths in the residuals indicating that the model would likely benefit from other related features. For example, Figure 4.27h, which shows event `0x6A` (Unaligned load stores speculatively executed), has two clear paths. Event `0x6A` is actually counting both unaligned loads (`0x68`) and unaligned stores (`0x69`) and so the model would likely benefit from having both of these events present instead of event `0x6A`, if more events could be monitored simultaneously.

It was also important to observe the residuals for variables not included as input features. For example, while the effects of temperature due to the voltage, frequency and activity were absorbed in the model (see Section 4.4.1), the measured temperature (which is also influenced by ambient temperature conditions) is not an input feature (the model presented in Section 5 does include temperature sensor data as input features). There is not a strong, clear relationship between the measured temperature and the residuals (Figure 4.29d) showing that the model compensates for temperature effectively. However, there is a small deviation (particularly at lower temperatures) which is likely due to the fact that ambient temperature changes are not measured (the ambient thermal conditions were not stable).

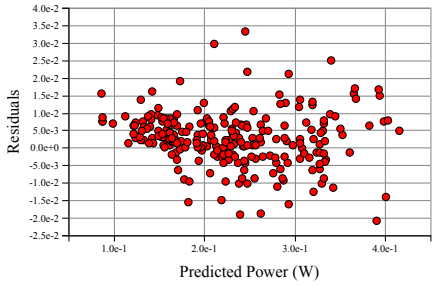
An interesting observation from the voltage residual plot (Figure 4.29b) is that the different voltage levels are not in one clear vertical line, but are in wider clumps (unlike the frequency residual plot in Figure 4.29a). This shows that the voltage levels are not stable; they vary between observations at the same DVFS level. This is addressed in Section 4.6.



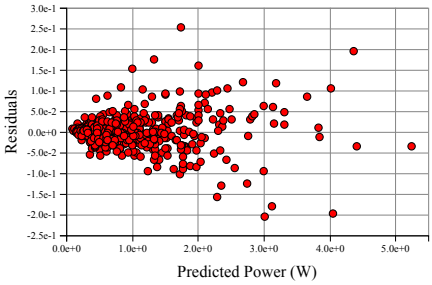
(a) Fitted values (all)



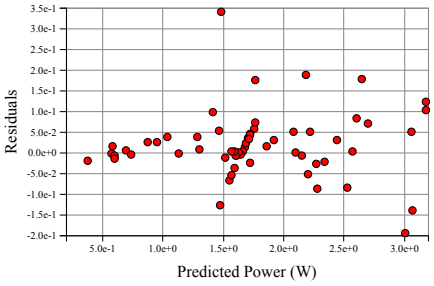
(b) Fitted values (200 MHz, 2 cores)



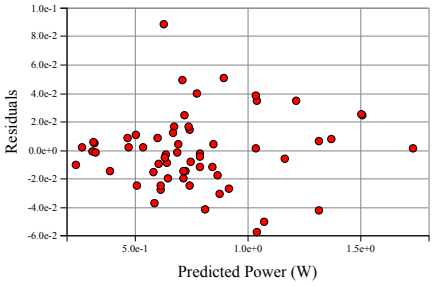
(c) Fitted values (200 MHz, all cores)



(d) Fitted values (all frequencies, 2 cores)



(e) Fitted values (1400 MHz, 4 cores)



(f) Fitted values (1000 MHz, 2 cores)

Figure 4.26: Residuals plotted against the fitted values

Homoscedasticity

The cone shaped residual plots (Figure 4.26a) indicate that the assumption of homoscedasticity (constant variance, see Section 3.1.6) has been violated. The problem of non-constant variance is known as heteroscedasticity.

The variance increases with DVFS levels and the number of active cores (Figure 4.29). Heteroscedasticity can be observed in the residuals at a single frequency and with multiple cores active, and vice-versa (Figure 4.26). When viewing fitted residuals at one specific DVFS level with a specific number of cores active, the residuals can appear to be homoscedastic. However, when viewing many cases it appears that there is some heteroscedasticity even at a fixed DVFS level with a fixed number of cores utilised (Figures 4.26).

The problem of heteroscedasticity is therefore inherent in PMC-based power modelling (there is a larger variance in power consumption between high power-consuming workloads than workloads that consume less power), yet has not been highlighted or addressed in previous work on PMC-based CPU power modelling. Heteroscedasticity is inherent in many problems. A common example is estimating the expenditure on food based on household income. A household with a low income

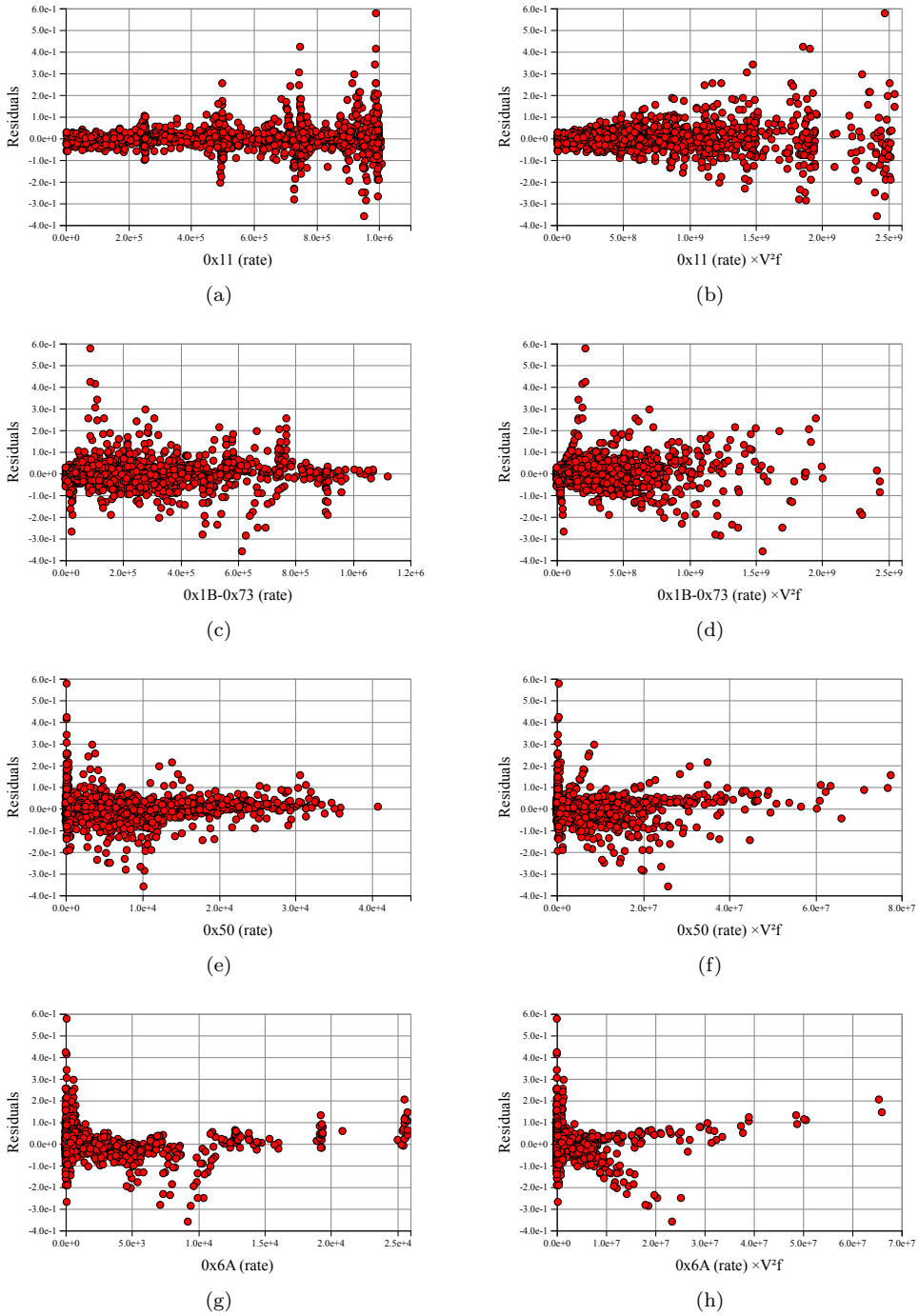


Figure 4.27: Residuals plotted against PMC event rates

will consistently spend a small amount on food. However, a household with a large income will sometimes spend a large amount on food (e.g. expensive restaurants), but will sometimes have inexpensive meals (such as simple home-cooked food or fast-food restaurants).

In the presence of heteroscedasticity, the OLS estimates of the coefficients are still unbiased and consistent, but they are no longer minimum variance estimators. The Gauss-Markov theorem no longer applies and the OLS estimators are no longer the Best Linear Unbiased Estimators (BLUE).

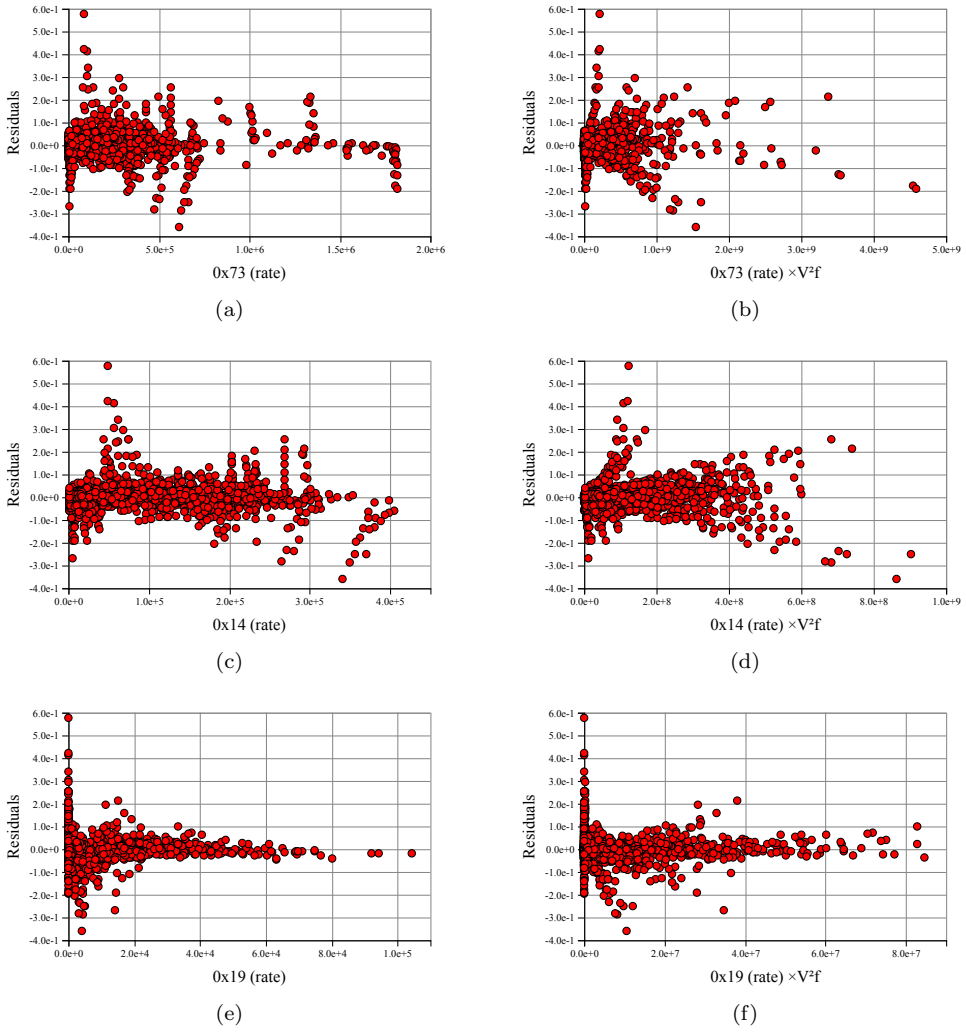


Figure 4.28: Residuals plotted against PMC event rates

Conventionally, the covariance matrix is [178, 330, 259, 208]:

$$\sigma^2\{\hat{\beta}\} = \text{var}(\hat{\beta}) = \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1} \tag{4.33}$$

And is typically estimated using:

$$\text{var}(\hat{\beta}) = \frac{\sum e_i^2}{n-p}(\boldsymbol{X}^T\boldsymbol{X})^{-1} \tag{4.34}$$

The estimates of the standard errors in the presence of heteroscedasticity are, however, biased meaning that values calculated using them are unreliable. For example the significance tests and confidence intervals may be too liberal or too conservative [317, 88].

This problem was addressed by using a heteroscedasticity-consistent standard error (HCSE) estimator of OLS parameter estimates (also known as using a heteroscedasticity consistent covariance matrix [HCCM] or robust standard errors). The regression model itself was still estimated using OLS (the model was still unbiased and consistent but was no longer efficient, however, the accuracy

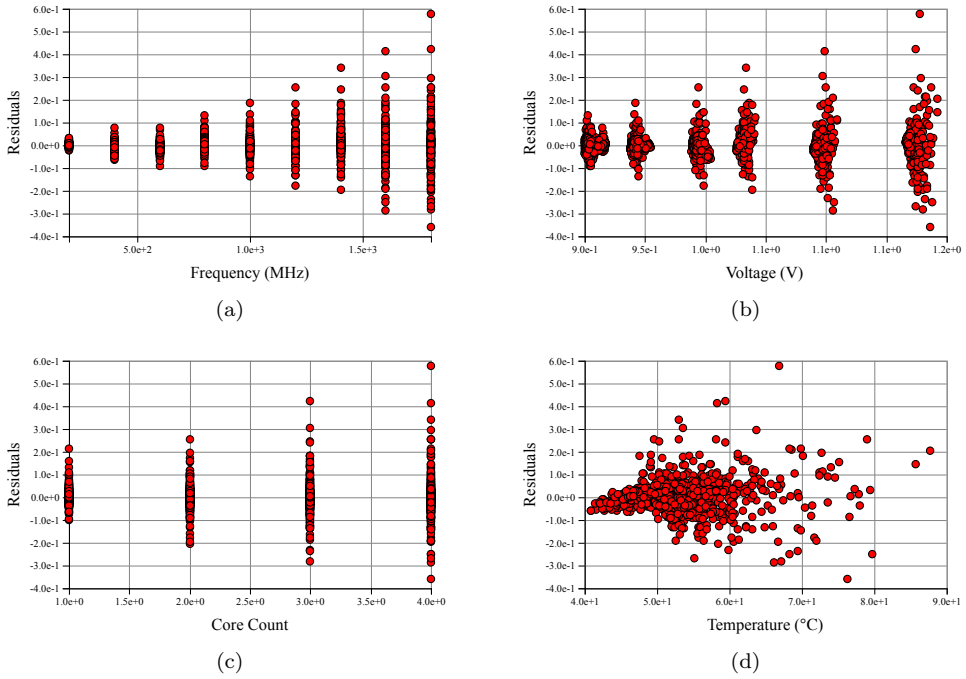


Figure 4.29: Residuals plotted against the frequency, voltage, number of active cores and the temperature

of the model is demonstrated in Section 4.4.5). However, using a HCSE meant that statistically consistent estimates of the variances and covariances of the OLS estimates could be calculated without the assumption of homoscedasticity [113]. The advantage of using HCSE estimators is that it does not require detailed knowledge of the heteroscedasticity (unlike Weighted Linear Regression [WLS]), an arbitrary transformation of $\boldsymbol{Y}$ or bootstrapping. The HC3 estimator was used [178, 183]:

$$\text{HC3} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T \text{diag}\left[\frac{e_i^2}{(1-h_{ii})^2}\right]\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1} \tag{4.35}$$

Rayes and Cai [122] give a good introduction and highlight the problem of HCSE estimators being largely unknown outside of the fields of statistics and econometrics and how they can eliminate the need for researchers to worry about heteroscedasticity when using OLS regression. Furthermore, it has been previously noted that heteroscedasticity "has never been a good reason to throw out an otherwise good model" [186]. All reported results, including those earlier in this chapter, were calculated using a HCSE estimator.

4.4.3 Model Evaluation

Both the Cortex-A7 and Cortex-A15 models achieve an R^2 value of greater than 0.99 (Table 4.6), showing that the fitted regression explains over 99% of the variation in the power consumption, despite being tested on a large number of diverse workloads. The adjusted R^2 (which compensates for the number of predictors) is less than 0.00006 lower than the R^2 value, showing that the R^2 value is not inflated by the number of predictors (independent variables) and that each predictor adds valuable information to the model.

Table 4.6: Results for the Cortex-A7 and Cortex-A15 CPU models (final model, trained on all workloads)

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

In order to conduct hypothesis tests on the estimated coefficients, $\hat{\beta}$, t-tests were used. The corresponding p-values were then reported (Table 4.4).

The average VIF of the input PMC events of both models is under 2.3, which is due to the PMC event selection method (Step 2, described in Section 4.3), and the average VIF of all of the model inputs (including the voltage and frequency) is less than five, due to the model formulation (Step 3, described earlier in this section). These low VIF values indicate that the model has good coefficient stability, which was also demonstrated in Section 4.3. Statistics for determining the statistical significance are presented in Table 4.6. The p-Value for F-Statistic row shows the p-value when the model is compared to a model with no predictors. The p-values are very low (under 0.00001 for both models); meaning that if the null hypothesis is true (there is no effect or relationship between the model inputs and power), the observed effect would be found in less than 0.001% of the experiments run due to random sampling error. In many scientific studies, $p < 0.05$ is considered borderline statistically significant and $p < 0.001$ is considered borderline statistically highly significant.

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

$$P.I.(95\%) = \pm(2 \times \text{SER}) \tag{4.36}$$

The SER (standard error of regression) gives the average distance between the observed values and the regression line (Table 4.6). Therefore, the approximated 95% prediction interval for the Cortex-A15 model is ±0.10 W; one can be 95% confident that the actual power is within the predicted power ±0.10 W for the next observation.

Table 4.7: Cortex-A7 Model Coefficients

Component Type	Term	Coeff.	Lower C.I.	Upper C.I.	p-Value
Static and B.G. Dynamic	Intercept	4.838e+1	3.363e+1	6.314e+1	$p < 0.0001$
Dynamic Activity	0x11×V^2f	7.640e-11	6.876e-11	8.405e-11	$p < 0.0001$
Dynamic Activity	0x14×V^2f	3.776e-10	3.610e-10	3.941e-10	$p < 0.0001$
Dynamic Activity	0x16×V^2f	1.889e-9	1.705e-9	2.073e-9	$p < 0.0001$
Dynamic Activity	0x13×V^2f	2.318e-10	2.023e-10	2.613e-10	$p < 0.0001$
Dynamic Activity	0x0f×V^2f	2.955e-9	2.777e-9	3.132e-9	$p < 0.0001$
Static and B.G. Dynamic	f	-2.081e-2	-2.848e-2	-1.314e-2	$p < 0.0001$
Static and B.G. Dynamic	V	-1.497e+2	-1.961e+2	-1.033e+2	$p < 0.0001$
Static and B.G. Dynamic	Vf	6.745e-2	4.594e-2	8.896e-2	$p < 0.0001$
Static and B.G. Dynamic	V^2f	-7.276e-2	-9.388e-2	-5.164e-2	$p < 0.0001$
Static and B.G. Dynamic	V^2	1.544e+2	1.061e+2	2.027e+2	$p < 0.0001$
Static and B.G. Dynamic	V^3	-5.306e+1	-6.965e+1	-3.646e+1	$p < 0.0001$
Static and B.G. Dynamic	fV^3	2.617e-2	1.894e-2	3.340e-2	$p < 0.0001$

Table 4.8: Model results from k-fold cross-validation

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

4.4.4 Model Coefficients

The coefficients for the Cortex-A15 model that has been developing throughout this chapter are reported in Table 4.4, while the coefficients for the Cortex-A7 model are reported in Table 4.7. The coefficients for the transformed PMC events are all positive and the voltage and frequency both have a positive influence on the power, as expected. The 95% confidence intervals (C.Is) of the coefficients are also reported, which take into account the sample size and the variance in the population. A narrow confidence interval indicates a low sampling error. The confidence intervals are very small, showing very low standard error and very high statistical significance for each coefficient. The p-values for every coefficient are very low, far lower than 0.05, confirming the statistical significance of every model coefficient.

4.4.5 Cross-Fold Validation

The results in Table 4.6 are derived from all of the observations used to build the model. K-fold cross validation was also used (see Section 3.9), which involved randomising the order of the observations, splitting the observations into k groups, then using $k-1$ of the groups to build the model (training dataset) and the one remaining group to validate the model (testing dataset). This process was repeated so that the model was built k times, with each group of observations being

used to validate the model. The reported cross-validated errors (Table 4.8) are the average of the testing datasets, so the model was always predicting the power for scenarios it had not seen before. The validated average error (mean absolute percentage error [MAPE]) was 2.8% for the Cortex-A15 model and standard deviation of each k-fold group error (%) was 0.022%, with the maximum error of 2.824%. The cross-validated error was 3.8% for the Cortex-A7 model. The Cortex-A7 model has a larger error, despite having a simpler micro-architecture, because the its power consumption is significantly lower. The lower absolute average power means that the percentage error is higher and the measured power is closer to the resolution of the built-in sensors. The SER of the Cortex-A7 (in Watts) is over three times lower than that of the Cortex-A15.

4.4.6 Model Decomposition

By looking at the cross-validated average errors for each of the 60 workloads (Figure 4.30), it can be seen that *cstm_bmp* (a custom synthetic workload) is the only workload with an average error over 6.5%. This is not due to measurement error or experimental inconsistency, but due to the model not accurately estimating this particular workload. Furthermore, the same workload exhibits a large error at all DVFS levels. This workload is a hand-written synthetic program, specifically created to exercise the CPU in a unique way. The model features do not contain enough information for accurately predicting this workload and its uniqueness means that the coefficients are not optimal for predicting this workload, with other workloads having a larger influence on the model fit. Section 4.4.2 discussed this outlier and the model influence further. The fact that there is one outlier shows the need to use a large number of workloads in power modelling. The error of the presented model is very low, particularly considering the large variance in power consumption between workloads (Figure 4.31, grey bars).

The model formulation allows one to see how the static power (which also includes the background dynamic power) and how each PMC event contributes to the overall power (Figure 4.31). Moreover, it can be seen that the information provided by the PMC events on the *type* of workload is essential to producing an accurate model. Each PMC event makes a significant contribution to the dynamic power prediction, working independently to identify different workload types (Figure 4.32), showing the merit of the presented PMC event selection method (Step 2) and its importance.

4.4.7 Sensitivity to Sampling Frequency

In PMC-based models, events are sampled at intervals. However, if the sample period is too low, the CPU frequency is too low, and there are very low levels of activity on the CPU core, then the PMCs may not increment fast enough between samples to give accurate values to the power equation. To observe the point at which this phenomenon occurs, the Cortex-A15 model was implemented and run at various sampling frequencies while the cluster clock frequency was set to the minimum value with no workloads running, except for the model itself and experiment monitoring software. Even in this worst-case scenario, the model error does not increase until the sampling frequency is beyond 500 Hz (Figure 4.33). Many techniques can, however, be employed to reduce this effect, such as adjusting the sampling frequency with the clock frequency, only relying on fast counters when there are low levels of activity on the core or detecting when a particular counter is too slow and sampling that particular counter at a lower rate. The proposed model formulation and

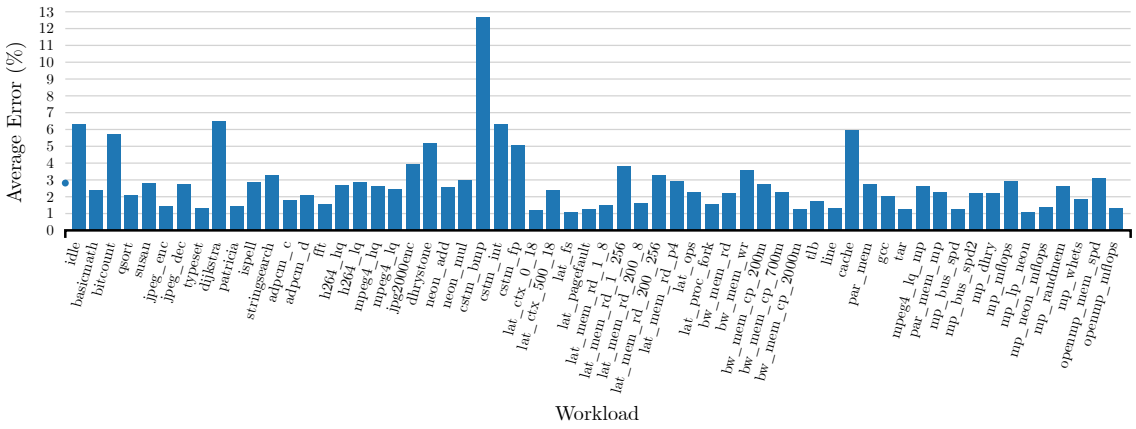


Figure 4.30: Average error (mean absolute percentage error [MAPE]) across each DVFS level for all 60 workloads (Cortex-A15 CPU model)

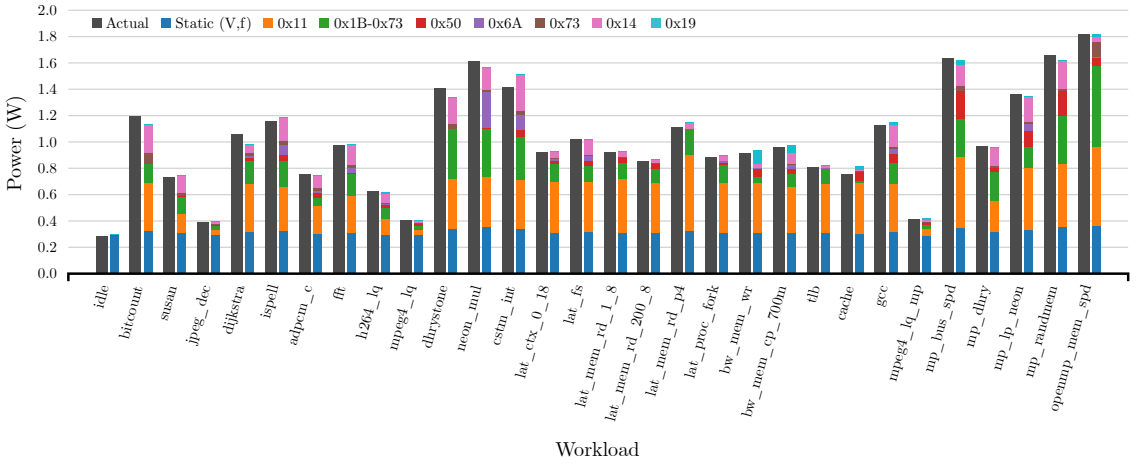


Figure 4.31: Actual (measured) power vs. the predicted power for half of the considered workloads, with the predicted power broken down into its constituting parts (Cortex-A15 CPU model)

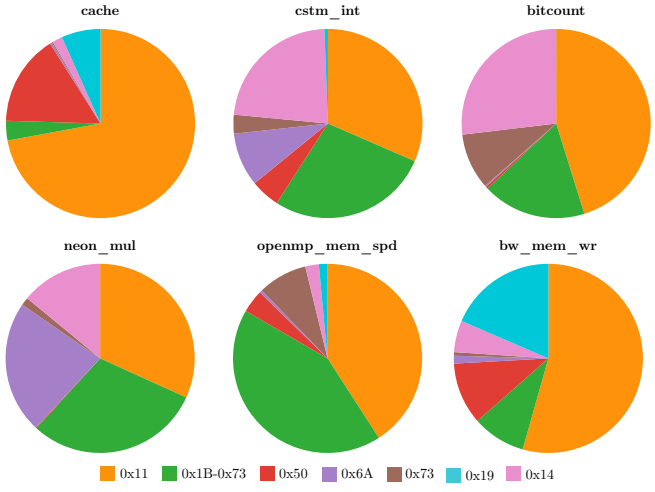


Figure 4.32: Contribution of each PMC event to the dynamic power prediction for six different workloads

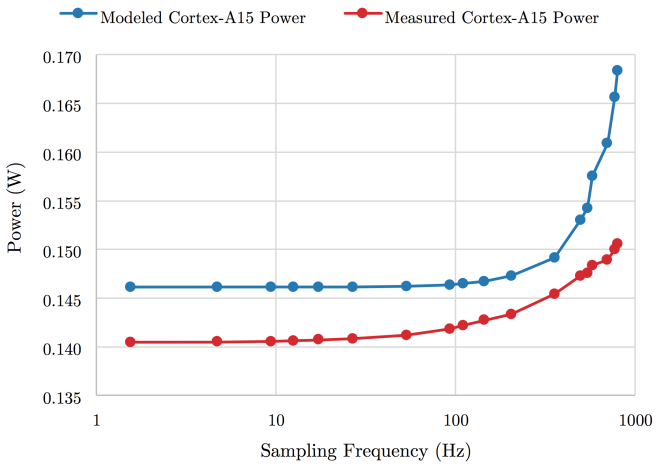


Figure 4.33: Modelled power and measured power for the Arm Cortex-A15 cluster running at 200 MHz with various sampling frequencies and no other workloads

stability allows the contribution of each individual event to be accurately known and if, for example, one PMC event is not occurring regularly enough (event 0x19 in this case), it can be dealt with individually with limited knock-on effects to the other coefficients and therefore the overall power consumption. Another source of error variance in this example is the overhead of writing the results to a file (only required for this experiment); as the sampling frequency increases, the more time the CPU spends running this workload, changing the *type* of workload running. The model accuracy naturally changes with the workload (Figure 4.31).

Figure 4.33 also shows the worst-case power overhead, which includes extracting and recording data from both the power sensors and the model for evaluation purposes; this is considerable compared to the overhead of the model itself. This work presents a modelling *methodology* that can be used with a variety of platforms and in many scenarios (both offline and online). When implementing the model, many optimisations and trade-offs can be made, depending on the number of available counters, required accuracy, sample frequency, etc. For example, with a fixed sampling frequency and known set of DVFS levels, many of the variables in the formula can be pre-computed and the software using this model can simply switch equation when changing DVFS level.

4.5 Comparison with Existing Works

In this section, the presented Cortex-A15 model developed using the proposed approach is compared to five models from four recent works in mobile PMC-based run-time power modelling.

The existing works are implemented in the hardware platform and are compared directly. To be consistent, each model is re-trained and implemented (including the proposed model), using data from the same experiment (the coefficients are re-calculated using OLS regression). Although the presented model formula works for any given voltage and frequency, existing works typically consider a single DVFS level or each one separately, and so this comparison considers a single clock frequency of 1 GHz. This simplification of the proposed model causes slight discrepancies between the statistics reported in this section and those reported in previous sections. The power for the quad-core Cortex-A15 cluster is modelled and the model inputs have been calculated considering

Table 4.9: Parameters of models included in this comparison

	Source	No. Evts.	n	Adj. R^2	Err. [%]
a	Pricopi *et al.* [238]	6	6	0.747	12.5
b	Walker *et al.* [305]	4	4	0.785	12.3
c	Rethinagiri *et al.* [243]	5	3	0.672	12.7
d	Rodrigues *et al.* [244]	3	2	0.760	15.2
e	Rodrigues *et al.* [244]	6	4	0.897	9.7
P	Proposed	6 + C	7	0.999	2.9

the activity of the overall cluster. The process for building and evaluating all of the models is identical, with the only difference being the choice of model inputs and the model equation.

Pricopi *et al.* [238] build a PMC-based power model for an Arm Cortex-A15, considering just a single core of their multi-core cluster (*Model a*, Table 4.9). Walker *et al.* [305] present equations for modelling an Arm Cortex-A8 single-core CPU using four PMC events (*Model b*, as presented in Chapter 3). The dual-core Cortex-A9 model presented by Rethinagiri *et al.* [243] is also considered, removing the frequency term as only one frequency is considered in this comparison. It is not clear in the original work whether the cache miss rate term in their model should take the L1 instruction cache into account, however, doing so would require more than six PMC events in total to implement their model, which is not possible on the platform considered in this work or the Arm Cortex-A9 (the original work used a simulator to obtain PMC data instead of recording it from a real board directly). Rodrigues *et al.* [244] present several models with varying numbers of PMC events, concluding that the same three PMC events (number of fetched instructions, L1 cache hits and dispatch stalls) can be used to yield an acceptable error ($< 5\%$) across multiple architecture types, including both high performance and low power CPUs. They simulate two cores representative of an Intel Nehalem and an Intel Atom processor using SESC and Wattch. Unfortunately, on this Arm-based platform, there is no PMC event that represents dispatch stalls. This work therefore implements their model named *Exp 2* (*Model d*), which does not use dispatch stalls and uses just two model inputs (but requires three PMC events to calculate on the platform), and *Exp 6* (*Model e*) which uses five model input events, but the unavailable dispatch stalls event has been omitted, meaning the implementation of this model has just four model inputs. Two inputs to *Model e* each require two PMC event counters to derive on the platform. One term in this model counts the L1 cache hits, which is implemented as just the L1 data cache hits as all of the limited number of performance counters were in use. The proposed model implementation uses all six PMCs and the separate cycle counter. The number of required PMC events (*No. Evts.*) and the number of model independent variables (n) used in each model is shown in Table 4.9.

These five models (a, b, c, d, e) and the proposed (P) model are trained using the small set of diverse workloads (discussed in Section 4.3). The adjusted R^2 is calculated to measure how well the models fit their *training* data (Table 4.9). The high adjusted R^2 value achieved by the proposed model (*Model P*) demonstrates how the chosen inputs and model formula captures the largest amount of useful data for predicting power consumption, allowing the proposed model to closely fit the training data. Out of the implemented existing models, *Model e* captures the largest amount of useful information and therefore most closely fits the dataset. The VIF of each independent variable was analysed to give an indication of the model stability. *Model e* has four independent

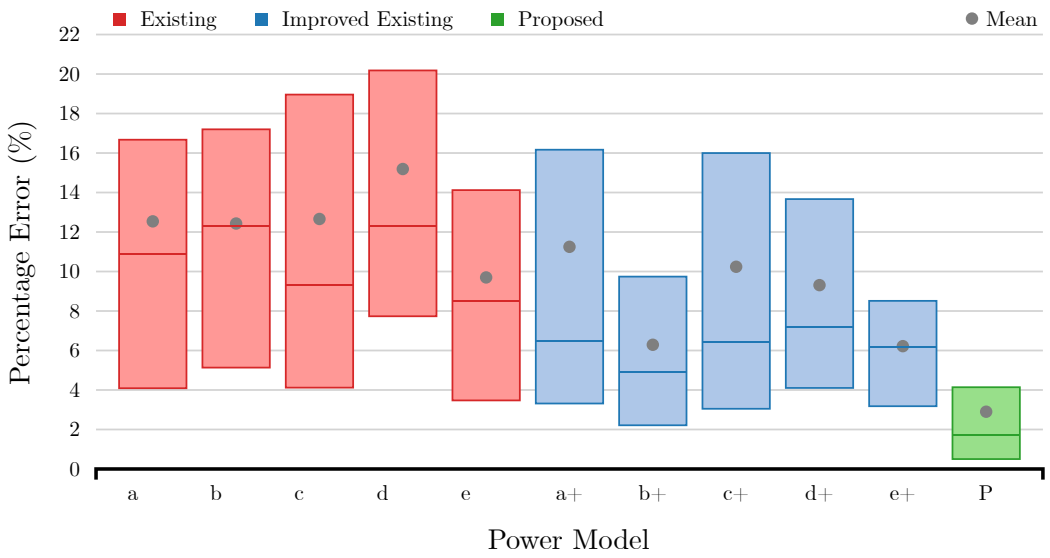


Figure 4.34: Box plot of error distribution for each model, trained with 20 diverse workloads and validated with 60 workloads, showing the median, lower quartile, upper quartile and arithmetic mean

variables and two of the coefficients have a VIF of 15.7 and 9.7, meaning that these coefficients have a standard error 4.0× [$\sqrt{15.7}$] and 3.1× larger than if no multicollinearity was present. Despite capturing more useful data from the model inputs, the proposed model has less variance inflation, signalling a small amount of repeated data in the seven inputs; the highest VIF of any coefficient in the proposed model is 4.8. The effect of errors in the coefficients can clearly be seen in Table I of [243], where two power models (*Exp 4* and *Exp 5*) for the same system feature the same input (dispatch stalls, D) but with a coefficient of +1.25 in one model and −0.47 in the other. This shows how the modelling methodology is not capturing how each variable individually affects the power consumption, forming an unstable model.

Each model is then validated on the full set of 60 workloads (equivalent to the D bars in Figure 4.20) and the resulting error distribution calculated (Figure 4.34, red and green boxes). The proposed model best utilises the limited number of training workloads to achieve an average error of 2.9%; 3× smaller than the next best model. Furthermore, the box plot in Figure 4.34 shows that the proposed model has a significantly narrower range of errors across the 60 workloads. This means that greater confidence can be placed in the proposed model to contain the error within a smaller interval. For example, the error of *Model e*, which has the smallest average error out of the existing works, has a maximum error of 49%, whereas the proposed model has a maximum error of 13%. The approximate 95% prediction interval (explained in Section 4.4) of the proposed model is ±0.014 W whereas the prediction interval for *Model e* is ±0.13 W.

Pricopi *et al.* [238] (*Model a*) report a low error of 2.6%. However, they build and validate with a small number of 15 workloads and report a minimum and maximum power consumption of 4.54 W and 5.16 W across their training workloads, respectively; a very narrow range when compared to many of the testing workloads (Figure 4.31). It was found that two independent variables of the model have VIFs larger than 130, contributing to the high average error across the diverse set of 60 workloads (Figure 4.34). This highlights the importance of considering stability and validating

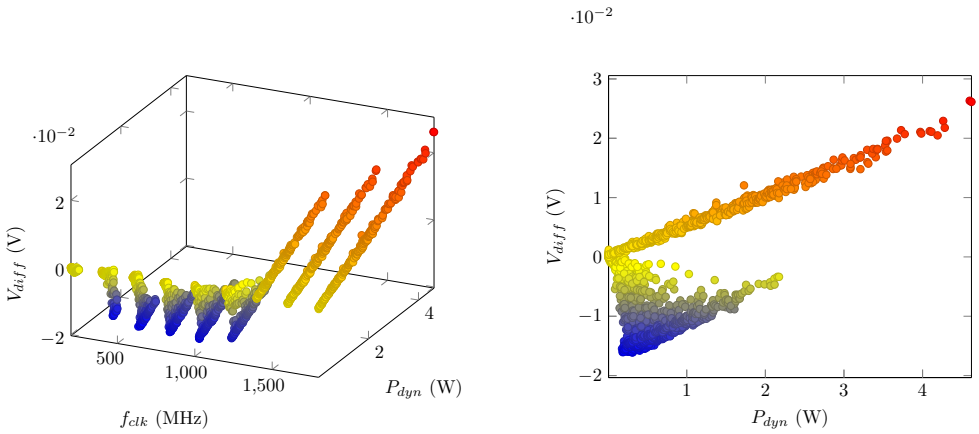


Figure 4.35: The difference between the CPU idle voltage and the measured voltage, V_{diff}, against the CPU dynamic power consumption, P_{dyn}, for workloads running at various DVFS levels, f_{clk}.

with a large number of diverse workloads when building reliable power models.

In Section 4.3 it was highlighted how the cycle count provides unique data to the model (it is used to determine how much time the cores spend in a low-power [inactive cycle] state). However, this event is not included in any of the existing models (they either use simulated data that may not take this feature into account or an old development board that does not have this feature enabled). The existing models (*a*, *b*, *c*, *d* and *e*) were therefore re-built with the cycle count and named *a+*, *b+*, *c+*, *d+* and *e+*, respectively. This improves the existing models but the proposed model still achieves an error 50% lower than any other model (Figure 4.34, blue boxes). Note that the cycle counter is *not* used to calculate the IPC (instructions-per-cycle) as it only counts *active* cycles on this platform.

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

4.6 CPU Voltage Model

The presented power models work with any specified frequency or voltage, which is important for design-space exploration. In online run-time management scenarios, the voltage cannot be directly measured and so the idle voltage for the current operating frequency can be used. However, the voltage supplied to the CPU by the (non-ideal) voltage regulator varies under the power drawn by the CPU (previously observed in Figure 4.29b). Furthermore, output voltage characteristics of the

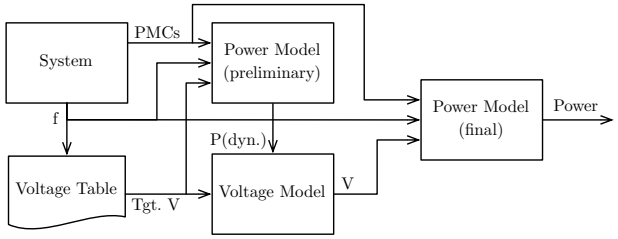


Figure 4.36: Run-time power estimation setup with the voltage model

voltage regulator is different for each DVFS level (Figure 4.35). This phenomenon greatly affects the power consumption and needs to be taken into account when modelling the power.

This section presents a model built using multiple linear regression to predict the CPU voltage from the current CPU clock frequency, target voltage (idle CPU voltage at that frequency), and dynamic power calculated from the PMC events (Step 4 of the power modelling methodology, Figure 4.1). The presented power model is first used to perform a preliminary prediction of the dynamic power using the target voltage and then this is fed into the voltage model, along with the target voltage itself (Figure 4.36). This voltage model then outputs the estimated run-time voltage and this is used as the voltage input of the main (final) power model. This voltage model is specific to the platform and each frequency needs to be separately considered. To the extent of the authors' knowledge, this is the first work in this area to consider voltage change due to the dynamic load and to demonstrate how it can be modelled.

To give an example, at a frequency of 1800 MHz, the power model error is 2.6% when using the measured CPU voltage. However, as the measured voltage may not be available to the run-time manager of a real device, the target voltage is used, resulting in a significantly larger power error value of 8.5%. If the voltage model is used, the changing run-time voltage can be accounted for and the power modelling error is reduced to 3.0%. This section identifies the problem, quantifies its effect on the error, and demonstrates how a proposed solution is effective in mitigating it. A more detailed analysis of this phenomenon (including the effects of temperature) is the topic of future work (Section 7.2).

4.7 Conclusion

This chapter has presented a detailed and statistically-rigorous automated methodology and corresponding software tools for building accurate and stable PMC-based run-time power models. The approach was illustrated using measured data from two mobile CPUs with significantly differing microarchitectures: an Arm Cortex-A7 and Arm Cortex-A15. The resulting models achieved an error of 3.8% and 2.8%, respectively, and both achieved an R^2 value of over 0.99.

The approach uniquely considers model stability and this chapter demonstrates how it allows the resulting models to make more accurate predictions on a vast set of diverse observations, even when trained on a limited set of workloads. Techniques for applying unsupervised classification to PMC event data are also presented.

Furthermore, the problem of heteroscedasticity in run-time CPU power models is identified and addressed and this chapter also shows how the model formulation and stability allows one to reduce the

model formulation experiment time by 100× while trading off less than 0.6% error. Techniques for identifying outliers and determining workloads that require different features for accurate estimation are presented.

The models are deployed on a real device and the error and overhead as a function of sample period analysed. A detailed comparison with the state-of-the-art was also conducted. This chapter also highlighted how the CPU voltage supplied by the non-ideal voltage regulator is sensitive to the dynamic activity of the CPU and a CPU voltage model is presented, which improves the accuracy of the power model by as much as 5.5% in situations where the voltage cannot be measured.

To address the lack of an established method of collecting PMC data on mobile devices, the platform-dependent experimental software tools were released to enable other researchers make use of high-quality, measured data from mobile platforms for investigations requiring measured CPU PMC, temperature, voltage or power data.

Chapter 5 extends the models presented in this chapter by modelling the effects of the temperature (measured using the thermal sensors) on the power consumption, modelling CPU idle states, decomposing the power consumption further, observing the power effects of moving data in the memory hierarchy, and proposing a temperature model.

Chapter 5

Thermally-Aware Composite Run-Time CPU Power Models

The power models developed in Chapter 4 were demonstrated to be accurate and stable, and *absorbed* the effects of temperature due to the CPU voltage, frequency and switching activity. However, the effects of ambient temperature on the power consumption were not captured, and idle power states were not considered.

The methodology and models developed in this chapter were built on top of those presented in Chapter 4, and rely on the robust model formulation (presented in Section 4.4) that captures the relationships between the input features and the power consumption effectively. Using this as a solid foundation, a new methodology for adding thermal-awareness and analytically decomposing the power into its constituting parts is presented. Experimental data was collected from the quad-core Arm Cortex-A15 cluster of the Hardkernel ODROID-XU3 development board (as used in Chapter 4) and the resulting model achieved a MAPE 3.7% across 39 diverse workloads, eight Dynamic Voltage-Frequency Scaling (DVFS) levels and with a CPU temperature ranging from 31°C to 91°C. Moreover, the effect of switching cores offline was investigated and the power model was further decomposed to estimate the static power of each CPU and L2 cache, the dynamic power due to constant background (BG) switching, and the dynamic power caused by the activity of each CPU individually. The power impact of moving data between levels of the memory hierarchy was also analysed, and a methodology for developing a simple CPU temperature model proposed.

Much of the work presented in this chapter was published in the 26$^{\text{th}}$ International Workshop on Power and Timing Modeling (PATMOS), 2016 [308].

5.1 Introduction

Chapter 3 highlighted the lack of existing work on *top-down* power modelling that used measured PMC-data combined with power measurements collected on real mobile devices, largely due to technical challenges involved in doing so. Chapters 3 and 4 overcome this and significantly improve on the state-of-the-art by presenting a statistically-rigorous methodology that ensures stability, deals

with heteroscedasticity and achieves a low error on three mobile CPUs of differing microarchitecture. However, the effects of ambient temperature on power consumption are not considered and the model could be improved by further decomposing the estimated power consumption, and therefore providing more information on *how* the power is consumed.

This chapter uses the methodology in Chapter 4 as a starting point and models the effects of temperature on the CPU power consumption and adds more advanced thermal compensation.

Furthermore, a recent study by Zhang *et al.* found that most multi-core smartphone models were ill-suited largely due to CPU idle states not being considered [334]. The effect of switching cores offline on the CPU power consumption is measured and modelled in this chapter. This also enables the power consumption to be analytically decomposed.

The proposed model significantly improves the accuracy across wide temperature ranges, results in a more stable model with statistically significant coefficients, and provides a breakdown of the power consumption. This provides higher quality online data to a run-time manager. Moreover, the power breakdown allows this accurate and validated top-down model to be combined with bottom-up techniques (discussed in Chapter 6), allowing some flexibility in model specification for research and design-space exploration applications. Using a combination of the two, as opposed to relying fully on a bottom-up approach, means that more confidence can be placed in the resulting estimations.

The methodology in this chapter focusses on mobile CPUs due to the significant importance of energy-efficiency in mobile applications and because the diversity of mobile workloads makes power modelling particularly difficult. However, the presented methodology is generic and applicable to other systems (e.g. desktop and server CPUs). The resulting models themselves accurately model the specific CPU implementation they were built on.

The key contributions of this chapter are:

1. A novel methodology for modelling the effects of temperature on the static power consumption in PMC-based power models;
2. An accurate, thermally-compensated, PMC-based power model for an Arm Cortex-A15 that takes CPU idle states into account;
3. Decomposition of the state-of-the-art PMC-based power models to give static and dynamic power estimations for individual components;
4. An evaluation of the power effects of accessing various levels of the memory hierarchy;
5. A proposed temperature modelling methodology for use in systems without thermal sensors (such as full-system simulation frameworks).

5.2 Experimental Setup

The ODROID-XU3 development board by Hardkernel, which contains an Exynos-5422 SoC (System-on-Chip) featuring a quad-core Arm Cortex-A7 and quad-core Arm Cortex-A15 CPU is used. This is the same hardware platform as used in Chapter 4, and is described in detail in Section 2.4. In this work the higher-performance Arm Cortex-A15 cluster is used to illustrate the presented approach.

Thirty-nine diverse workloads were used from several benchmarking suites: MiBench [116], which is a suite of representative embedded workloads; LMbench [190], which contains microbenchmarks for activating and testing specific microarchitectural behaviours, such as memory reads at a specific level of cache; Roy Longbottom [179], which contains many multi-threaded workloads that make heavy use of the NEON SIMD processing unit and OpenMP; ParMiBench [132], which is a multi-threaded version of MiBench; and, ALPBench [173], which contains complex, parallel multimedia workloads.

5.3 Regression-Based Modelling Methodology

The power modelling methodology presented in Chapter 4 was used to develop the baseline power models, which were the starting point to the work presented in this chapter. The Powmon software tools (implementing the methodology presented in Chapter 4) were used to create power models. The models were therefore identical to those presented in Chapter 4 with the exception that they were trained with a different set of workloads (discussed in Section 5.2). The selected PMC events were kept the same as those selected in Section 4.3 (see Table 4.2) and the model terms were identical to the ones specified in Table 4.4.

The baseline models achieved a MAPE of 3.34% and an adjusted R^2 of 0.998 across the 39 workloads. The model equation for this baseline model is:

$$P_{cluster} = \underbrace{\left(\sum_{m=0}^{M-1} \beta_m E_m V_{DD}^2 f_{clk}\right)}_{\text{dynamic activity}} + \underbrace{f(V_{DD}, f_{clk})}_{\text{static and B.G. dynamic}} \tag{5.1}$$

where M is the total number of PMC events in the model; m is the index of each event; E is the cluster-wide PMC event rate (events-per-second) after being divided by the operating frequency in MHz, f_{clk}, and averaged across all cores; and V_{DD} is the cluster operating voltage. $P_{cluster}$ is the power for the overall quad-core Cortex-A15 cluster. This model formula breaks down the power consumed by the dynamic CPU activity and the idle power (which includes the static power and background (BG) switching power, and hence includes the f_{clk} term).

Such a model formulation is built by understanding the relationships between the variables. For example, a known component of CMOS power consumption is the dynamic power caused by switching activity, which is proportional to $V_{DD}^2 f_{clk}$. Therefore, the PMC events, which indicate dynamic activity, should only be included in the dynamic power calculation and be inserted into the model after being multiplied by $V_{DD}^2 f_{clk}$ (dynamic activity, Equation 5.1). Note that the PMC event rates have already had their frequency component removed in post-processing.

This approach has several benefits: it allows the power consumption to be broken down; results in a more stable model with more accurate coefficients; results in a model with more physical meaning; and allows relationships and power contributions to be deduced.

For example, it can be shown that there is a constant dynamic power component (BG switching) present by building the model with and without the constant $V_{DD}^2 f_{clk}$ (BG dynamic, Equation 5.2). When this component is added, the accuracy of the model increases significantly and the corresponding p-value of its coefficient is very low ($p < 0.0001$), indicating strong statistical significance.

One can therefore infer that there is a constant dynamic power component (i.e. present even when there is no activity on the cluster) and its magnitude can be estimated. The equation can therefore be rewritten as shown in Equation 5.2. Later in this chapter, it is shown how this can be used to decompose the power model further and understand the L2 cache behaviour when all cores are switched offline. This is not possible with the typical approaches.

$$P_{cluster} = \underbrace{\left(\sum_{n=0}^{N-1} \beta_n E_n V_{DD}^2 f_{clk}\right)}_{\text{dynamic activity}} + \underbrace{\beta_b V_{DD}^2 f_{clk}}_{\text{BG dynamic}} + \underbrace{f(V_{DD})}_{\text{static}} \tag{5.2}$$

5.4 Temperature Compensation

This section describes how the power models are extended to make use of the instantaneous temperature readings from the on-board thermal sensors to significantly improve the power prediction accuracy. The effect of CPU temperature on the static power consumption is modelled.

In Section 2.1 it was found how the static power consumption is made up of several leakage mechanisms which are dependent on temperature, including the sub-threshold leakage current, I_{sub}, the gate-oxide leakage current, I_{gate}, and the band-to-band tunnelling leakage current, I_{btbt}:

$$I_{leak} = I_{sub} + I_{gate} + I_{btbt} \tag{5.3}$$

The sub-threshold leakage current increases exponentially with temperature (Equation 2.6, page 12) and was the largest leakage power component. However, with more aggressive technology scaling the gate-oxide leakage current and band-to-band tunnelling current have become the dominant leakage power components. While the gate-oxide current has little dependence on the temperature, the band-to-band tunnelling current increases with temperature due to the narrowing band-gap (Equation 2.12).

The models built in Chapter 4 do not include any temperature compensation from the thermal sensors, despite its large effect on the static power consumption. The temperature of the device is mainly influenced by the CPU voltage (V_{DD}), the CPU switching activity, and the ambient temperature conditions. A relatively accurate model is achieved in this previous work by including higher order terms related to V_{DD} and f_{clk} in the static power equation (e.g. V_{DD}^3) to *absorb* the error in the static power estimation caused by the temperature changing due to the DVFS level. It does not absorb the error caused by temperature changes due to the ambient temperature. This section uses recorded thermal data to build a thermally-aware model that has superior accuracy, even with extreme ambient temperature differences. As well as the accuracy being significantly improved, this temperature-dependent model is better-suited for combining with bottom-up approaches and applying theoretical equations to further decompose the model.

Three experiments were run: one with the default fan settings on the ODROID board; one with the fan set to its maximum speed; and one with the fan off completely. All experiments were conducted under the same ambient conditions (though the environment was not thermally-controlled) and 39 different workloads were run at clock frequencies from 200 MHz to 1600 MHz in steps of 200 MHz.

Table 5.1: Comparison of three models using different model equations and under different thermal conditions.

	Model Eqn.	V. T	Avg. Error (%)	Adj. R^2	SER (W)
a	5.1	N	3.34223	0.997691	0.0423673
b	5.2	Y	5.97732	0.994498	0.0604720
c	5.5	Y	3.67805	0.997418	0.0414211

Two models were then built (labelled *a* and *b*, Table 5.1) using the methodology and software tools presented in Chapter 4; the first one using data collected at normal ambient conditions and built using Equation 5.1, and the second one using data from all three experiments (varying temperature, V. T) and Equation 5.2.

When observations taken with different fan settings are used with a model without thermal compensation (model *b*), the average error increases to 6%. This average value is optimistic as many of the observations have similar temperatures, particularly at lower frequencies. However, for observations with significantly varying temperature, the errors can be large. For example, when idle at 1600 MHz, the three experiments (automatic fan setting, fan on, fan off), which have average temperatures of $43.5°C$, $40.2°C$ and $73.5°C$, respectively, achieve average errors of 1.6%, 2.7% and 25%; despite the relatively low average error, individual observations with significant temperature deviations can have large errors. The thermally-aware power model, which is presented later in this section achieves errors of 1.2%, 0.96%, and 5.0% for these observations, respectively, therefore improving the accuracy by as much as 20%.

This dependence on temperature needs to be approximated in the regression model and the static power model requires the CPU temperature, T, as an input feature (Equation 5.5). When the residuals, $\boldsymbol{e}$, of model *b* were plotted against the temperature, it could clearly be observed that this relationship was not being captured by the model (Figure 5.1). A regression model was then built to predict the residuals using the CPU temperature. It was found that Equation 5.4 adequately estimated the residuals, without requiring a more computationally complex exponential term. These terms were then incorporated into the power model.

$$Residuals = \alpha_a T^2 + \alpha_b T \ + c \tag{5.4}$$

$$P_{cluster} = \underbrace{\left(\sum_{n=0}^{N-1} \beta_n E_n V_{DD}^2 f_{clk}\right)}_{\text{dynamic activity}} + \underbrace{\beta_b V_{DD}^2 f_{clk}}_{\text{BG dynamic}} + \underbrace{f(V_{DD}, T)}_{\text{static}} \tag{5.5}$$

As previously stated, the model created in Chapter 4 does not use temperature as an input but *does* indirectly compensate partially for temperature. As V_{DD} increases, the temperature, T also increases. The model in Chapter 4 includes V^2 and V^3 terms in the static power equation to absorb the temperature effects due to the cluster voltage. An important step is to remove all the terms that absorb these temperature effects before adding the new temperature compensation. Not doing so would result in a model with too many inputs and the same effect being captured by a combination of several inputs. This would result in both a less stable model and over-fitting of the temperature relationship.

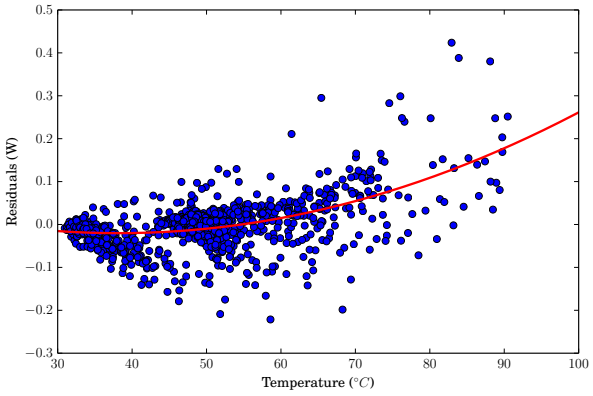


Figure 5.1: Residuals of model b plotted against temperature. Red line shows an equation predicting this trend.

Table 5.2: Cortex-A15 Model Coefficients and p-values, grouped into three components: dynamic activity (Dyn. act.) power; constant background dynamic power (BG Dyn.); and static power.

Comp.	Coefficient	Weight	p-value
Dyn. act.	0x11×V^2f	5.966e-10	$p < 0.0001$
Dyn. act.	0x1b-0x73×V^2f	8.885e-10	$p < 0.0001$
Dyn. act.	0x50×V^2f	1.091e-8	$p < 0.0001$
Dyn. act.	0x6a×V^2f	1.622e-8	$p < 0.0001$
Dyn. act.	0x73×V^2f	3.445e-10	$p < 0.0001$
Dyn. act.	0x14×V^2f	4.413e-10	$p < 0.0001$
Dyn. act.	0x19×V^2f	2.900e-9	$p < 0.0001$
BG Dyn.	V^2f	1.508e-4	$p < 0.0001$
Static	Intercept	-1.785e+0	$p < 0.001$
Static	VT^2	8.894e-4	$p < 0.0002$
Static	VT	-6.877e-2	$p < 0.003$
Static	V	1.850e+0	$p < 0.0009$
Static	T^2	-8.592e-4	$p < 0.0003$
Static	T	6.807e-2	$p < 0.004$

All of the V^2, V^3 and f_{clk} terms were removed from the static power components of the existing model, and the temperature compensation terms were added. The static power equation now only contains a combination of V, T^2, and T components, in addition to an intercept (Table 5.2). A lower apparent average error can be achieved by including every combination of these three components as input features. However, doing so reduces the quality of the model and it over-fits mechanisms within the model. Six terms are carefully selected (including the intercept) to predict the static power consumption, all of which have an associated p-value of less than 0.003 indicating that they are all statistically significant [201]. The dynamic power due to activity uses the PMC events (indicated by their hexadecimal identifier, Table 5.2) selected in Chapter 4, which should be referred to for a detailed explanation of the PMC events and the selection method.

The resulting temperature-compensated model equation (Model c, Table 5.1) achieves a 10-fold cross-validated mean absolute percentage error of 3.7% with observation temperatures as low as

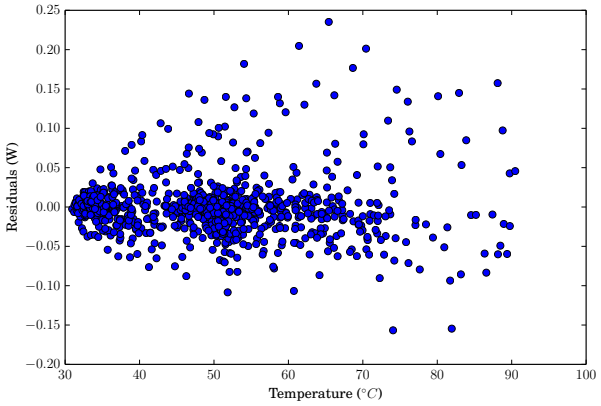


Figure 5.2: Residuals of model c (temperature-compensated) plotted against temperature. Red line shows an equation predicting this trend.

Table 5.3: Model results

Parameter	A15 Value
R^2	0.9974
Adjusted R^2	0.9974
No. Observations	1014
Std Err. of Regression (SER) $[W]$	0.0414
F-Statistic	14103.8
p-Value for F-Statistic	$p < 0.000001$
Avg. VIF (PMC events only)	2.78
Avg. VIF (inc. V and f)	3.56

30.9°C and as high as 90.6°C.

The same graph was re-drawn to show the residuals against temperature and there was no longer a trend between the two, confirming that the model has successfully captured the effect of temperature on the power consumption (Figure 5.2). The cone shape of the residuals indicates the presence of heteroscedasticity (non-constance variance, see Section 3.1.6), which is inherent to PMC-based power modelling. This problem is addressed by using a heteroscedasticity-consistent standard error (HCSE) estimator as described in Section 4.4.2.

Figure 5.3 shows how the modelled static power varies with temperature at several fixed voltages, while Figure 5.4 shows how it varies with voltage at several fixed temperatures. Later in this Chapter it will be shown how temperature impacts the static power for measured observations. A summary of the model results is presented in Table 5.3 while the cross-fold validated results are presented in Table 5.4.

5.5 Model Decomposition and Offline Cores

In the previous section a thermally-aware model that estimates the overall power of a quad-core Arm Cortex-A15 cluster has been described. In this section, that model is decomposed, allowing it

Table 5.4: Model results from k-fold cross-validation

Parameter	A15 Value
No. Folds (k)	10
Fold Group Size	101
Avg. Err. (MAPE) [%]	3.73
Mean Sq. Err. (MSE) [W^2]	0.00186
Root Mean Sq. Err. (RMSE) [W]	0.0384

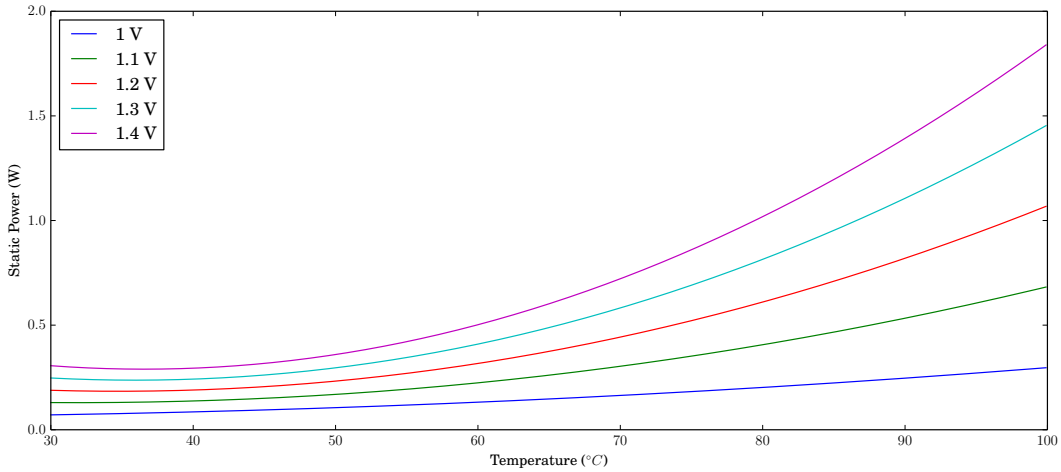


Figure 5.3: Model output response when varying the temperature input at various cluster voltage points.

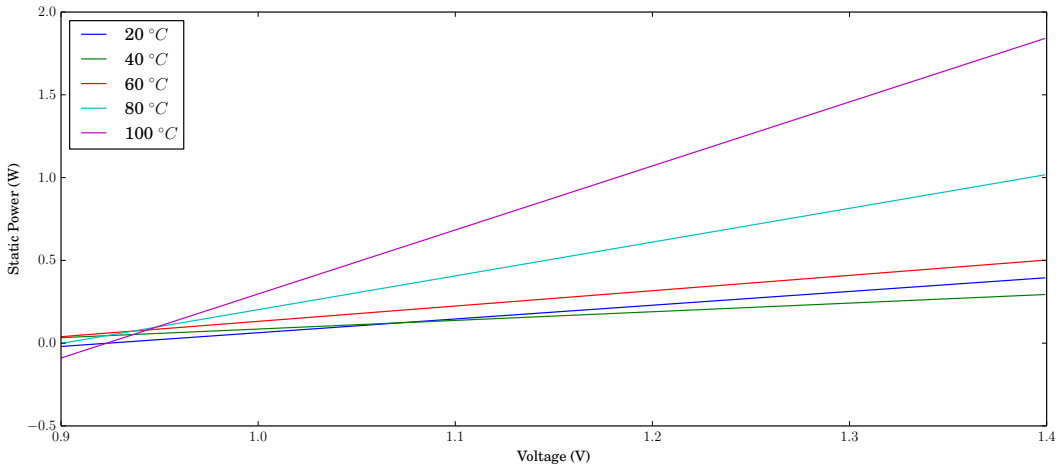


Figure 5.4: Model output response when varying the voltage input at various temperature points.

to give accurate estimations of the static and dynamic power consumed by each core and to predict the power consumption when cores are switched offline.

In Chapter 4, the PMC events used to estimate the dynamic activity power consumption are averaged across all four cores in the cluster. Therefore, the assumption is that the dynamic power impact of a PMC event is the same, whether, say, 95% of the cluster-wide events come from one core and 5% from the rest, or each core contributes to 25%. This is confirmed by estimating the dynamic activity power of each core individually and comparing this directly with the measured

power consumption. The modelling methodology presented in Chapter 4 calculates the arithmetic mean of the PMC events across all four cores and uses this data to calculate the model coefficients. Therefore, the counters from each core are taken and then divided by four, before multiplying each event by the calculated coefficient. This per-CPU estimated dynamic activity power consumption includes the dynamic activity of the CPU itself and the dynamic power of the L2 cache caused by that CPU.

The quad-core Cortex-A15 power is then measured when it is idle and each core is switched off one-by-one to observe how static power is consumed (Figure 5.5). This experiment is conducted at every DVFS level to see how it varies with voltage (plotted in red) and frequency (x-axis). The blue line shows the measured cluster power consumption when all four cores are online but idle. The purple line shows the power when only one core is online and the brown line shows the power when all cores are switched offline.

The power decreases significantly when the last core is switched off. At first glance one may assume that this may be because the shared 2 MB L2 cache is switched off at this point. However, it is observed that, even when the voltage is constant (e.g. 200 MHz-800 MHz), the power (when more than 0 cores are online) increases with frequency, showing that there is a component of dynamic power. As shown in Section 5.3, a constant dynamic component (BG dynamic) is present. This component occurs even when there is no activity (e.g. the system is idle) and it is therefore present in Figure 5.5. This component is calculated using V_{DD}^2 and f_{clk} as well as the static power of the final online core (by subtracting the average power between n=4, n=3, n=2 and n=1 from n=1). It is found that the magnitude of these two components at each frequency fit the gap between the measured power when all four cores are off and n=1. The difference between n=1 and n=0 is therefore due to the BG dynamic power. It can therefore be inferred that the drop in power when the last core is switched off is due to a constant dynamic power (BG dynamic) component disappearing (possibly due to clock gating in the L2 cache and bus) and that the L2 cache remains powered but inactive, as there is a significant static power component remaining.

The idle power can therefore be broken down into the following components:

- static power of the L2 cache and cluster-wide logic;
- constant background (BG) dynamic power which is present when one or more cores is online;
- static power of each of the four cores and their respective L1 caches.

Unlike with the dynamic power consumption, the proportion of static power consumed by each component remains constant, with respect to the total static power, across each DVFS level. Each core and its L1 cache consumes 11.8% and the remaining static power consumption, consisting largely of the L2 cache, makes up 52.6%.

It is therefore possible to break down the static power estimation, which takes both voltage and temperature into account. The base static power (mostly consisting of the L2 cache static power) is calculated using the average temperature value recorded from all four thermal sensors, while the static power for each CPU is calculated using its respective thermal sensor.

With this idle power breakdown added to the model, it can be seen how the static power is broken down across different DVFS points when the fan was manually switched on (Figure 5.6) and when the fan was permanently off (Figure 5.7).

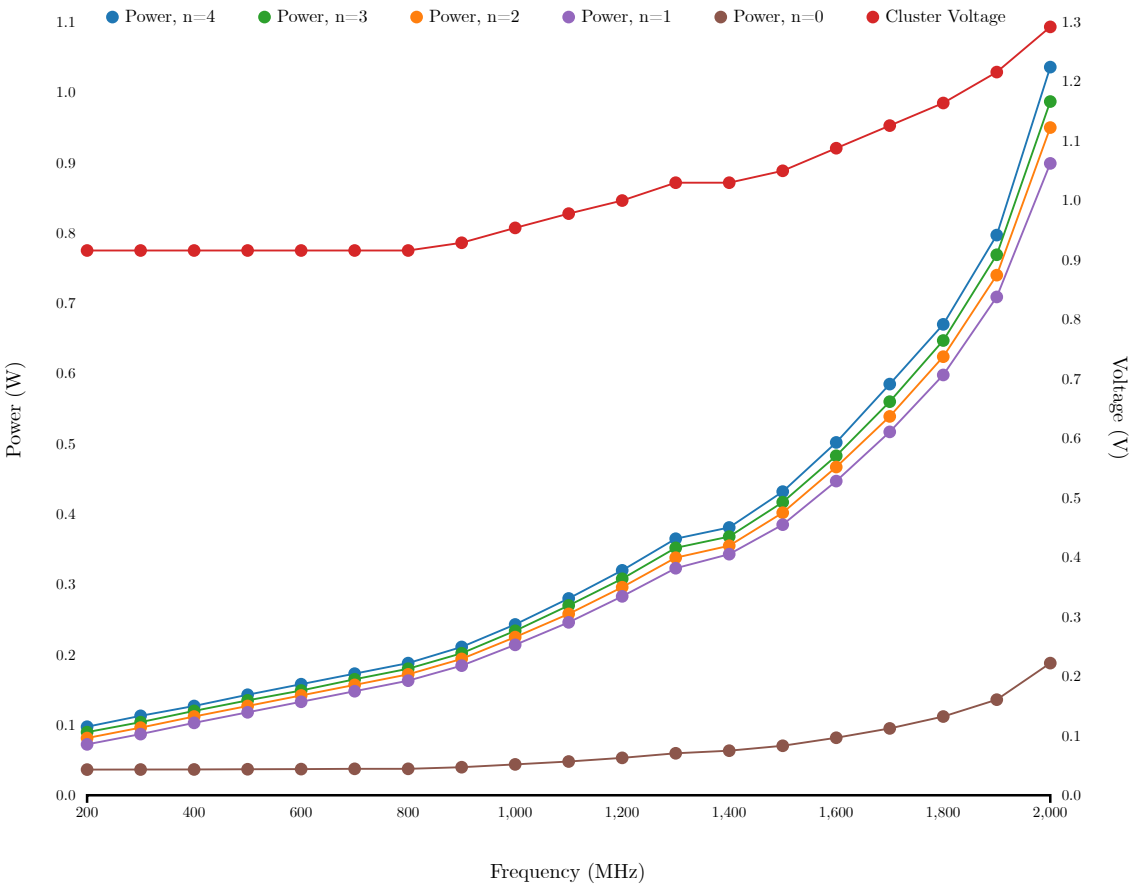


Figure 5.5: The measured idle quad-core Cortex-A15 cluster power at various clock frequencies (f_{clk}) when all four cores are online (n=4) and switching each off in turn until no core is online (n=0). The cluster voltage is also shown.

Note how the static power remains almost constant at 400 MHz, 600 MHz and 800 MHz whereas BG dynamic power increases as it is proportional to the clock frequency. There is a small increase in static power from 400 MHz to 800 MHz due to the temperature increasing with the faster switching activity.

At 1600 MHz with the fan on, the static power consumption is less than 0.14 W (temperature: 40.2 °C) and with the fan off, the static power consumption is over 0.33 W (temperature: 73.5 °C). This illustrates the importance of considering temperature within a model, particularly if the model is providing per-component static and dynamic power predictions. When summing the individual components of static power and the BG dynamic component, the model achieves an average idle power prediction error of less than 7% across the full range of temperatures considered and every DVFS level.

Each component of power is calculated and summed to obtain the overall cluster power and it is then compared with the measured cluster power. As previously reported, the model achieves an average error of 3.7%. Figure 5.8 shows the error of each individual workload (aggregated over DVFS levels and fan settings). The power consumption and its breakdown of each workload is also plotted, again aggregated over every DVFS level and fan setting (Figure 5.9). The static power varies across different workloads because the intensity of the workloads affect the temperature. It also varies because the voltage provided by the non-ideal voltage regulator changes with the dynamic

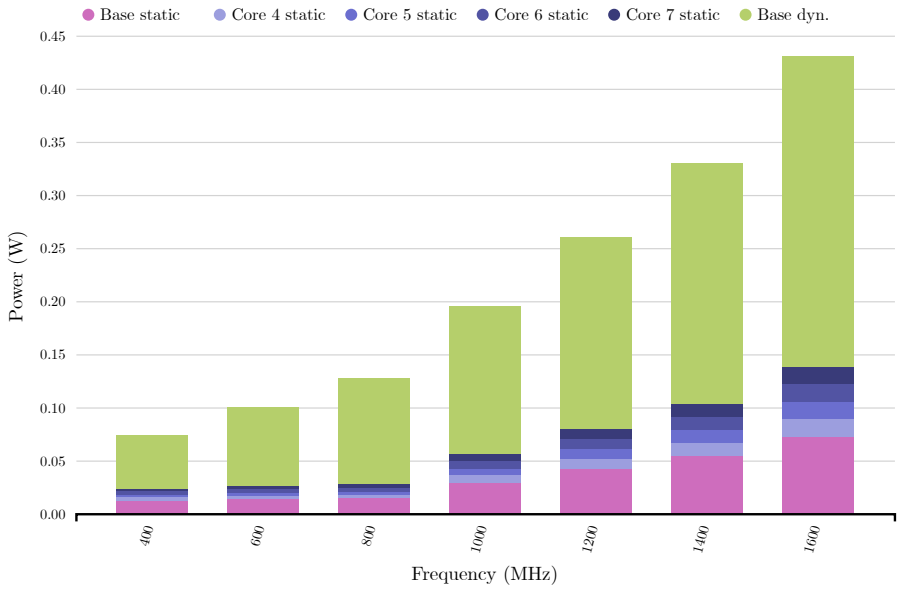


Figure 5.6: Idle power consumption across different clock frequencies, showing the contribution from the static power and BG dynamic power. Fan switched on. Average temperature is 35.5 °C.

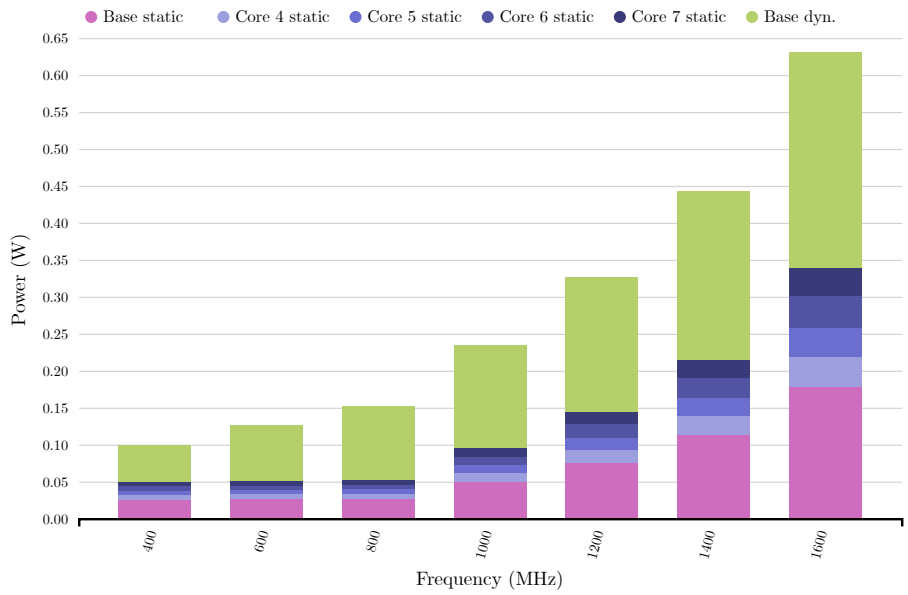


Figure 5.7: Idle power consumption across different clock frequencies, showing the contribution from the static power and BG dynamic power. Fan switched off. Average temperature is 60.1 °C.

CPU current draw. This effect was identified and addressed in Section 4.6.

5.6 Thermal Modelling

This chapter has presented an improved power modelling methodology to that presented in Chapter 4 which models the effects of temperature, as measured from the thermal sensors (therefore including the effects of ambient temperature), on the leakage power consumption. However, some platforms, including simulation frameworks, do not have temperature sensors. This section extends

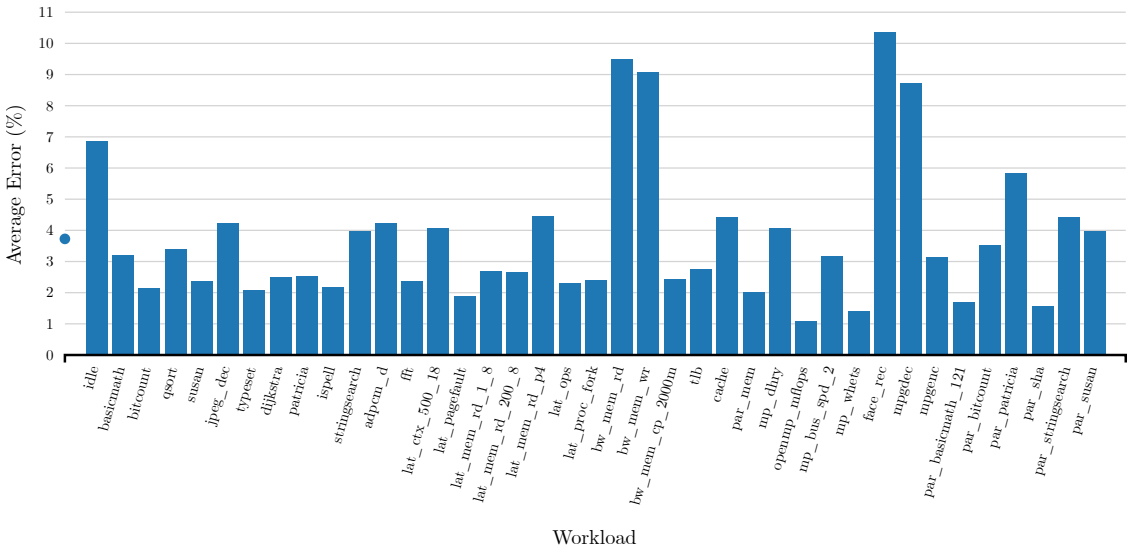


Figure 5.8: Average error (mean absolute percentage error [MAPE]) for all 39 workloads, aggregated across each DVFS level and three fan settings

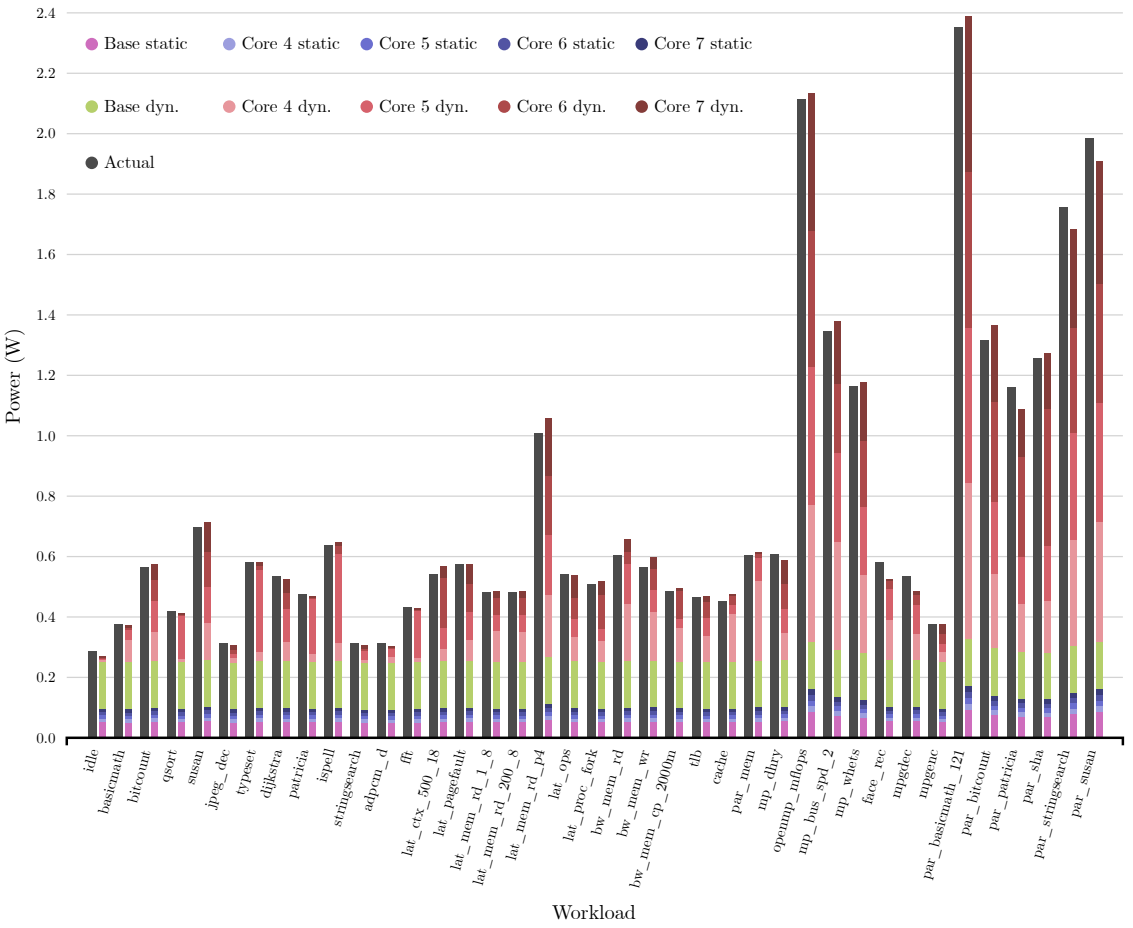


Figure 5.9: Power breakdown for all 39 workloads, aggregated over every DVFS level and three different fan settings

the methodology and software tools to enable empirical thermal profiling.

The HotSpot [127] tool enables thermal modelling for design analysis, as well as run-time thermal management, or dynamic thermal management (DTM) [269]. Mesa-Martínez, Ardestani and Renau [195] characterise an AMD K8-based processor using an infrared (IR) imaging system to provide insights and recommendations to researchers working on thermal modelling. Lee and Skadron [170] present a PMC-based thermal model for an Intel Pentium 4 processor, which can be used for online temperature sensing. However, they conclude that many challenges in developing a reliable, low-overhead temperature sensing mechanism remain.

This section proposes a simple equivalent circuit (lumped element) temperature model (Table 5.5). The ground potential is made equal to the ambient air temperature, which is, in practice, influenced by heat dissipation of other components in the device. Note that this technique is modelling the value of the built-in thermal sensors, it is not deriving the core temperature, which varies between the specific components inside the CPU. GemStone was extended to enable monitoring of the thermal sensors. The power modelling analysis and gem5 calibration (see Chapter 6) analysis extracted average workload samples. GemStone therefore had to be heavily extended for the time-series thermal analysis, which also required fusing data from multiple sources that used different sampling rates. A RaspberryPi board was set up to record ambient thermal conditions while the GemStone software tool captured PMC events and on-board thermal readings while running workloads (Figure 5.10). Workloads were run while different numbers of cores were idle, on different cores, and at different DVFS levels. GemStone was configured to add a period of idle workload execution before and after running each workload so that the transient thermal effects could be monitored (Figure 5.13).

Once the thermal capacitance and thermal resistance (between the sensor and ambient temperature) have been derived, the simple, yet abstract temperature model can be used to provide run-time thermal estimations using the basic equivalent circuit (Figure 5.11). The input of the thermal model is the estimated power consumption of the power model.

There are certain practical factors that must be considered. For example, the thermal sensors corresponding to each core are not all aligned in the same way to their corresponding core. Furthermore, there are other components on the chip, including the GPU and memory controller, that significantly impact the temperature.

Once a thermal capacitance and resistance has been derived, the thermal model can be used to provide online temperature predictions and improve the accuracy of the power model leakage estimation (Figure 5.12).

5.7 Energy Cost in the Memory Hierarchy

In this section it is shown how it is possible to derive the energy cost of the movement of data between different levels of the memory hierarchy. The *lat_mem_rd* [189] micro-benchmark from LMbench [190] was used measure the read latency of accessing various sizes of arrays. As the array size increases, the memory latency increases as the CPU is accessing more distant memory (L1 data cache, L2 cache and main memory; details of the memory hierarchy are provided in Section 2.4). The effects of TLB misses and prefetching can also be observed in the latency plots. A modified

Table 5.5: Thermal variables with their electrical equivalents

Thermal	Electrical
Temperature at spatial location (K)	Node Voltage (V)
Heat Flow (W)	Current (A)
Thermal Resistance (K/W)	Resistance (Ω)
Thermal Capacitance (Ws/K)	Capacitance (F)

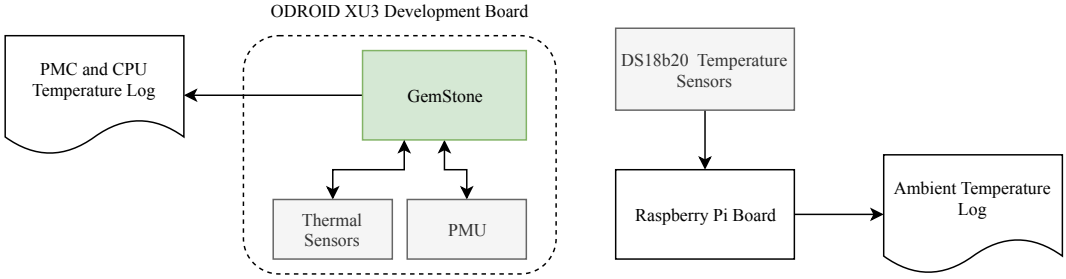


Figure 5.10: Overview of the thermal characterisation setup

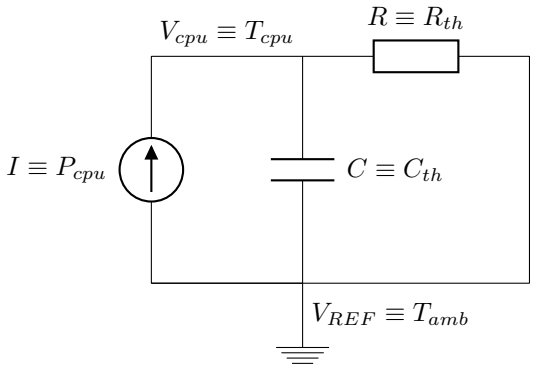


Figure 5.11: Example of the most basic from of an equivalent circuit CPU temperature model

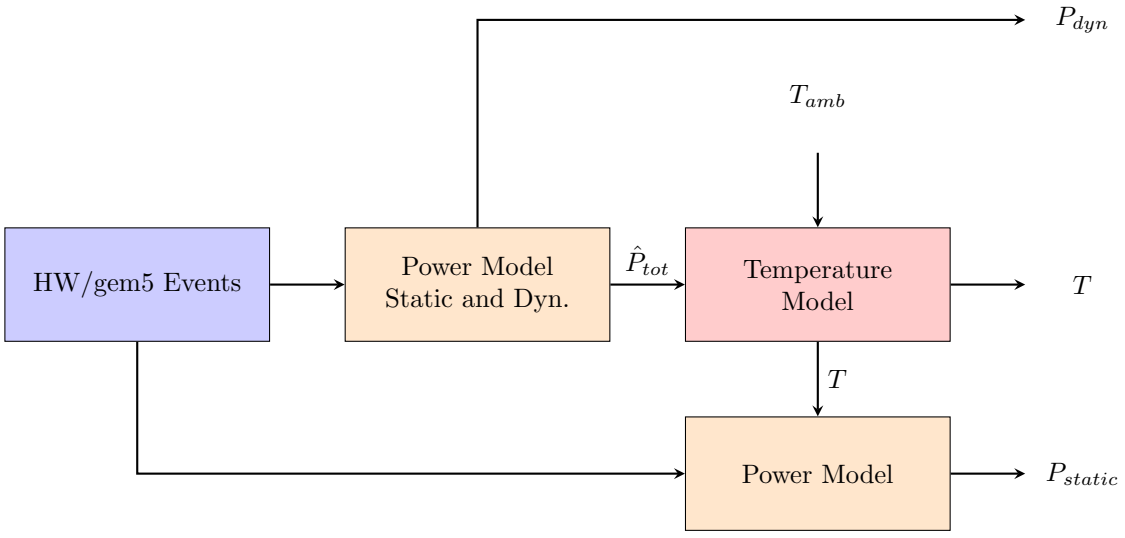
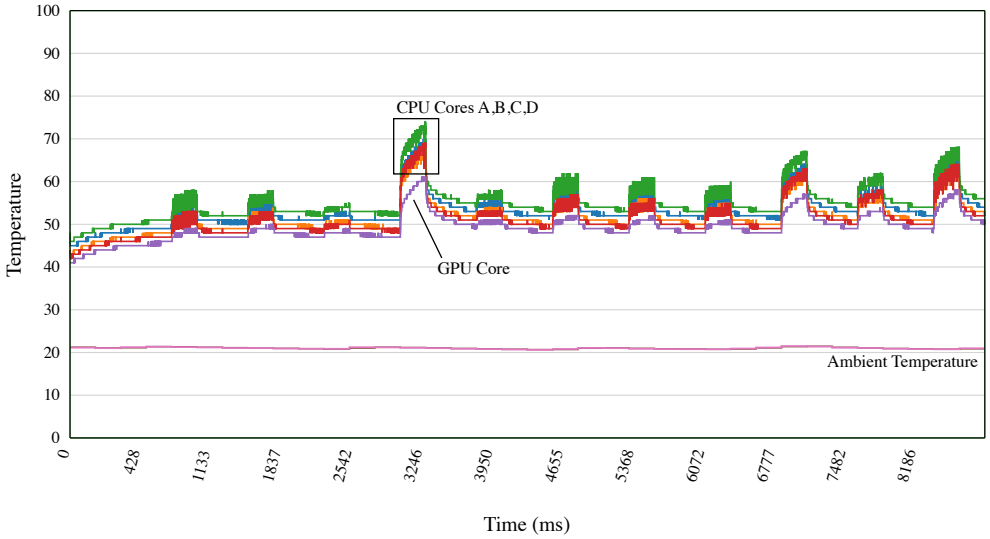
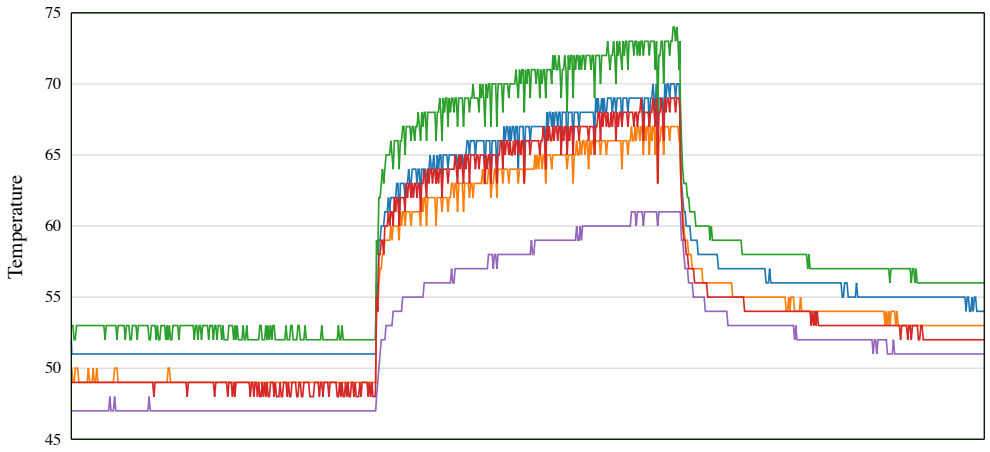


Figure 5.12: Online thermal and power prediction block diagram



(a) Set of workloads

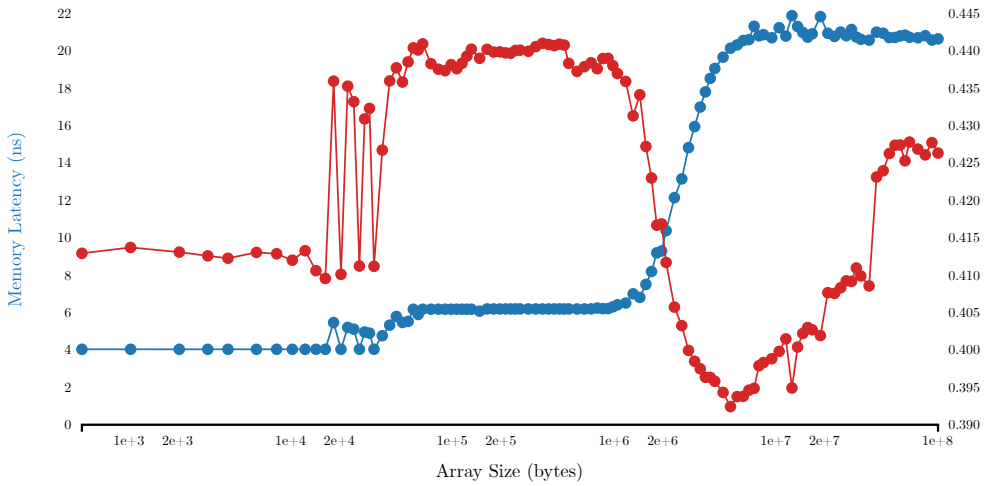


(b) Parallel Susan implementation (ParMiBench)

Figure 5.13: Temperature (degrees Celsius) readings from the four CPU core sensors, a GPU core sensor, and an ambient temperature sensor while running selected workloads on the Arm Cortex-A15 cluster

version of the *Powmon* software was used to simultaneously record specific PMC events related to the memory hierarchy as well as the power consumption. All experiments were conducted on the Arm Cortex-A15 cluster of the ODROID-XU3 board. By carefully tuning micro-benchmarks and combining the latency data with the PMC data and the measured power, it is possible to derive the energy cost of moving data between different levels of the memory hierarchy. The stride of the reads could be adjusted which affected how effectively the memory prefetchers operated. The examples in Figures 5.14, 5.15, 5.16 and 5.17 use a small stride of 8 bytes and the L2 prefetcher can be seen to be active, which affects the power consumption and event rates. The stride was used to obtain samples with different amounts of prefetching to aid the decomposition of the power consumption.

As the array size increases, the memory latency increases in steps as it accesses more distant levels of the memory hierarchy. The Cortex-A15 cluster power consumption (which contains the four cores, their respective L1 instruction and data caches, as well as the shared L2 cache) varies as different levels of the memory hierarchy are accessed (Figure 5.14).



(a) Cortex-A15 Power (W)

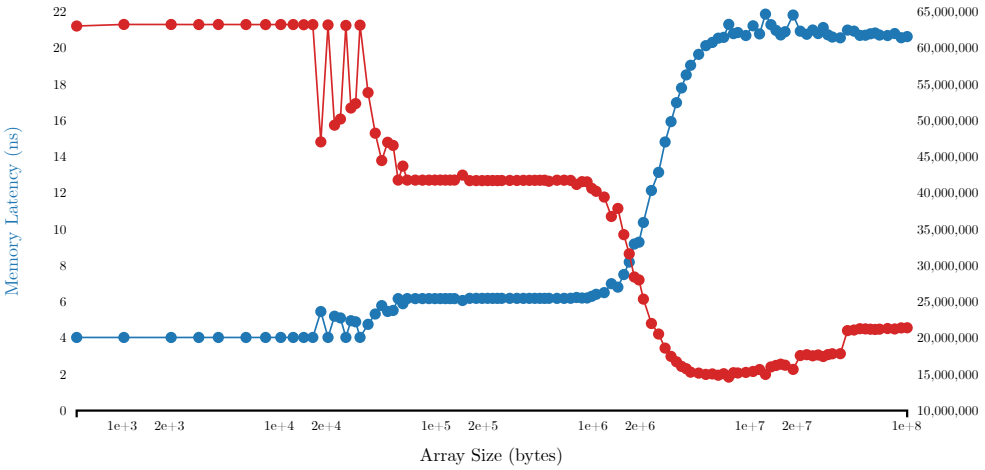
Figure 5.14: Memory latency (left axis) plotted with Cortex-A15 power consumption (right axis). Maximum array length of 100 MB and a stride of 8 bytes.

The first level of graph shows the memory latency of accessing the L1D cache (4 ns), as confirmed by the PMC event rate of L1D cache accesses (Figure 5.15a). After the size of the array becomes larger than approximately 32 KB (the size of the L1D cache), the rate of the L2 cache accesses increases (Figure 5.15b). The maximum Cortex-A15 power consumption occurs at this point as the energy cost of reading data from the large L2 cache is very high. After the array size exceeds approximately 2 MB, the main memory is accessed, as indicated by the bus accesses (Figure 5.15c). At this point the main memory power consumption also increases dramatically (Figure 5.16b). After the array size increases to above 7 MB, there is another increase in power consumption (both Cortex-A15 and memory power) and the number of bus accesses. This is due to the L2 prefetcher (see Section 2.4) activity (Figure 5.16a), which is higher than the case where the stride is set to a higher level (such as 128 bytes).

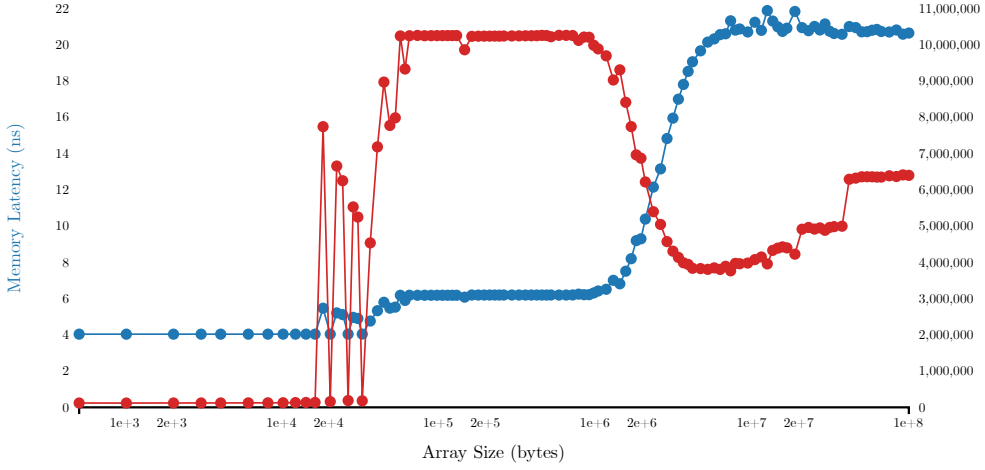
This analysis enables the energy required to access various levels of the memory hierarchy to be understood and quantified. Furthermore, Chapter 4, experimentally demonstrated the importance of diverse training observations, which can be obtained by taking samples at these four identified levels. The LMbench lat_mem_rd benchmark was modified to enable a specific point of the memory latency plot to be identified and held for accurate power sampling. This workload was also used in Chapter 6 for measuring the memory latency of a full-system simulator (where sweeping the entire range requires an impractical amount of computation time). In Section 7.2, an adaptive model training workload is proposed by building on this idea and code.

5.8 Conclusion

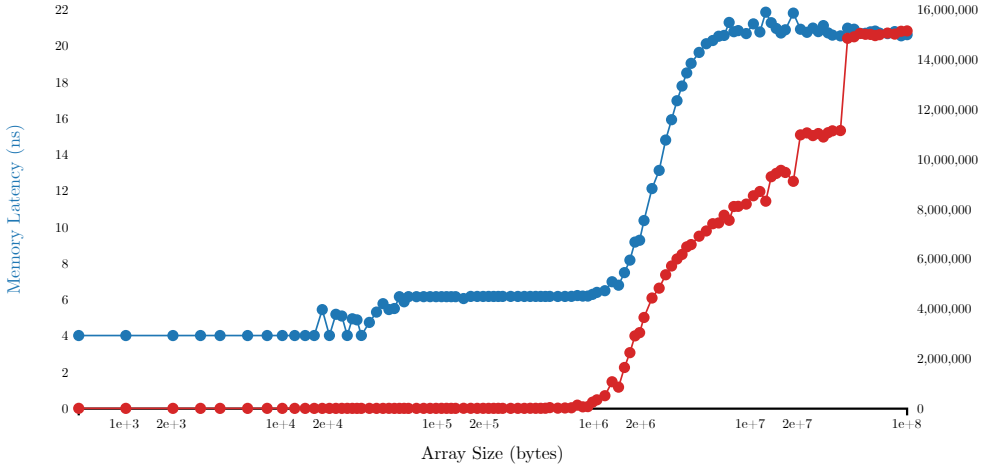
This chapter presented a methodology for developing accurate and decomposable thermally-aware run-time power models which have uses in both online energy management and design-space exploration. The approach was illustrated using measured data from a mobile device utilising an Arm Cortex-A15 quad-core CPU. This chapter demonstrated how it is possible to build stable models that can efficiently capture the relationships between the model inputs and power, allowing one



(a) `0x04` L1D Accesses (rate)

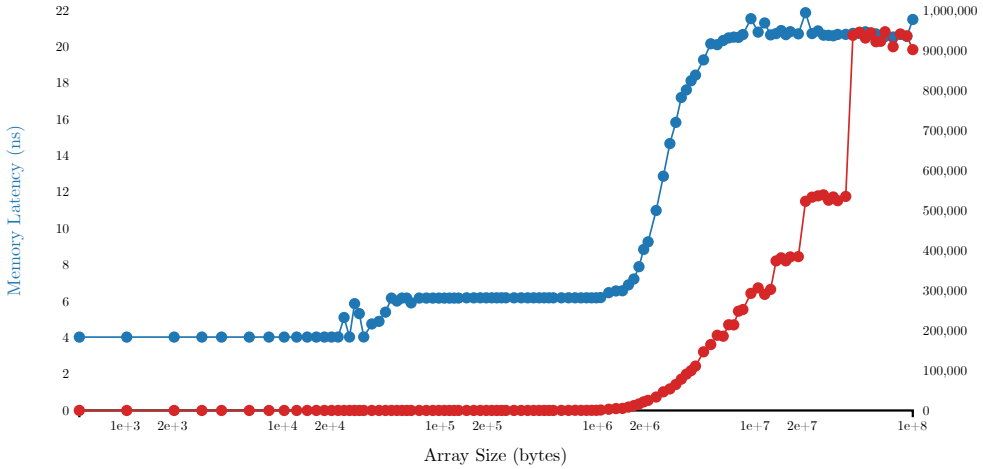


(b) `0x16` L2D Accesses (rate)

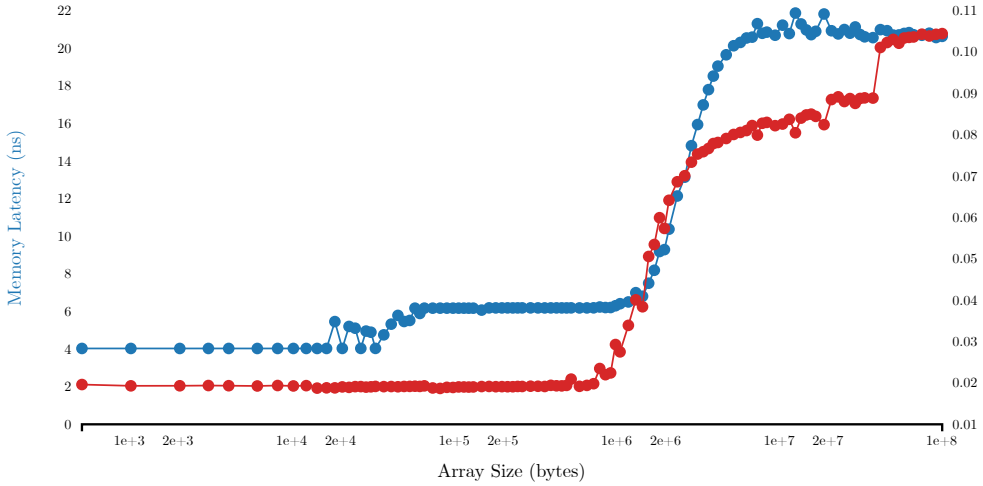


(c) `0x64` Bus Accesses (rate)

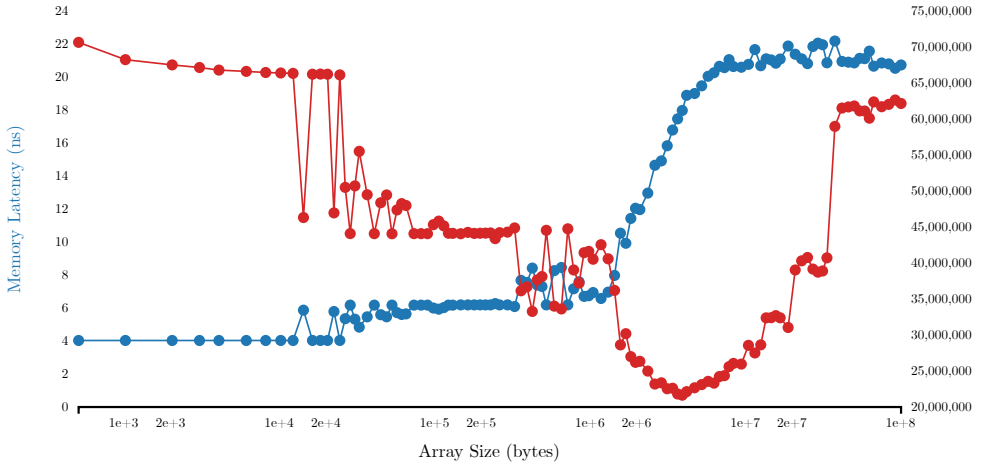
Figure 5.15: (continued) Memory latency (left axis) plotted with various PMC event rates (right axis). Maximum array length of 100 MB and a stride of 8 bytes.



(a) `0xe8` L2 Cache Prefetches (rate)

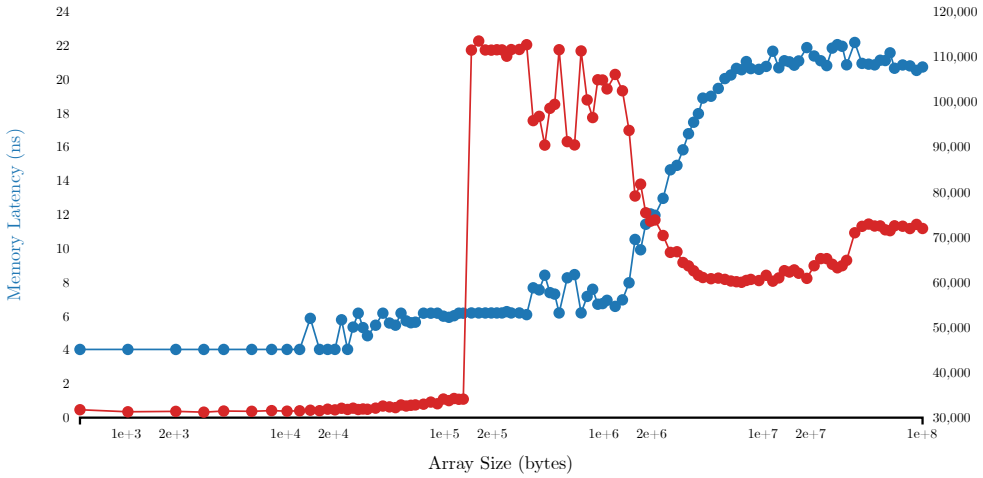


(b) Main Memory Power (W)



(c) `0x08` Instr. Arch. Exec. (rate)

Figure 5.16: (continued) Memory latency (left axis) plotted with PMC event rates and memory power consumption (right axis). Maximum array length of 100 MB and a stride of 8 bytes.



(a) `0x05` L1 TLB Refill (rate)

Figure 5.17: (continued) Memory latency (left axis) plotted with a PMC event rate (right axis). Maximum array length of 100 MB and a stride of 8 bytes.

to decompose the power model into smaller components, by significantly improving on the model formulation presented in existing works. The effect of temperature on the static power consumption was analysed and thermal compensation was added to the model. This chapter demonstrated how it improves the accuracy by as much as 20%. The power impact of switching CPUs offline was evaluated and power components were analytically devised to infer internal CPU behaviour and break down the power consumption. The presented models are able to accurately predict the temperature-compensated static power of each CPU and the L2 cache, the background dynamic switching power, and the dynamic power caused by the workload of each CPU. The validity of the analytical methods and assumptions were verified by testing the proposed model using measured data with large temperature deviations, a diverse selection of workloads, eight different voltage and frequency (DVFS) levels and with different numbers of CPU cores utilised. Each component of the model was calculated individually and summed together to form the cluster power estimate, which was found to have an average error of 3.7%. Furthermore, a temperature model was proposed for situations where thermal sensors are not available, such as simulation tools, and a technique for analysing the energy cost of data movement within the memory hierarchy was proposed, which can also be used to collect diverse training observations for power model training.

The novel techniques and methodologies proposed in this chapter, and the previous two chapters, enable accurate and stable power modelling for guiding RTM. Chapter 6 explores the applicability of these empirical power models for design-space exploration purposes and develops techniques for improving the performance models that provide inputs to the power models.

Chapter 6

Hardware-Validated Performance and Energy Modelling

Full-system simulation frameworks such as gem5 are used extensively to evaluate research ideas and for design-space exploration. Moreover, energy-efficiency has become the key design constraint in recent years and many works use a separate power modelling framework to evaluate energy consumption. While such tools are convenient and flexible, they are known to contain sources of error which are often not fully understood and potentially impact the conclusions drawn from investigations. This chapter enables accurate, hardware-validated performance, power, and energy modelling of CPUs by first presenting a methodology to evaluate and identify sources of error in CPU performance models, and secondly integrating and optimising the empirical models developed in Chapters 3, 4 and 5 for use with such performance models. This chapter proposes coupling top-down empirical power models with performance simulators, and postulates that this approach provides more accurate and reliable results for studies using simulation tools when compared with the typical approach of using a performance simulator in conjunction with a high-level bottom-up modelling framework, such as McPAT. Hierarchical clustering, correlation analysis, and regression techniques are used to identify sources of error without requiring detailed CPU specifications and enable existing models to be improved, new models to be developed, validation of simulator changes, and testing of model suitability for specific use cases. Furthermore, the GemStone open-source software tool is presented, which automates the process of characterising hardware platforms, identifying sources of error in gem5 models, applying power analysis, and quantifying the effect of errors on the performance, power, and energy estimations. In addition, the mean percentage error in execution time was found to swing from -51% to $+10\%$ between two versions of the same gem5 model, underlining the need for an automated tool to validate models against reference hardware, ensuring accuracy and consistency.

The work presented in this chapter was published in the 2018 IEEE International Symposium on Performance Analysis of Systems and Software [304].

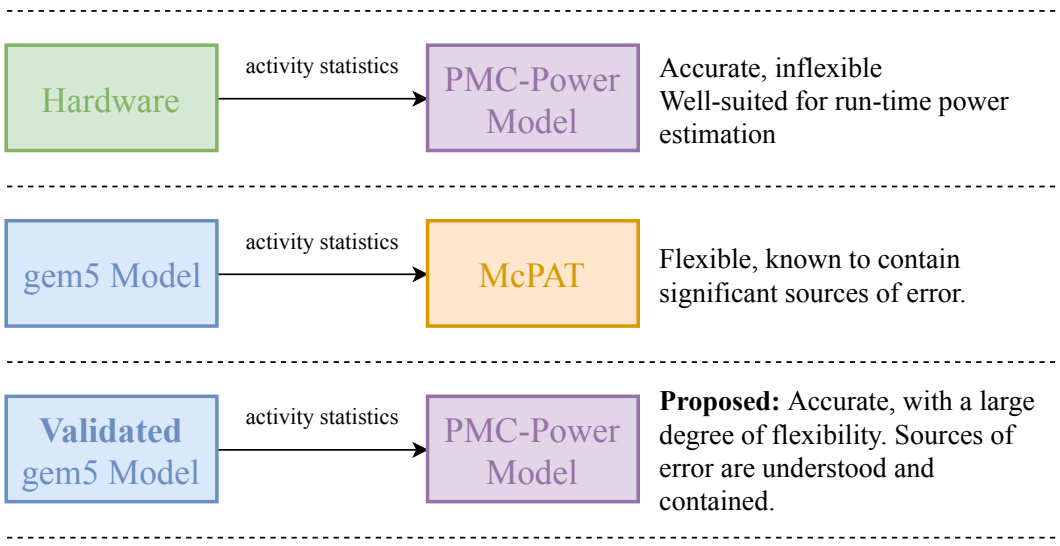


Figure 6.1: Proposed simulation methodology

6.1 Introduction and Literature Review

Architectural and micro-architectural CPU simulators are heavily relied upon in both academia and industry to evaluate new ideas and proposals in many fields, such as: system-level design [40], microarchitecture [221], compilers [83], benchmark suite analysis [301], scheduling and run-time system management [320], reliability [64], emerging memory technologies [61, 181], persistent main memory [142], genomic algorithm acceleration [258], high-performance computing [49], cloud computing [264], and distributed systems [200]. Due to its active development community and support for many Instruction-Set Architectures (ISAs), including x86 and ARM, gem5 [30] has become a widely used simulation framework (see Section 2.7.1)

In recent years, energy efficiency has become a primary design constraint in modern computer systems, and architectural simulators are often coupled with an energy simulation tool. For example, McPAT [174] is a widely used power, area and timing simulation tool that is often used with gem5 to provide energy analysis. Many recent works have used gem5 to provide performance statistics to McPAT for energy analysis [218, 56, 318, 289, 246, 239], despite large sources of error in McPAT being well documented [315, 216, 48, 171].

While such simulation tools are invaluable to research, they inherently contain errors which can impact the conclusions drawn from research, particularly if the sources of error are not well understood. This potentially affects the quality and integrity of research that relies on the tools. Recent works have focussed on these errors and their effects [315, 216, 117, 171]. While not as flexible, the carefully developed models in Chapters 3, 4, and 5 demonstrated superior accuracy and a higher level of confidence when compared to high-level bottom-up frameworks.

6.1.1 Simulation Use Cases and Baseline Models

When evaluating research ideas or conducting design-space exploration, accurate performance and power baseline (i.e. reference) models are key to ensure representative results and correct conclusions. A common use case of a full-system simulator is evaluating a proposal for a specific part of the

system. This could be a change in hardware within the CPU itself, such as changing the behaviour (e.g. modifying the way instructions are re-ordered, the cache coherency protocol or replacement policy) or changing the size of components (e.g. the re-order buffer, a TLB, a branch prediction history table, or a cache). The changes could also be related to other system components, such as evaluating emerging non-volatile memory (NVM) technologies for main memory. The proposed change could also be in the software running on the system, for example, system software evaluation (e.g. different scheduling policies or run-time management algorithms), compiler assessment or application software evaluation.

To evaluate such proposals, a baseline model is first used to obtain baseline results, the changes are then applied to the model, and the new model is used to evaluate the changes (e.g. any improvements or deterioration in performance or energy are measured). A baseline model based on a typical system is therefore a critical component of a simulation framework. The gem5 simulation framework provides the flexibility required to make the proposed changes to the model, either through changing the configuration scripts, or extending the framework itself. However, if there are significant errors in the reference model, it may not respond in a representative way to the change under test, potentially leading to incorrect or misleading results.

This chapter proposes calibrating a gem5 model, ensuring it is representative and suitable as a baseline model, before pairing it with an accurate and stable empirical power model. A researcher would apply the changes proposed in their investigation to the calibrated gem5 model and it would provide the modelled impact of the changes on the execution time and the architectural and micro-architectural event counts. The modelled events of the before and after scenario would be used as inputs to the empirical power model for evaluating the power and energy (and optionally also temperature, using the techniques outlined in Chapter 5) change.

While the accurate, hardware-validated empirical model is fixed for a particular implementation, the model does capture the power changes due to the change in system behaviour (captured in the gem5 events). In many use cases there is zero or negligible power change due to a physical hardware change compared to the power difference due to the change in behaviour. For example, there would be zero power difference due to software changes in works investigating HMP scheduling, RTM, compiler optimisation or software improvements. In many works investigating hardware changes, the hardware change itself will have minimal impact on power consumption compared to the behavioural changes (the "knock-on effects"), which are captured by the empirical model through the input statistics. Therefore, using an inflexible, but highly accurate, empirical power model is preferable in many situations to a flexible bottom-up simulation framework that is known to contain significant sources of errors in many subcomponents of the system.

In cases where the hardware change power impact could not be considered negligible, the power consumption of the affected component (which is typically the component of interest to the investigation) can be considered separately. For example, there are two different types of power impact on the system when doubling the size of the branch predictor global history table:

1. The prediction accuracy would likely increase, causing fewer wasted cycles and unnecessary cache and TLB accesses. These "knock-on" behaviour changes are captured accurately by the fixed empirical power model.

2. The increased size of the table will increase the leakage power consumption and the dynamic power of accessing the table will also be higher. This power change is not captured by the

empirical power model and must be modelled using bottom-up approaches such as SPICE, RTL-level, gate-level, or higher-level CACTI simulation. While estimating the power of the whole system using SPICE, RTL, or gate-level simulation may be infeasible due to the simulation time and the lack of publicly available IP, simulating the single component under investigation is feasible and would be the most accurate method of understanding the power impact of the proposed change. In many situations, back of the envelope calculations considering the area change would suffice (it is the "knock-on" effects that are more difficult to model). Once the power impact of the individual component under investigation has been found, it can be analytically combined with the empirical power model.

Many changes that have significant (when compared to the total power consumption) direct effects on the power consumption due to the physical power consumption would likely be related to large structures within the CPU, such as the caches, the TLBs, or the mBTB. Modelling the physical power delta of changing these components is relatively straight-forward using the techniques listed above. Furthermore, unlike small components in the system that have parameters that are not publicly disclosed, the sizes of these large components are typically published in datasheets.

6.1.2 Performance Simulation Error

This work augments the gem5 simulation framework with accurate, hardware-validated empirical PMC-based power models of a real hardware platform. However, through extensive experimental evaluation, significant errors in the workload execution time are found when validating existing gem5 models on a wide selection of workloads. Furthermore, analysis of individual event statistics from the model (e.g. L1D cache misses), which are required as inputs to power modelling tools (including both McPAT and the PMC-based power models), have even larger errors. A key cause is specification error, which explains errors caused by incorrectly setting model parameters due to a lack of information about the device being modelled (see Section 2.7.3). To address this problem, a comprehensive methodology for systematically evaluating gem5 models against hardware platforms and identifying sources of errors is presented (Section 6.4), along with a corresponding software tool, *GemStone*, which automates the process. GemStone collects data from hardware platforms; combines the data with gem5 simulation results; identifies sources of error in the gem5 model using statistical techniques; and evaluates performance, power and energy accuracy (as well as their scaling across frequencies) using the aforementioned power models. While some works have evaluated gem5, its accuracy and usefulness depends on the specific model being used and the purpose it is being used for. Furthermore, the active development community constantly improves gem5, meaning that running the same setup with two different versions of gem5 can produce different results; inevitably, bugs can also be introduced occasionally (Section 6.7). There is therefore a need for an automated tool that compares gem5 models against reference hardware platforms to ensure its accuracy, consistency and applicability for specific use cases. All graphs in this chapter, with the exception of Figure 6.6, are generated by GemStone.

By building and integrating empirical hardware-validated power models into the gem5 simulator itself, sources of error in the power simulation are reduced and the accuracy for a specific CPU is known (Section 6.5).

6.1.3 Contributions of this Chapter

The result is a framework for building accurate, reliable and hardware-validated gem5 models for performance and energy evaluation. This is important as results from performance and energy simulators underpin the results and conclusions of many works of research and development; limitations in simulation tools can lead to incorrect conclusions and reduce the quality of research if they are not understood.

The key contributions of this chapter are as follows:

1. a methodology for evaluating performance models and identifying specific sources of error (Section 6.4);
2. empirical power models of an Arm Cortex-A7 CPU and Arm Cortex-A15 CPU, optimised for gem5 events (Section 6.5);
3. evaluation of how modelling errors affect performance, power and energy estimations, including how they scale with DVFS changes, between micro-architectures, and between different types of workloads (Section 6.6);
4. the GemStone open-source software tool is presented, which characterises hardware platforms, evaluates gem5 models, identifies sources of error using statistical methods, and applies power and energy analysis. Software, models, datasets and full results are made available.

6.2 Methodology Overview

The existing gem5 model of a Hardkernel ODROID-XU3 (included in the gem5 simulation framework itself) was first validated against a hardware platform. Significant errors both in the estimated execution time and estimated micro-architectural events were identified and this motivated the development of a methodology for identifying sources of error in the gem5 model (Section 6.4). Existing techniques of identifying model errors were first attempted and their shortcomings demonstrated (Section 6.4.1). The proposed methodology, which builds on findings and techniques developed in Chapter 4, does not require detailed knowledge of the CPU being modelled or require the hardware PMC events to be directly mapped to the modelled events. Once the errors in the existing model were understood, the power models developed in Chapter 4 were adapted and optimised for use within gem5 and the effects of gem5 model errors on the modelled power and energy were evaluated (Section 6.5).

Many use cases of a performance simulator involve changing the DVFS level or HMP core type. The models must therefore accurately capture the performance and power scaling between DVFS levels and between different microarchitectures. The modelled performance, power and energy scaling as the DVFS level and HMP core types are changed was compared to that measured on the hardware platform (Section 6.6). As well as simply considering the average scaling across all of the workloads, individual workload clusters were considered.

The identified sources of error were then evaluated in more detail (Section 6.7) and the effect of addressing the key identified source of error on the performance, power and energy evaluated.

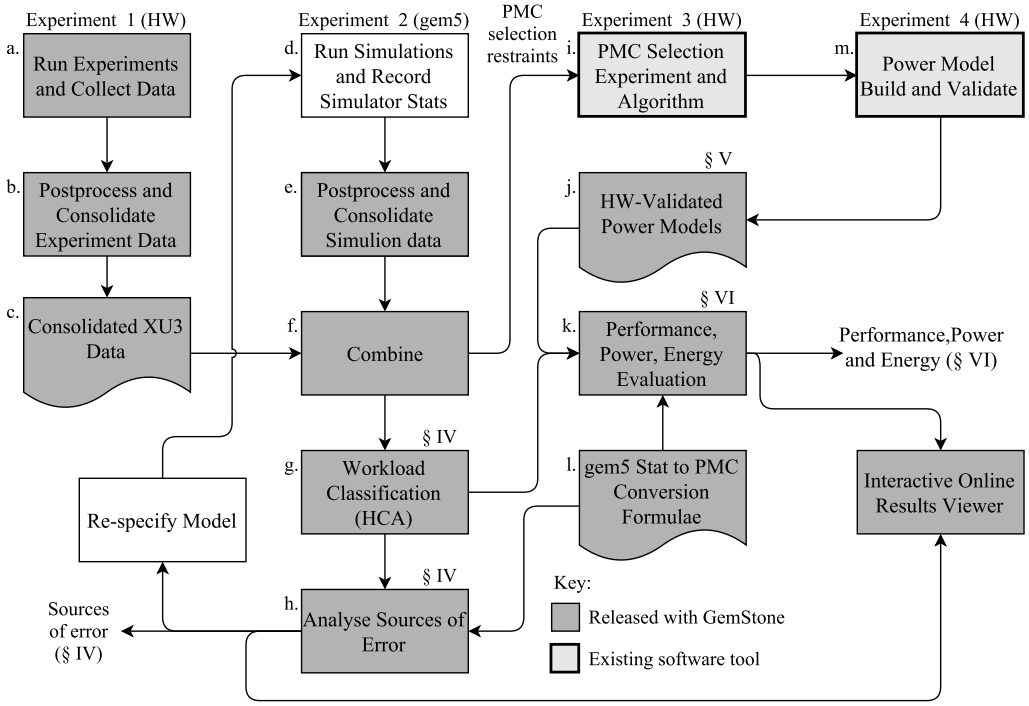


Figure 6.2: Experimental setup and methodology overview

The full experimental methodology, including the running of experiments and collecting data on both the hardware platform and simulation framework; evaluating the gem5 model; identifying the sources of error in the gem5 model; applying the empirical power models to the hardware and simulation data; exporting power equations for run-time analysis in gem5; and evaluating the performance, power and energy across clustered groups of workloads, DVFS levels and HMP core types was automated in the GemStone software tool.

While the methodology is demonstrated on the gem5 simulation framework with mobile CPUs implementing the ARM architecture, the methodology itself is applicable to any simulation framework or CPU architecture.

6.3 Experimental Setup and Workflow

The methodology presented in this chapter is demonstrated on the same Hardkernel ODROID-XU3 development board used in Chapters 4 and 5 as well as [48]. It contains a quad-core Arm Cortex-A7 CPU cluster (optimised towards low-energy), and a quad-core Arm Cortex-A15 CPU cluster (optimised towards high-performance), and is described in detail in Section 2.4.

A set of 65 workloads from several benchmarking suites were used to evaluate the gem5 models and empirical power models, including MiBench [116], ParMiBench [132], LMbench [190], Roy Longbottom's PC Benchmark Collection [179], PARSEC [29], Dhrystone [11] and Whetstone [72]. PARSEC workloads were run both with a single thread and four threads. The LMbench and Roy Longbottom benchmarking suites were not included in the workloads used for execution time error characterisation.

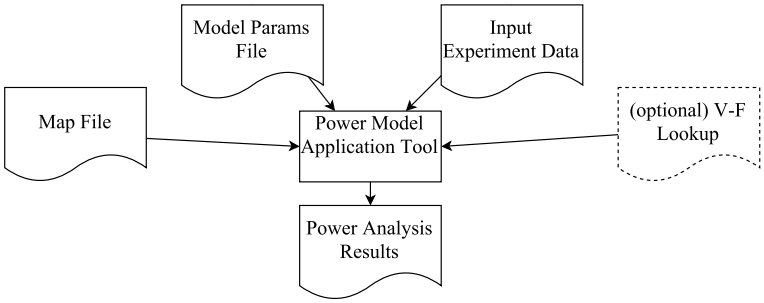


Figure 6.3: Software tool for applying power models to either hardware collected data or gem5 data

The overall experimental setup comprises of four key experiments for data collection (Figure 6.2): 1. collection of performance and PMC data from the hardware (HW) platform for a baseline to compare the gem5 models against; 2. running of the gem5 model simulations; 3. collecting power and PMC data for every workload and PMC event (for power model PMC event selection); and 4. collecting power and PMC data (across the events selected in 3) for every frequency to build the empirical power models. Experiments 3 and 4 and the corresponding post-processing and analysis (boxes i and m) are automated by the *Powmon* software tools (Chapter 4). The power sensors on the ODROID-XU3 provide readings at 3.8 Hz (the sensors internally sample at a higher frequency and provide an average). While this rate can be increased, it comes at a cost of overhead. The workloads were therefore repeated so that they exercised the CPU for at least 30 seconds to obtain accurate and repeatable power measurements.

Experiment 1 (box a) collects data for evaluating the gem5 models and has different requirements to Experiments 3 and 4. Single runs of MiBench, ParMiBench, PARSEC (both single threaded and multi-threaded), Dhrystone and Whetstone (45 workloads in total) were run on the ODROID-XU3 board. Each workload was run five times and the observation with the median execution time used. The experiment was repeated to capture 68 PMC events (only a limited set of PMC events can be measured simultaneously) and the standard deviation of workload execution time between the runs was checked. When running at 2 GHz on the Cortex-A15 (the maximum Cortex-A15 clock frequency), throttling occurred due to high CPU temperatures. A frequency of 1.8 GHz was therefore the highest used and a 5 second delay (where the CPU was idle) was inserted between workloads to allow the CPU to cool down. This experiment was run at 200 MHz, 600 MHz, 1 GHz and 1.4 GHz on the Cortex-A7 and 600 MHz, 1 GHz, 1.4 GHz and 1.8 GHz on the Cortex-A15. Throughout this chapter the ODROID-XU3 hardware platform will be referred to as HW.

The *ex5_LITTLE.py* and *ex5_big.py* CPU models built into the gem5 simulator were used in this work for Experiment 2. They are designed to represent the Exynos-5422 SoC found in the ODROID-XU3 board and are based on the work presented by Butko *et al.* [48] (Section 2.7.3). The simulator was running Ubuntu 11.04 (kernel: 4.4). The simulations were run with the same workloads and DVFS levels as Experiment 1. This gem5 model will be referred to as the *model* throughout this chapter for simplicity.

The results from the gem5 experiments and the gem5 hardware validation experiments were collated and combined (box f), and the workloads clustered to identify patterns and errors between workload types (box g). Knowledge of PMC events that are not available or reliable were fed back to the PMC event selection algorithm so power models can be formulated with events that work well in gem5. Equivalent gem5 events were found to the PMC events chosen in the model (box l) and a

software tool is presented to apply the power model to both the hardware collected data and the gem5 simulated data for power and energy analysis (part of box k). This software tool (Figure 6.3) is compatible with the *Powmon* model building software [309], allowing the models created by that software to be applied to gem5 simulations or HW data. The advantage of this tool is that power models can be applied to gem5 results after the simulation, meaning that the selected power model or the voltage for a selected frequency can be changed without re-running the gem5 simulation. Note that the software tool can also export equations that can be inserted into the gem5 simulation platform for simulated run-time estimation. The methodology for deriving the sources of error in gem5 (box h) is described in the next section and allows the gem5 models to be iteratively improved to match the hardware platform.

6.4 Identifying Sources of Errors in gem5

This section evaluates the existing gem5 models against the hardware (HW) platform and describes a methodology for identifying sources of error. A key problem in CPU simulation is specification error (Section 2.7.3), where models cannot be specified accurately because of insufficient (publicly available) information about the device being modelled. Moreover, there are difficulties in precisely matching many HW PMCs to gem5 events as the hardware documentation is not detailed and the specifics of many events are implementation defined [180]. This section applies several statistical methods to analyse relationships between modelling errors and workload types, HW PMC events, and modelled events to identify the sources of error, without requiring detailed CPU specifications or matched events.

For workloads from the PARSEC suite the gem5 model predicts program execution time with a MAPE of 25.5% and a Mean Percentage Error (MPE) of -7.5% across both CPU clusters and at all tested DVFS levels. A negative MPE indicates that the gem5 model underestimates performance (overestimates the execution time). However, when testing on a larger set of workloads (45 in total) from different benchmarking suites, the MAPE is 40% and a MPE is -21%, highlighting the importance of considering many diverse workloads. The Cortex-A7 model achieves a higher accuracy and tends to underestimate execution time (MAPE and MPE at 1 GHz of 20% and 8.5%, respectively) while the Cortex-A15 model significantly overestimates execution time (MAPE and MPE at 1 GHz of 59% and -51%, respectively).

The workload errors have a similar pattern across all frequencies tested, and the MPE on both the Cortex-A7 and Cortex-A15 becomes gradually more positive with frequency. The workload *par-basicmath-rad2deg* has the highest MAPE of 285% at 600 MHz (264% at 1.8 GHz and 268% at 1 GHz, it has an MAPE of 69% at 600 MHz on the Cortex-A7).

Hierarchical Cluster Analysis (HCA, see Section 4.3.4) was used to group workloads of similar behaviour together, using the inter-correlation between the measured hardware PMCs from the HW platform. The clustering in the resulting dendrogram concurs with intuition of the types of workloads (Figure 6.4). The measured execution time MPE between the gem5 model and the hardware platform was combined with the workload clusters (Figure 6.5) and the following observations were made:

1. the MPE varies significantly between different workloads;

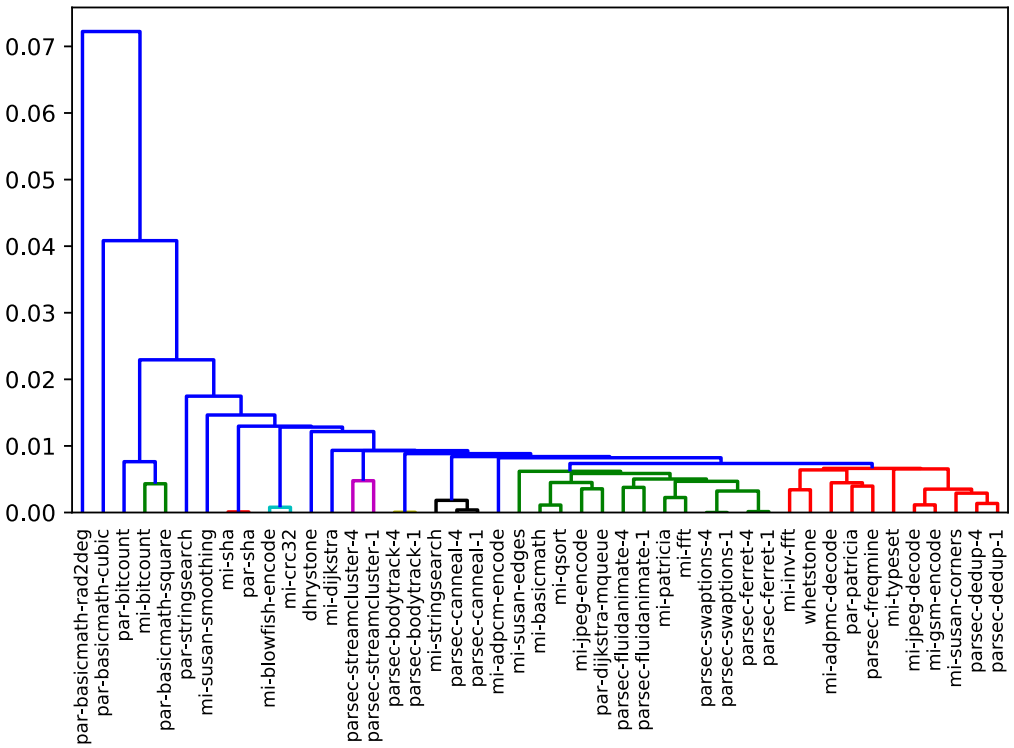


Figure 6.4: Dendrogram of workload clustering. (MiBench prefix: *mi*, ParMiBench prefix: *par*, PARSEC prefix: *parsec*)

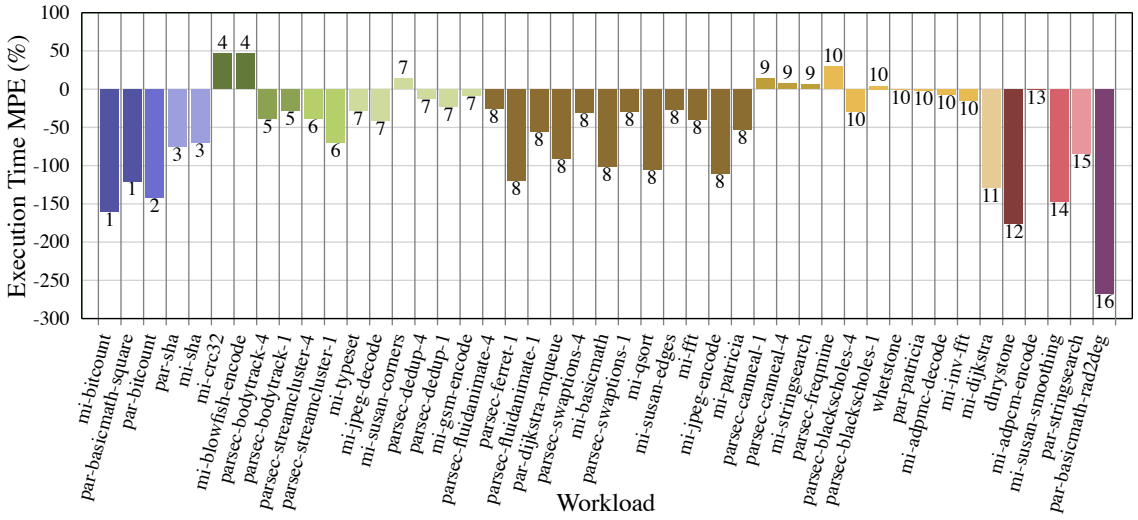


Figure 6.5: Execution time Mean Percentage Error (MPE) for each workload running at 1 GHz on the Cortex-A15 cluster. Workloads are ordered, coloured and labelled (above bars) by cluster designation from Hierarchical Cluster Analysis (HCA) of the hardware PMC correlation. A positive error indicates an overestimation of the performance (underestimation of the execution time).

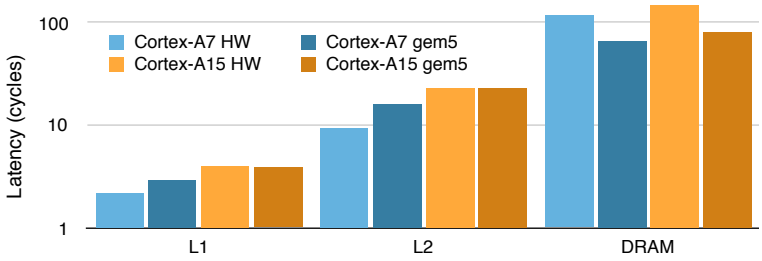


Figure 6.6: Measured memory latency with a stride of 256

2. workloads of the same cluster exhibit similar MPEs (e.g. cluster 4: +47%, cluster 8: -66%, cluster 10: -3%);

3. workloads with significantly large MPEs tend to be in a cluster of their own (as they exhibit specific and repeated micro-architectural behaviour). For example, HCA showed that *par-basicmath-rad2deg* had the largest distance from the other workloads (Figure 6.4), and it also has the largest execution time MPE (Figure 6.5).

Clearly, by understanding the common properties of workloads in a cluster with a high MPE, an intuition of what parts of the model could contain errors can be obtained. Section 6.4.1 looks at the traditional approach of using microbenchmarks to understand model errors while the remainder of Section 6.4 reapplies HCA and other techniques to understand the key sources of error in the model.

6.4.1 Micro-Benchmarks

Micro-benchmarks are often used to identify specific system metrics. The *LMbench* micro-benchmarking suite was run on both the hardware platform and the model and the results were compared. For example, the latency of accessing specific parts of the memory hierarchy was measured using the *lat_mem_rd* benchmark. However, the simulation time to run this was not practical and the modified version of this benchmark developed for Section 5.7 was used to simulate three fixed points on the memory latency graph after being derived and measured on the hardware platform (Figure 6.6). Different stride lengths were used to influence the behaviour of the prefetcher. It was found that the DRAM memory latency was too low in the model and that the Cortex-A7 L2 cache latency was too high, with the other measurements being very close between the gem5 model and HW platform. Memory latency, operation latency and memory bandwidth tests corroborate the tests conducted in [48]. They show several aspects of the model that can be improved (such as DRAM memory latency, as also highlighted in [48]) but these results do not explain the significant negative execution time MPEs (as extra memory latency would push the error in the positive direction).

While micro-benchmarks are well suited for measuring specific micro-architectural metrics, they do not give an idea of where the large sources of error are for different workloads or which sources are most significant. The remainder of this section employs statistical methods to accomplish this, and focusses on the Cortex-A15 to demonstrate the approach. For example, it will be shown how fixing the memory latency problems actually makes the error significantly larger if other sources of error, which are not apparent from the *LMbench* results, are not addressed first. Furthermore, the

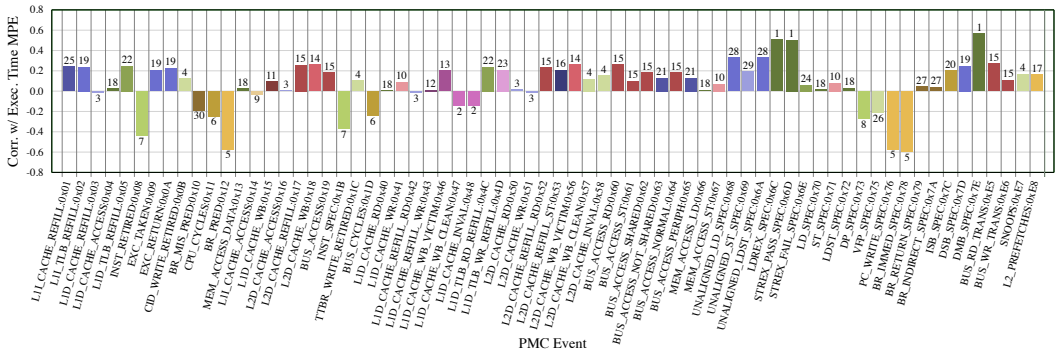


Figure 6.7: Correlation of each HW PMC (rate) with the execution time MPE, labelled and coloured by clusters derived from HCA of the correlation between PMC events across different workloads.

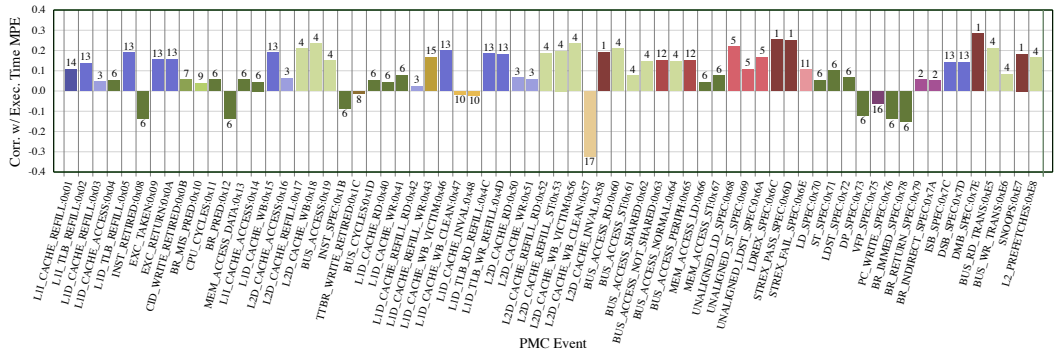


Figure 6.8: Correlation of each HW PMC (total) with the execution time MPE, labelled and coloured by clusters derived from HCA of the correlation between PMC events across different workloads.

analysis conducted later in this section also shows that the optimistic memory latency is partially due to a problem with how memory barriers are modelled.

6.4.2 Cluster and Correlation Analysis (HW PMC Events)

The HCA of the workloads (Figure 6.5) demonstrated how the execution time MPE was closely related to the *type* of workload, as indicated by HW PMC events. In this section, HCA is used to identify clusters of PMC events that correlate with each other across the workloads. This enables groups of PMCs with similar behaviour to be identified and shows the relationships between the events. The correlation between each PMC event and the execution time MPE was calculated and then combined with the HCA to establish how the workload execution time is affected by the PMC event cluster rates (Figure 6.7) and the total number of cluster PMC events over the whole workload (Figure 6.8). A positive correlation means that the execution time of a workload with a high rate of the event in question tends to be underestimated.

The event rates with the largest positive correlation all appear in Cluster 1, which contains events related to memory barriers and exclusive instructions (`0x6C`, `0x6D`, `0x7E`, see Figure 6.7), which occur frequently in concurrent applications, suggesting the cost of inter-process communication could be too low. Cluster 28, which contains events counting unaligned memory accesses, also has a large positive correlation.

As discovered earlier in this section, many of the larger errors are associated with workloads whose

execution time is overestimated. Cluster 5 has the largest negative correlation and contains events related to the rate of branches and control flow operations (`0x12`, `0x76`, `0x78`). However, the rate of branch *mispredictions* (`0x10`) has a negative but notably smaller (in magnitude) correlation. Note that execution time of workloads with a high total number of mispredicted branches on HW tends to be underestimated by the model (Figure 6.8). Cluster 7 (instructions retired and instructions speculatively executed), Cluster 8 (integer data processing speculatively executed) and Cluster 26 (floating point operations speculatively executed) also have notable negative correlations, showing that CPU, integer and floating-point intensive workloads tend to have large negative MPEs. This does not necessarily mean, for example, that the integer operation latencies are incorrect; workloads with many integer operations may also have many branch instructions which could be a source of error. Note that both Cluster 8 and Cluster 26 have a similar negative MPE. The HCA helps identify whether event errors are likely to be from the same source of error or not; the HCA groups these into separate clusters (with a significant distance between them), showing that these are two separate sources of error. Additionally, the correlation between execution time and execution time error was tested and found to be negligible, showing that the duration of a workload does not impact the error.

6.4.3 Cluster and Correlation Analysis (gem5 Events)

The previous section analyses how the error is correlated with HW PMC events. Conducting the same analysis using the estimated gem5 events gives a different insight and is contrasted with the analysis of the previous section to identify differences between the modelled system and the hardware platform. The gem5 simulation outputs thousands of statistics compared to the tens of hardware PMC counters. The analysis scripts automatically identify the same statistics for different CPUs of a cluster and create a new event counter which is the sum of the counts of each CPU. The events with an absolute correlation of over 0.3 were extracted, resulting in a total number of 94 events. The largest cluster in these selected events was *Cluster A*, which was made up of 31 events and had the largest negative correlation with every event in the cluster having a correlation lower than -0.51. The vast majority of the events were related to the ITLB (Instruction Translate Lookaside Buffer, see Section 2.4). Most of the events concerned accesses to the *itb_walker_cache* specifically, which is designed to approximate the L2 ITLB component in the real hardware. Included in these were both hits and misses in the L2 ITLB. There were, however, also events related to the *itb* component (modelling the L1 ITLB) but these events were only related to misses, showing that large negative execution time errors tend to occur when there are many L1 ITLB misses in gem5, and the L2 ITLB is accessed (resulting in a hit or a miss). This could suggest that the latency in the L2 ITLB is too high in the model, or that the source of error is highly correlated to this event. There are several events in *Cluster A* that are not directly related to the ITLB: *iew.exec_nop*, *fetch.TlbCycles*, *iew.predictedTakenIncorrect*, *fetch.PendingTrapStallCycles*, and *branchPred.RASInCorrect*. Note that the fact that events related to branch mispredictions are in the same cluster as events related to L2 ITLB accesses shows that there is a strong relationship between these two components.

Fourteen events with large negative correlations (between -0.46 and -0.31) appear in *Cluster B*. Most of the events are related to predicted and mispredicted branches, e.g.: *commit.branchMispredicts*, *fetch.predictedBranches* and *branchPred.usedRAS* (Return Address Stack).

The next largest cluster is *Cluster C*, the events in which all have a smaller negative correlation of

−0.35 or −0.36. All events in this cluster are related to L1I cache misses. Other gem5 events with negative correlation are related to both the L2 MSHR (Miss Status Holding Registers), uncacheable latency due to CPU data and the L2 overall miss rate.

Forty of the gem5 events have a positive correlation and the largest cluster has three gem5 events, which are related to the fetch rate and the number of instructions committed per cycle. Other gem5 events with a positive correlation relate to the L2 cache writebacks and the L2 cache miss latency, again suggesting the DRAM memory latency is too low.

This disparity between this analysis (analysing the modelled events with execution time error) and the analysis of profiled HW PMCs with execution time error (Section 6.4.2) identifies differences between HW and the model, and also allows an understanding of which events are the source, and which ones are simply correlated with the source. For example, the differences in branches and mispredicted branches between the two analyses suggest a significantly larger misprediction rate in the model; the BP is a potential source. The difference in L1 ITLB misses and the high correlation between modelled branch mispredictions and L2 ITLB accesses suggest that a high L2 ITLB latency is not a key source; though it could exacerbate the errors. There are inconsistencies in the TLB events between the HW and model.

6.4.4 Regression Analysis

Regression analysis was used to approximate the relationship between the hardware PMC events and the gem5 model error. The regression analysis is an important step in the methodology as events with a large correlation are not necessarily the most useful in identifying the sources of error. A forward-selection stepwise approach (see Section 4.3.6), using the R^2 value (see Section 3.1.5) as an optimisation metric, was used to identify which hardware PMC events to use as inputs to the model. Both the total event counts and the rates were made available as candidates to the selection process. The dependent variable was set as the difference between the measured hardware execution time and the estimated gem5 execution time and a frequency of 1 GHz was considered in this analysis. The process adds events to the model until the *p-value* of any of the terms rises above 0.05 (a common rule of thumb is that terms with p-values above 0.05 are not statistically significant).

The model selected seven events and achieved an R^2 and *Adjusted* R^2 (compensating for the number of predictors) of 0.97, showing that a model just using the hardware PMCs can accurately predict the gem5 model execution time error. The single best PMC event to predict the error was PC_WRITE_SPEC (total), showing that the gem5 model error is highly dependent on workloads that have (when executed on HW) the largest number of branches. The regression analysis finds SNOOPS and L1D_CACHE_REFILL_WR to be important in predicting error, despite not being found to be significant in the PMC correlation and cluster analysis. The other events in this selection (which include LDREX_SPEC and BR_RETURN_SPEC) largely corroborates the previous analysis.

The same analysis is conducted using the gem5 event statistics, and eight events were automatically selected, resulting in a model that achieved an R^2 and Adjusted R^2 of 0.99. The eight selected events included *commit.commitNonSpecStalls*, *branchPred.indirectMisses*, *dtb.prefetch faults* and *l2.ReadExReq hits (data)*. The first one is the total number of times the commit unit has had

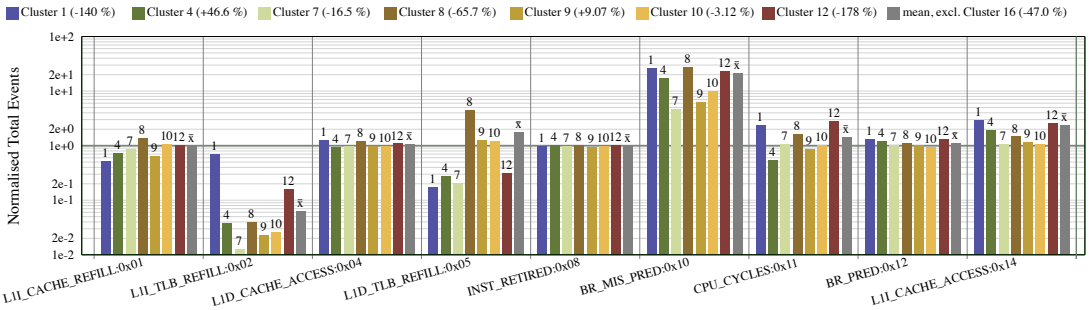


Figure 6.9: Total gem5 events normalised with their HW PMC equivalent (bars over 1 indicate that gem5 overestimates the PMC event). Results shown for selected clusters. The mean bars exclude Cluster 16.

to stall due to a non-speculative instruction reaching the head of the re-order buffer (ROB). The second again highlights branch misprediction.

The regression analysis shows how the gem5 error can be accurately predicted simply from the hardware PMC events. It has also largely reinforced the conclusions from the previous analysis and identified some parts of the system to look at in more detail (e.g. snoops, DTLB prefetch faults, L1D cache writebacks, L2 data hits).

6.4.5 Event Comparison

The previous analysis has shown that the key sources are related to the Branch Predictor (BP) and the ITLB hierarchy, specifically when accessing the L2 ITLB. While many of the PMC events could not be directly mapped to gem5 events due to the reasons described above, some events related to the branch predictor and TLB hierarchy, could be directly equated. Key gem5 events were matched and normalised to their HW PMC equivalents (Figure 6.9). As well as comparing gem5 and PMC events for the mean of all of the workloads, the mean of selected clusters was also observed. As expected, there was a negligible difference in the total number of instructions committed (`0x08`) between HW and the gem5 model. Significantly fewer (0.06×) ILTB misses (`0x02`) occurred in the model and they are very workload dependent (*Cluster 1* has 0.7×, while *Cluster 7* has 0.01×, Figure 6.9). However, the gem5 model predicts 1.7× L1 DTLB misses. The model has 1.1× predicted branches (`0x12`) and this is relatively consistent between clusters (Cluster 1 has 1.32× while Cluster 10 has 0.93×). However, the gem5 model has a significantly higher number (mean: 21×) of branch mispredictions (`0x10`). In fact, for Cluster 16 (not shown in Figure 6.9) the model has 1402× branch mispredictions than HW. Interestingly, while the MPE is correlated highly with modelled L2 ITLB accesses, these occur less often in the model; branch mispredictions occur more often.

In HW, the BP has a mean prediction accuracy of 96% while in the gem5 model it is 65%. The lowest prediction accuracy in the gem5 model is 0.86%, which occurs for the same workload (*par-basicmath-rad2deg*, Cluster 16) that achieves the highest prediction accuracy in HW (99.9%). This workload has an execution time MPE of −268% (at 1 GHz).

The number of active CPU cycles (`0x11`, Figure 6.9) closely follows the per-cluster errors (Figure 6.5). On average, the gem5 model only executes 1.1× more instructions speculatively than HW, meaning that the overestimated execution time is largely due to stalled cycles (this was also

identified in Section 6.4.3 with the *fetch.PendingTrapStallCycles* and *iew.exec_nop* events being in the same cluster as the L2 ITLB accesses that was highly correlated with MPE, as well as in Section 6.4.4 where the *commitNonSpecStalls* event was selected) and only partially due to more instructions being speculatively executed. This does vary with cluster though, with clusters 4, 10 and 12 having 1.3×, 0.9× and 1.4×, respectively. Despite this, there are over 2× more L1I cache accesses in the model, assumed to be due to gem5 accessing it for every instruction, as opposed to decoding the entire cache line, but this is a topic for further investigation, and is not identified as being a key source of error. Other events that diverge significantly are the L1D_REFILL_WR:`0x43` (9.9×) and L1D_WB:`0x15` (19×). The number of L2 prefetches are also significantly overestimated by the gem5 model.

This section has concurred with findings from the previous sections that did not rely on matched PMC and gem5 events, and quantified ITLB misses and BP performance metrics.

6.4.6 Summary

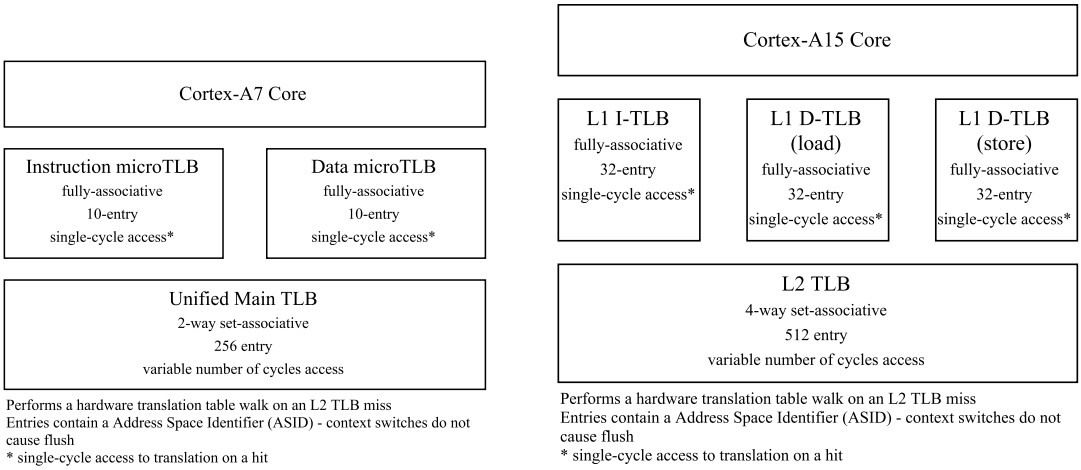
This section has presented several methods that together evaluate the gem5 model performance and identify sources of error without requiring detailed CPU information or direct equivalents between the HW and modelled events.

While microbenchmarks found the modelled DRAM memory latency to be too low and discrepancies in the operation latencies (Section 6.4.1), statistical analysis techniques found the larger sources of error in the BP accuracy and TLB. Other sources of error were found to be related to the L1I cache misses, operations for concurrent programs (memory barriers and snoops), L1D cache writebacks, the L2 cache miss rate and the L2 prefetcher.

It was found that the performance loss in the gem5 model was due to stalled cycles, rather than executing too many incorrect instructions speculatively. This was identified in both Section 6.4.3 and Section 6.4.4, and was confirmed when directly comparing the number of instructions speculatively executed in Section 6.4.5.

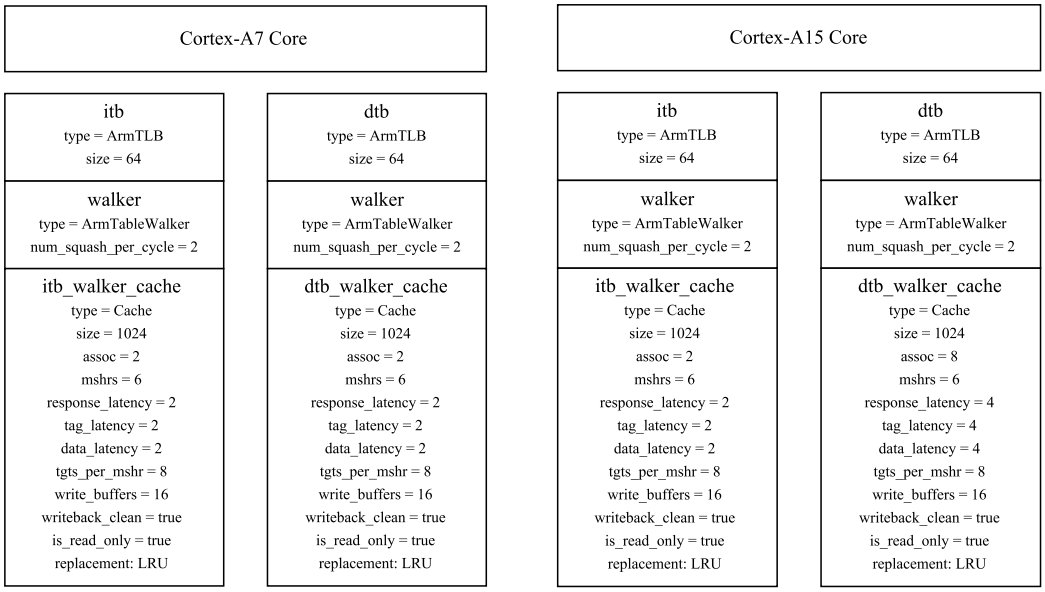
In the gem5 cluster and correlation analysis it was found that events related to the ITLB were most highly correlated with error. However, in the same cluster were branch mispredictions, showing a strong dependency between branch mispredictions and L2 ITLB accesses. The disparity in branches predicted and branches mispredicted between the HW cluster and correlation analysis, and the gem5 cluster and correlation analysis showed the significant difference in branch prediction accuracy of the HW and model. Importantly, while the error was highly correlated with HW branch predictions, it was *not* highly correlated with hardware mispredictions or L1 ITLB misses. This points to the branch predictor as being the key source of the error, while there is clearly also a large difference in ITLB behaviour. From the direct PMC comparison, it was seen that there are many more branch mispredictions in the gem5 model than in the hardware and there were actually, despite their large correlation with error, *fewer* ITLB misses in the gem5 model than in the hardware.

The TLB discrepancies found by the automated methodology can be explained with the CPU's documentation [13], which shows that the TLB hierarchy in the hardware differs significantly from the one specified in the model (Figure 6.10). The Cortex-A15 has a shared 512-entry 4-way set-associative L2 TLB whereas the model has two separate 1 KB 8-way set-associative caches simulating



(a) Cortex-A7 HW TLB Hierarchy

(b) Cortex-A15 HW TLB Hierarchy



(c) Cortex-A7 gem5 TLB Hierarchy

(d) Cortex-A15 gem5 TLB Hierarchy

Figure 6.10: Comparison of the Translate Lookaside Buffer (TLB) Hierarchy between HW and the gem5 for both the Arm Cortex-A7 and Arm Cortex-A15 CPUs in the Samsung Exynos-5422

the L2 TLBs (one for instructions, one for data). The latency of these caches (4 cycles) appears high compared to the Cortex-A7 L2 TLBs (also 1 KB, latency of 2 cycles, albeit it is 4-way set associative) as well as the L1D cache (2 cycles, 32 KB, 4-way set-associative). Additionally, as they are not unified they will have a lower combined hit ratio than a single TLB of double the size. The low number of simulated L1 ITLB misses results from a 64-entry L1 ITLB being specified when HW has a 32-entry one (also highlighted in [117]). However, changing this to the correct value results in a significantly larger MAPE, as expected, due to other errors present in the gem5 model.

Other, less significant, sources of error were also identified, including with integer and floating point operations. HCA identified that these two errors were from different sources, which was confirmed when it was later found that the error correlation with floating point operation was due to the gem5 model classifying all floating point operations as SIMD operations. Other identified sources of error related to the memory latency, inter-process memory communication, and the instruction cache.

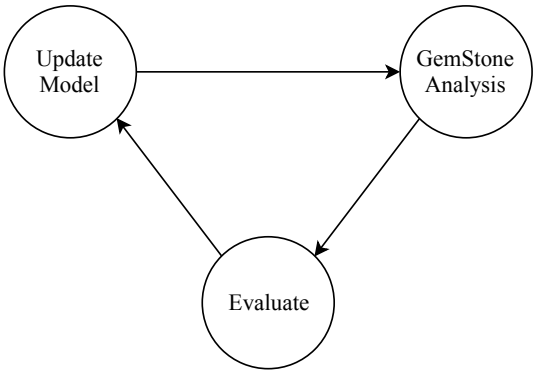


Figure 6.11: Iterative model improvement

There is interaction between the components of the model and changes to each part of the system have knock-on effects. It is therefore important to work on each component individually, and evaluate the full system after each change. It is also necessary to address the most significant sources of error first, otherwise changes to other parts of the system may not show a representative difference. Once the key sources of error have been identified, they can be addressed. This may require further investigation, for example, the targeted use of specific microbenchmarks or through adjustments of model parameters in a trial-and-error fashion (the effect of the changes can clearly be visualised). The process is then repeated in an iterative fashion, addressing the most significant source of error each time until the model converges to an acceptable accuracy for the use case (Figure 6.11). GemStone automates the process of re-running gem5 experiments, comparing with the HW data, and re-analysing the sources of error to make the iterative approach of improving gem5 models straightforward. Furthermore, as well as enabling improvements to the model, the output from GemStone allows the errors, and their potential impact on the specific model use case, to be understood. While many of the graphs and stages of analysis generated by GemStone have been omitted for brevity, the key types of analysis and uses of them have been demonstrated. The key source of error was identified to be the BP and Section 6.7 discusses improvements that have since been made to it.

6.5 Power Modelling

This section presents empirical power models designed to use the output statistics from gem5 to calculate the power consumption. The PMC-based model building methodology and corresponding *Powmon* software presented in Chapter 4 was used, which automates the two key stages of the methodology: PMC event selection and model formulation. The models were first developed and validated on HW using PMC events before being integrated into gem5 using modelled events. The models were therefore developed using PMC events found to have accurate and reliable gem5 equivalents.

Chapter 4 demonstrated that the produced models are robust and maintain their accuracy even with unforeseen CPU workload patterns. The first step was to verify the effectiveness of the existing models by using the existing model coefficients with the data collected from this experiment, which uses workloads that were not considered in Chapter 4 (PARSEC and ParMiBench). A MAPE of 5.6% was achieved, larger than the quoted 2.8%. However, there are several reasons for this: the

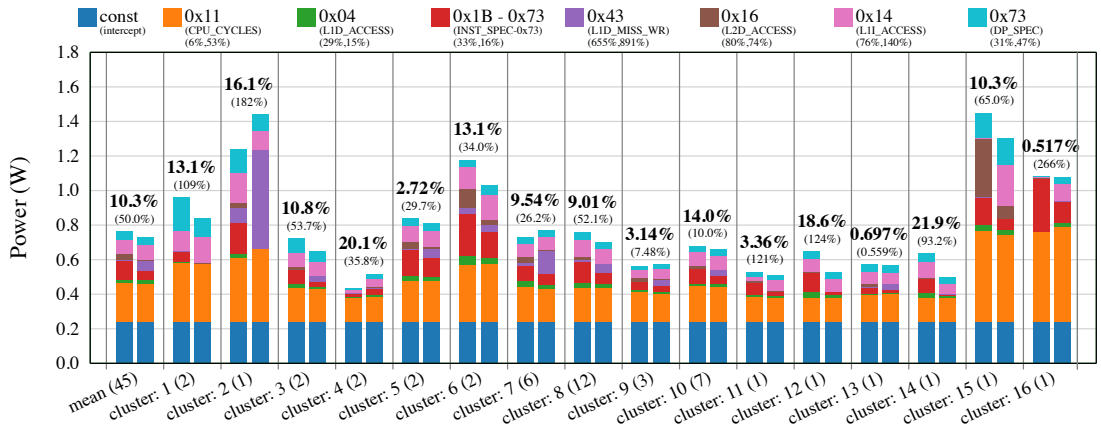


Figure 6.12: Comparison of estimated power between using the HW PMC events (left bar) and the gem5 events (right bar) for each cluster (number of workloads in cluster in brackets in X-Axis labels) as per Figure 6.5 (Cortex-A15 CPU). The power MAPE is above each bar pair in bold, with the energy MAPE below it in brackets. The bars are colour-coded to show the power contribution from each model component. $f_{clk} = 1$ GHz

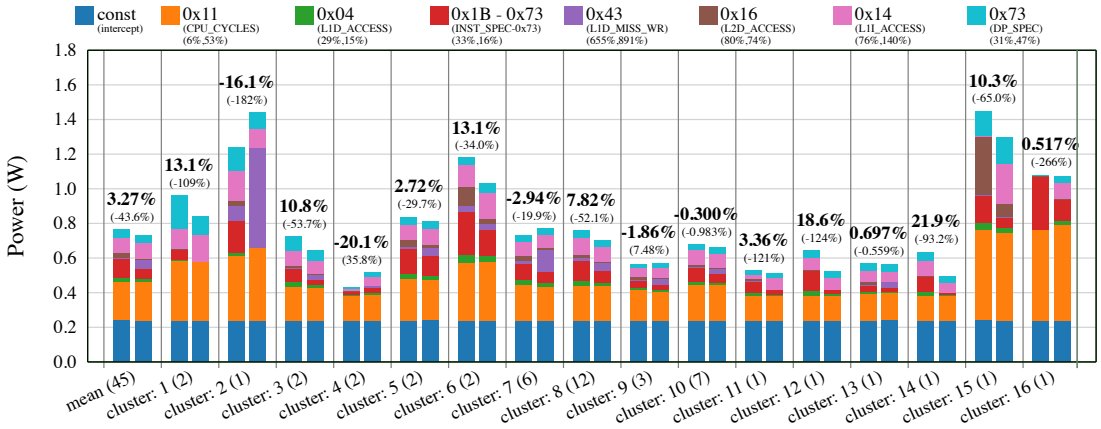


Figure 6.13: The same graph as shown in Figure 6.12 but with the Power MPEs (bold typeface) and energy MPEs show above the bars.

board is not identical and components such as the SoC, power sensors and voltage regulators are subject to variation; the model can be affected by differences in storage media (the read write speeds of the SD card/eMMC); and the ambient temperature conditions have a large effect on the leakage power, as modelled in Chapter 5. The coefficients were re-trained using the same PMC event selection from Chapter 4 and the data collected from the HW platform. A MAPE of 2.8% was achieved, corroborating the observation that the PMC event selection is effective on workload sets that were not used in the selection process.

However, there were some PMCs in the original selection that were not readily available in the gem5 statistics (e.g. unaligned memory accesses) or that were found to be particularly inaccurate. The PMC event selection experiment and algorithm (from Chapter 4) was run with no restrictions on which PMC events could be chosen to obtain a baseline model. Because this experiment uses different workloads to Chapter 4 the PMC event selection differed. The new model achieved a slightly higher MAPE of 4%, but the R^2 value, which measures the *goodness-of-fit* and is the metric that the model building process is optimising for, is improved, as expected. However, there were some issues with the chosen PMC events: `0x15` (L1 data cache writebacks) had an MPE of over

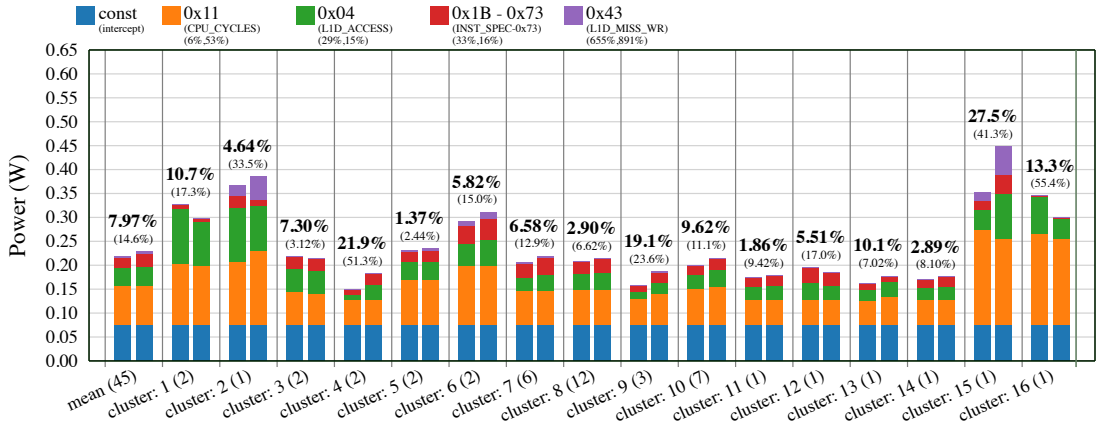


Figure 6.14: Comparison of estimated power between using the HW PMC events (left bar) and the gem5 events (right bar) for each cluster (number of workloads in cluster in brackets in X-Axis labels) as per Figure 6.5 (Cortex-A7 CPU). The power MAPE is above each bar pair in bold, with the energy MAPE below it in brackets. The bars are colour-coded to show the power contribution from each model component. $f_{clk} = 1$ GHz

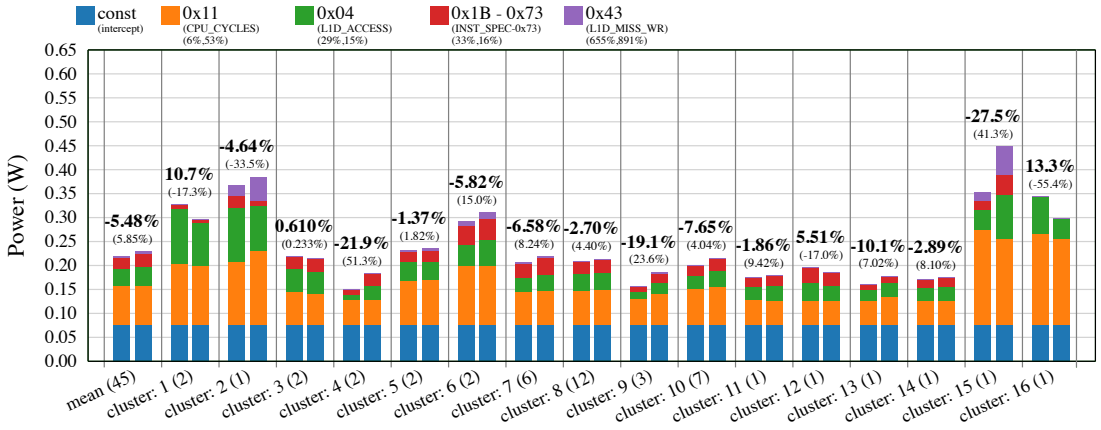


Figure 6.15: The same graph as shown in Figure 6.15 but with the Power MPEs (bold typeface) and energy MPEs show above the bars.

1000% for both the total and rate; `0x75`, (floating-point speculatively executed) were misclassified in the gem5 model as SIMD floating point instructions. The approach used was to remove PMCs from the selection pool if it was not readily available in gem5 or if it had a significant error and there was an alternative event available. A trade-off had to be made as selecting only events that were modelled well in gem5 resulted in selecting events with a large inter-correlation (resulting in a poor model). The chosen events for the Cortex-A15 model are shown in the legend of Figure 6.12. Event `0x1B` has `0x73` subtracted from it to reduce multicollinearity. The model was built using all 65 workloads (Section 6.3) and validated against the HW platform. It achieved a MAPE of 3.28%, standard error of regression (SER) of 0.049 W, and Adjusted R^2 of 0.996. The mean Variance Inflation Factor (VIF) across all model inputs was 6, indicating a low level of inter-correlation, as required. The *p-values* for all coefficients were lower than 0.0001 with the exception of two, which were lower than 0.02. Substituting PMC events with ones that were readily available and modelled with reasonable accuracy caused some degradation of the model but its accuracy and VIF is still within an acceptable level. The maximum MAPE out of all 621 observations was 14% (*parsec-canneal-4* at 1400 MHz).

The Cortex-A7 model achieves an adjusted R^2, MAPE and SER of 0.992, 6.64%, and 0.014 W, respectively.

This section has built power models for the Cortex-A7 and Cortex-A15 clusters using PMC events suitable for the gem5 models and validated them using power measurements from the hardware platform. While the model parameters are omitted from this chapter for brevity, all the parameters, an extended set of model quality statistics as well as software implementing them are made available.

6.5.1 Alternative Approach

An alternative approach is to run identical workloads on both HW and the gem5 simulator, take the measured workload power from hardware and add it to the gem5 data, and build the models using the gem5 statistics instead of the hardware PMCs. This is a poor approach for several reasons. Unlike the hardware PMCs, there are (significant) errors in the gem5 statistics, meaning that the model is formulated with erroneous data but fitted to the accurate HW power value (it is effectively using PMC data from one CPU to model the power of another).

Furthermore any improvements to the gem5 model result in an unpredictable (and incorrect) change to the estimated power and energy consumption. The key goal of the stable model building method is to reduce coefficient errors. By training the model with erroneous inputs (with respect to the power) from the outset, it is impossible to reduce coefficient errors. Testing this with the same model building software resulted in a poor model with a low perceived error of 12%, but when employing cross-fold validation (testing on workloads that are not in the training set) resulted in an error of 104%. Coefficient p-values were greater than 0.2.

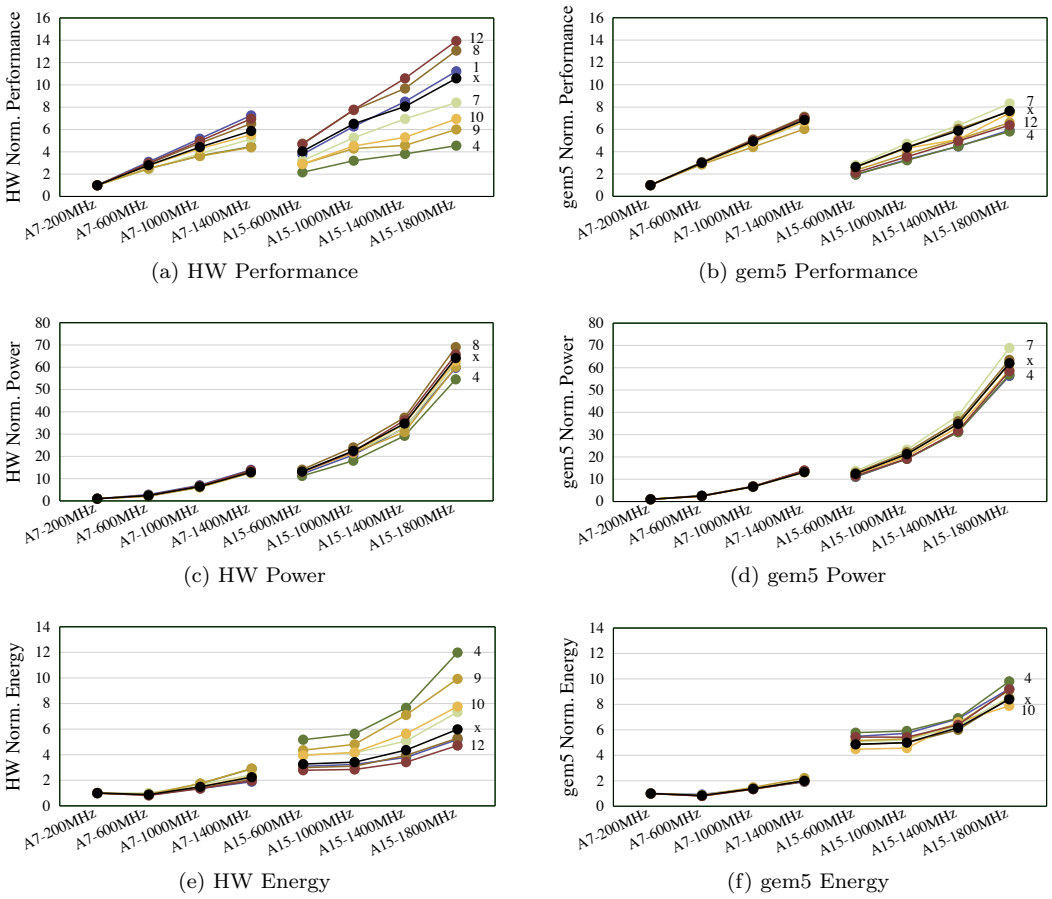
6.6 Performance, Power and Energy Evaluation

This section combines the gem5 model analysed in Section 6.4 with the hardware-validated power models presented in Section 6.5 and evaluates the effect of gem5 modelling errors on the resulting power and energy estimations. The power model application software tool (Section 6.3, Figure 6.3) applies the same power model to PMCs collected from HW and gem5 modelled event statistics. The resulting power and energy is then compared. The gem5 estimated power is not compared to the hardware measured power for two reasons:

1. the power sensors do not provide accurate power readings for short workload durations and repeated workloads behave differently to single runs;

2. the measured power depends heavily on the temperature and voltage conditions which are not modelled in gem5.

Therefore, a fair comparison is achieved by using the equivalent PMC events and modelled gem5 events with the same model, with the same voltage-frequency lookup values. The power model uses the event *rate* to estimate the power consumption, as opposed to the total number of events. The chosen events and their MAPEs are shown in the legend of Figure 6.12 (first number is the rate MAPE, the second is the total MAPE).



(a) HW Performance (b) gem5 Performance

(c) HW Power (d) gem5 Power

(e) HW Energy (f) gem5 Energy

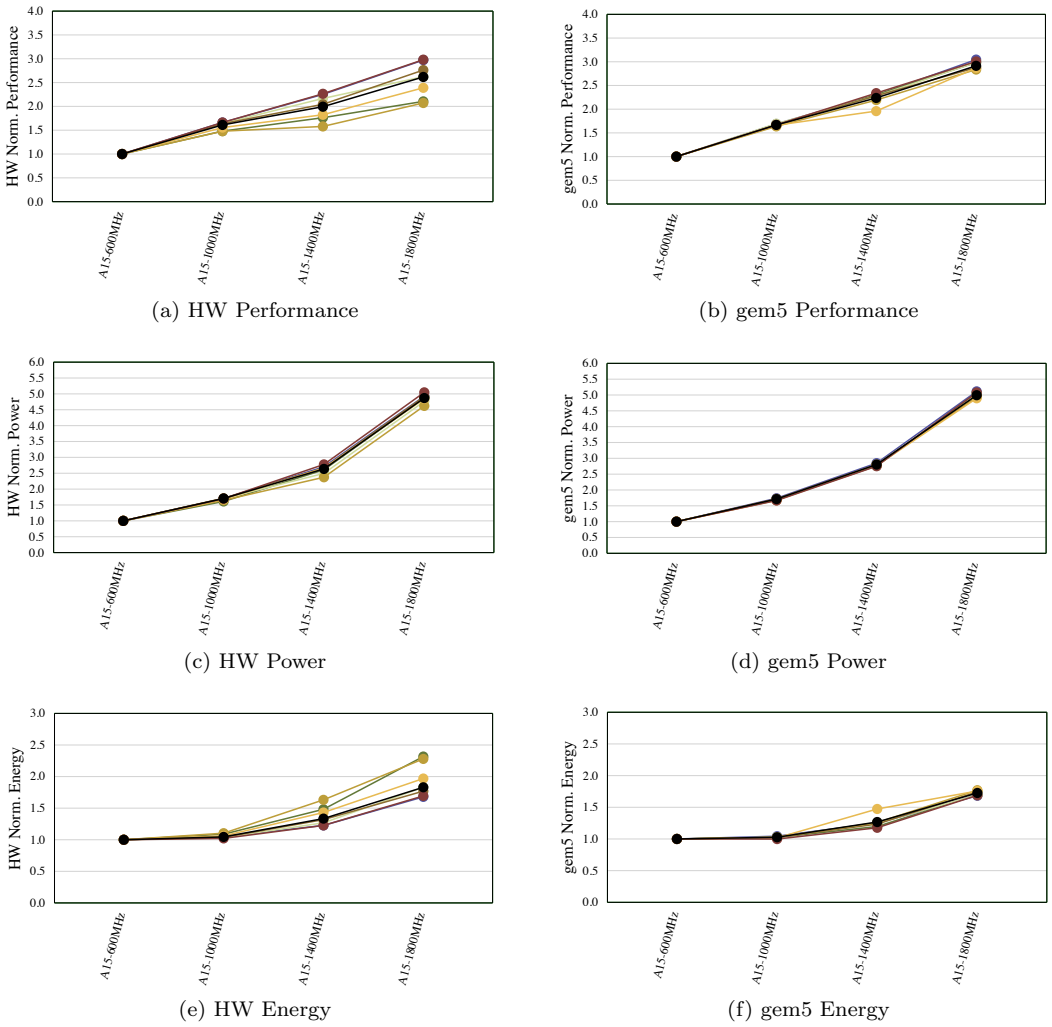
Figure 6.16: Performance, power and energy scaling normalised to 200 MHz on the Cortex-A7 CPU. Cluster numbers and colours correspond to Figure 6.5. Line `x` (black line) is the mean of all clusters (excluding cluster 16).

The estimated power from the HW PMC events and the gem5 modelled events was compared for each cluster (Figure 6.12). Despite large errors in the gem5 modelled events, the mean power MPE and MAPE for the 45 workloads are 3.3% and 10%, respectively. The reason for this is clear when observing each predicted power component; the largest components are the intercept (static and constant dynamic power) and `0x11` rate (which has a low MPE). Many of the other components cancel each other out. For example, in Cluster 13, `0x43` in the gem5 model is 9.7× larger than the measured HW equivalent. However, the error in this component is offset by `0x1B-0x73` and `0x16` being 2× and 6.7× larger, respectively, and a power MAPE of just 0.7% is calculated.

While the power error is low, the energy errors, which are dependent on the estimated execution time, are significantly larger. The energy MPE and MAPE are -43.6%, and 50.0%, respectively. The energy MAPE of each cluster varies significantly; from as low as 0.6% (Cluster 13) to as high as 266% (Cluster 16) (Figure 6.12 [energy MAPE in brackets below the power MAPE]). Moreover, a cluster can have a very low power error but a very large energy error.

The Cortex-A7 model achieves a power MPE and MAPE of -5.48%, 7.97%, respectively, and an energy MPE and MAPE of 5.85%, and 14.6%, respectively. The Cortex-A7 model achieves lower power and energy errors due to the higher accuracy of the Cortex-A7 gem5 model.

The trade-offs between DVFS levels and different cores (e.g. in an Arm big.LITTLE [13] system) are



(a) HW Performance (b) gem5 Performance

(c) HW Power (d) gem5 Power

(e) HW Energy (f) gem5 Energy

Figure 6.17: Performance, power and energy scaling normalised to 600 MHz on the Cortex-A15 CPU. Cluster numbers and colours correspond to Figure 6.5. Line `x` (black line) is the mean of all clusters (excluding cluster 16).

important for many investigations. The performance, power and energy, normalised to the lowest frequency (200 MHz) of the Cortex-A7 cluster was calculated to see how the scaling of the gem5 model compared with HW (Figure 6.16). Selected clusters from the HCA were also considered to see how different workload types scaled. A key observation is that the modelled Cortex-A15 performance is lower, with respect to the Cortex-A7, than measured from HW.

When considering only the Cortex-A15 scaling, the mean speedup running at 1800 MHz compared to 600 MHz is 2.7× and 2.9× for HW and the model, respectively, showing that the model accurately estimates the mean speedup. However, the model does not capture the workload diversity; the speedup range is 2.1× to 3.2× for HW and 2.8× to 3.0× for the model. The minimum speedup is Cluster 9 in both cases, but the maximum speedup is Cluster 2 for HW and Cluster 11 for the model. The energy increase estimated on HW has a range of 1.7× to 2.3× (mean: 1.8×), while the model estimates a range of 1.6× to 1.9× (mean: 1.7×). This would need to be considered for studies that consider the scaling of frequencies, or trading off between the 'little' and the 'big' CPU.

6.7 Improvements to the gem5 Models

Its active development community means that gem5 is constantly being updated and improved. Since the analysis in Section 6.4 identified significant errors in the BP, a change (bug fix) has been made to the BP used in the *ex_big* (Cortex-A15) CPU model. GemStone allows the latest version of gem5 to be pulled, and the experiments and analysis repeated automatically with less than 15 minutes of user time required. Once the gem5 simulations have completed and the analysis run, the results can be compared directly, side-by-side. Running GemStone with the new model results in a significantly improved MAPE and MPE of 18% and +10%, respectively, meaning that the new gem5 model underestimates execution time on average. The energy MAPE improved from 50% to 18%.

The individual workload MPEs have improved significantly, particularly those workloads in clusters 16, 14, 12, and 11 (Figure 6.19, when compared to Figure 6.18). On closer inspection, it can be seen that, while the branch predictor fix improved the workloads that originally had a negative MPE (Figure 6.20), the workloads with a positive MPE still have large positive MPEs (Cluster 4) while workloads that previously had a small positive or negative MPE now have a large positive MPE (clusters 7, 2, 9 and 10). Note that some workloads have a different classification in the more recent experiment due to the shown workloads being part of a larger set being run and analysed.

Running the same cluster and correlation analysis on the HW (as described in Section 6.21) with the new model shows that events related to memory barriers and inter-process communication now have the highest correlation with MPE. Events related to the L1D cache have the largest negative correlation with the MPE. HW events related to branch predictions still have a large (though not quite as large) negative correlation with MPE. However, since the fix, there is a larger negative correlation between the mispredicted branches and the MPE. Furthermore, branch predictions and mispredictions are in the same event clusters (29). The branch prediction performance has improved and it is no longer the key source of error. The measured HCA distance between event cluster 29 (branch prediction and misprediction) and clusters 32 (VPF), 33 (DP), and 31 (instructions speculatively executed and instructions retired) has reduced. There are large and negative errors in workloads that tend to have high rates of instructions, control flow events, and floating point operations. The workload with the largest MPE is the only workload in Cluster 15 (*par-basicmath-cubic*, note that cluster 15 contains a different workload to that in cluster 15 of the model before the branch predictor fix). From looking at the power model comparison after the branch predictor fix (Figure 6.23), it can be see that cluster 15 has a much higher rate L1I accesses and integer data processing instructions, while simultaneously having a much lower rate of instructions speculatively executed, when compared with HW. There are a higher number of instruction cache accesses in the gem5 model, which was already identified in the previous analysis. After the branch predictor was fixed, this became a more dominant source of error.

The events with the largest negative correlation with MPE in the gem5 cluster and correlation analysis (described in Section 6.4.2), after the branch predictor fix, were *commit.op_class_0::SimdFloatMisc* and *iq.FU_type_0::SimdFloatMisc* (correlation of -0.48). This is related to the earlier identified issue of floating point operations being misclassified as SIMD instructions, which requires more L1 cache accesses in the gem5 model. From looking at the direct PMC event comparison (Figure 6.22), it can be seen that, while for some clusters the number of L1I cache accesses was much higher, for others it was lower (as expected from the previous observation). Clusters 1 and 11 have a slightly

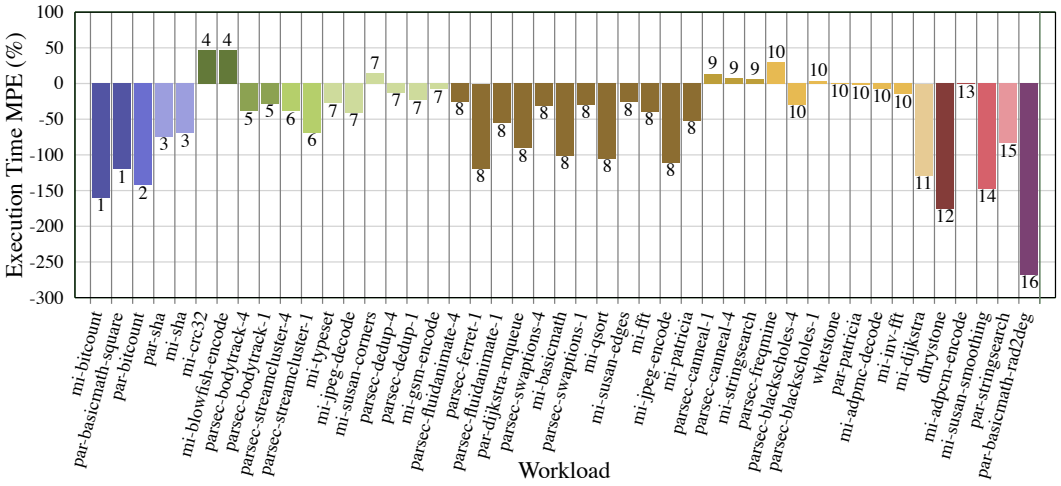


Figure 6.18: Execution time MPE for each workload running at 1 GHz on the Cortex-A15 (grouped and labelled by workload cluster allocation) before branch predictor modification (reproduction of Figure 6.5 for comparision purposes)

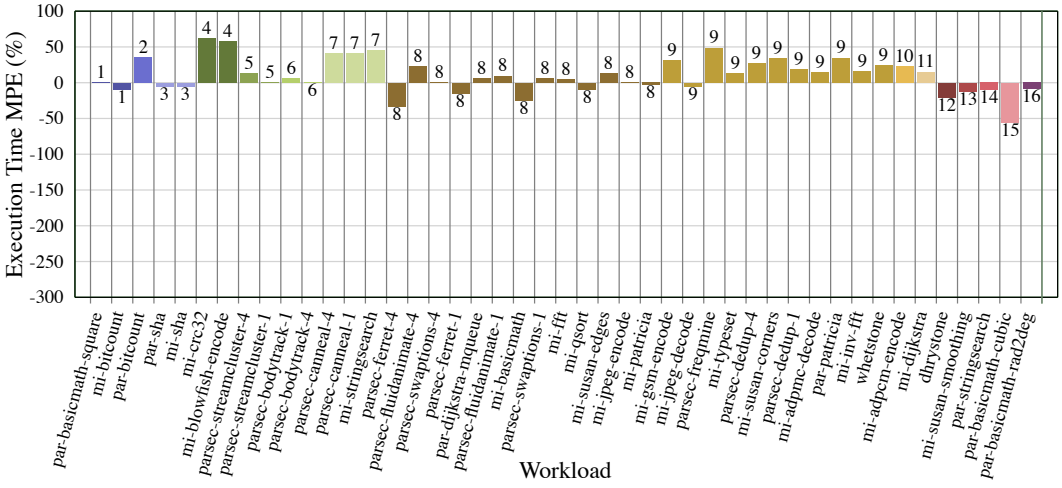


Figure 6.19: Equivalent to Figure 6.18 but after branch predictor modification

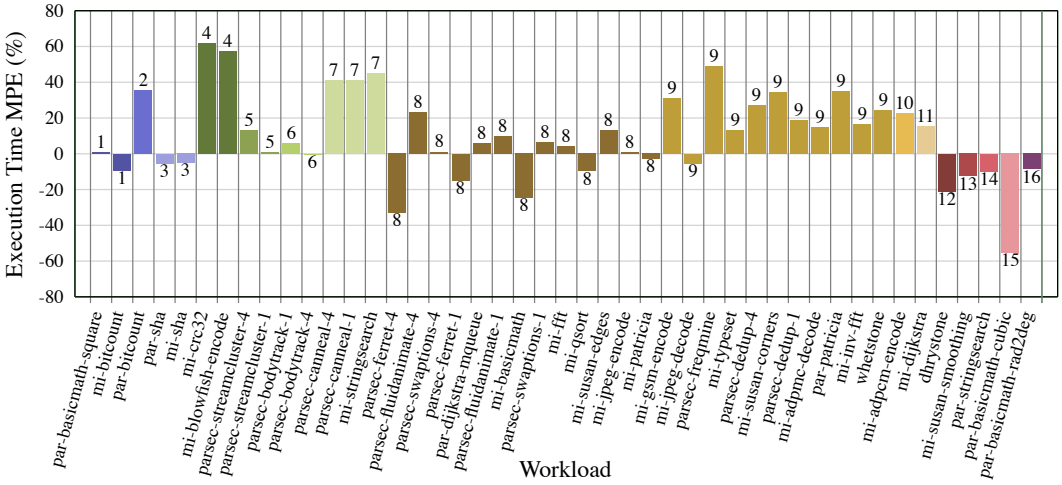


Figure 6.20: Equivalent to Figure 6.19 but with adjusted axis scaling for closer inspection of MPE

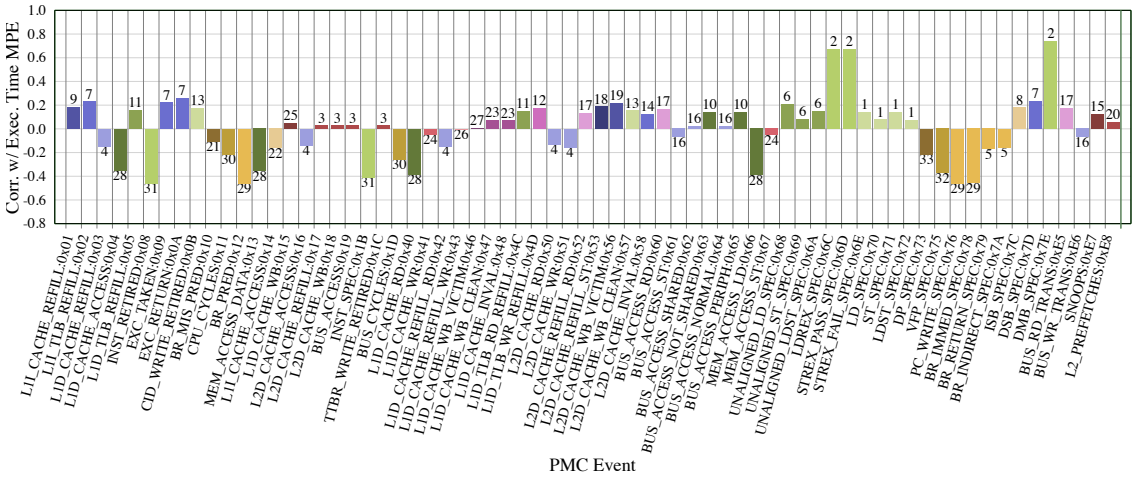


Figure 6.21: Correlation of each HW PMC (rate) with the execution time MPE (after branch predictor change), labelled and coloured by clusters derived from HCA of the correlation between PMC events across different workloads.

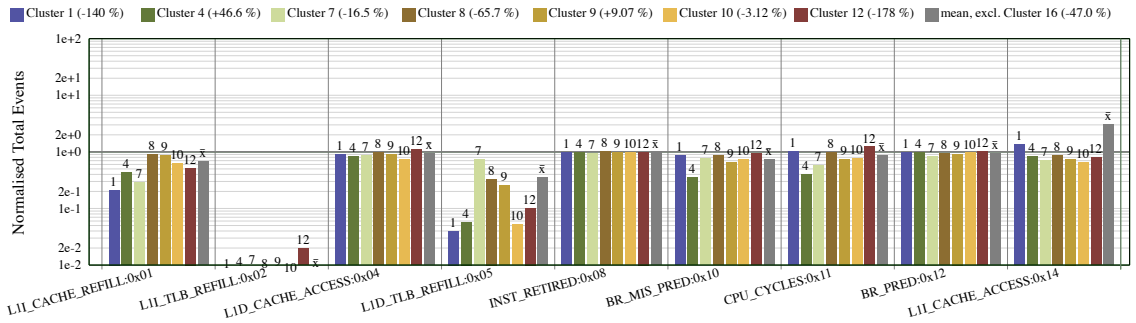


Figure 6.22: Total gem5 events normalised with their HW PMC equivalent (bars over 1 indicate that gem5 overestimates the PMC event). Results shown for selected clusters. The mean bars exclude Cluster 16.

higher number of L1I cache accesses in the model, while clusters 14, 15 and 16 have a significantly higher number of L1I cache misses. These clusters all have a significant number of floating point operations.

Comparing Figures 6.9 and 6.22 confirms that the modification to the branch predictor significantly improves its performance to be closer to that of the real hardware. In fact, the modelled branch predictor outperforms the one in hardware. Furthermore, after making this change, the number of ITLB misses has decreased dramatically, confirming the conclusion from the previous analysis that the branch predictor induced L1 ITLB misses. Modelled ILTB misses are now even more infrequent than their occurrence on the HW platform, caused by the 64-entry L1I TLB being specified in the model. However, due to the fact the performance penalty of accessing the L2 ITLB is far too high, amending the size of the L1 ITLB actually makes the model perform worse.

Despite the improvements in the gem5 model, it actually causes the power error to increase, largely due to the L1D miss write event now having a large rate error on the *par-bitcount* workload. This event error appears in all repetitions at the same DVFS level as well as the other DVFS level experiments. The power MAPE at 1 GHz was 10.3% before the branch predictor fix and 12.8% after the fix. Note that the large deviation between the modelled and measured rate of event `0x14` for Cluster 15 in Figure 6.23 explains the high mean for `0x14` in Figure 6.22.

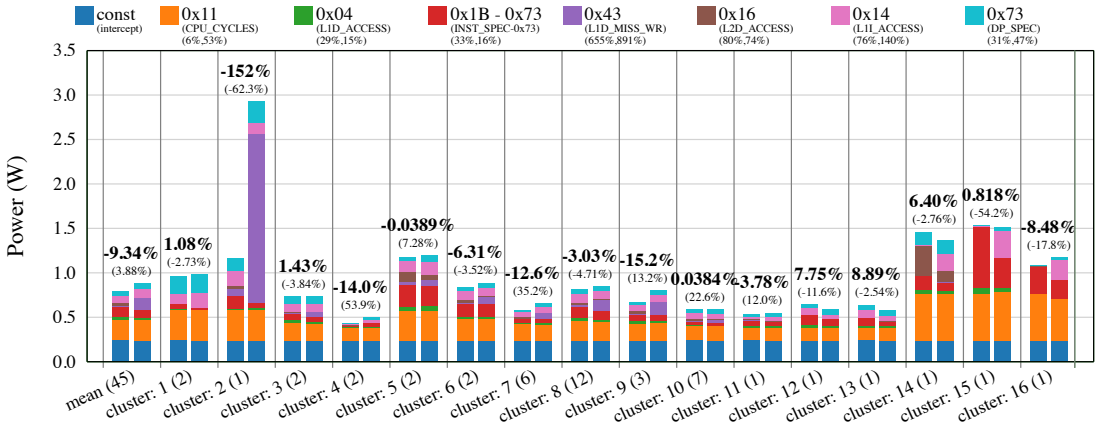
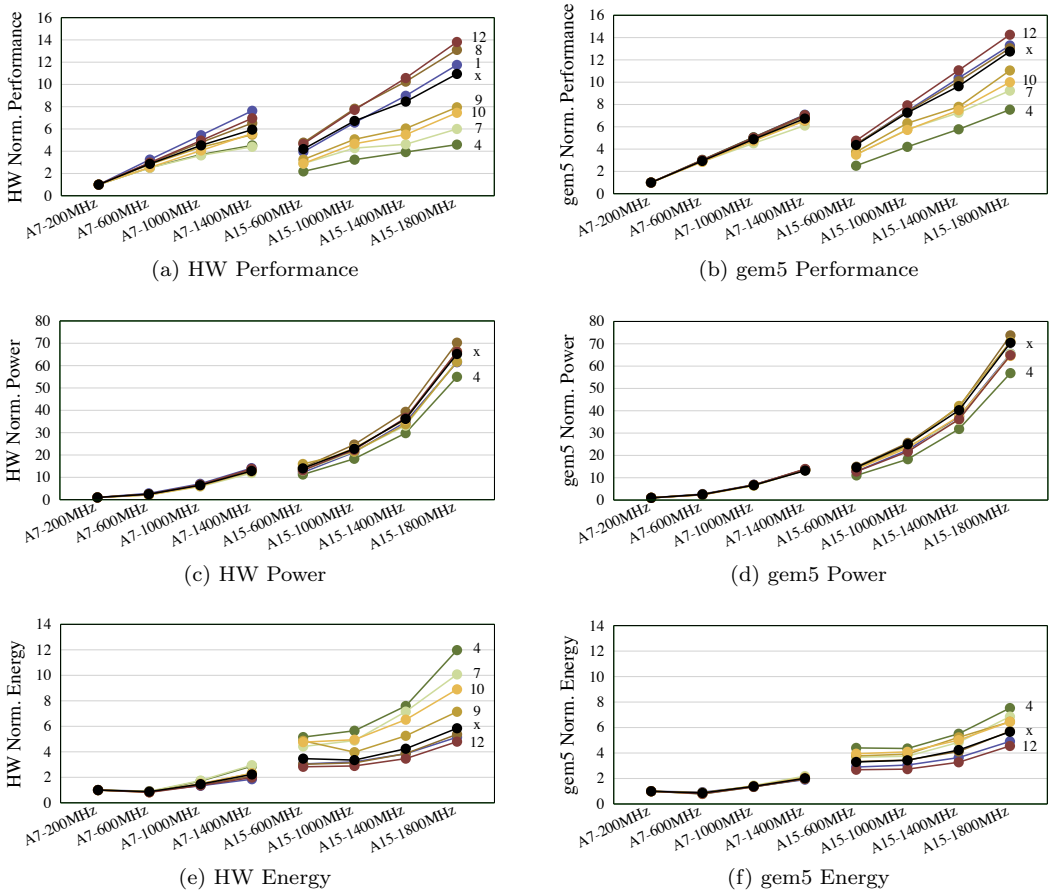


Figure 6.23: Comparison of estimated power between using the HW PMC events (left bar) and the gem5 events (right bar) for each cluster (number of workloads in cluster in brackets in X-Axis labels) as per Figure 6.5 (Cortex-A15 CPU, after branch predictor change). The power MAPE is above each bar pair in bold, with the energy MAPE below it in brackets. The bars are colour-coded to show the power contribution from each model component. $f_{clk} = 1$ GHz. Note that the workloads in some clusters have change and so Figure 6.20 must be checked before comparing with clusters in Figures 6.13 and 6.12



(a) HW Performance

(b) gem5 Performance

(c) HW Power

(d) gem5 Power

(e) HW Energy

(f) gem5 Energy

Figure 6.24: Performance, power and energy scaling normalised to 200 MHz on the Cortex-A7 CPU. Cluster numbers correspond to Figure 6.5

The performance scaling across HMP cores and DVFS level was significantly improved after the branch predictor fix (Figure 6.24). The Cortex-A7 performance scaling does not capture the diversity of the workloads of different clusters. However, the average scaling from 200 MHz to 1400 MHz in the model is very close to that on hardware. The scaling between the Arm Cortex-A7 and Arm Cortex-A15 has improved significantly. The Cortex-A15 scaling is close to that of hardware and the diversity between workloads of different clusters has been captured. The workloads in CPU-bound workload clusters scale very accurately as the DVFS level is increased. However, the effect of the optimistic memory latency is clear from the scaling. The workloads that do not scale well on HW have a far more linear curve in the gem5 model. For example, the performance of cluster 4 (which is also the cluster with the largest MPE, see Figure 6.20) scales far to quickly on the model, whereas on HW the slowdown with frequency due to being memory-bound is clearly observed. While the power scaling is accurate across DVFS levels and HMP cores, this optimistic memory latency causes large errors in the energy scaling. For example, the energy increase of Cluster 4 (from 200 MHz on the Cortex-A7 to 1800 MHz on the Cortex-A15) is 12× on HW but the gem5 model reports it as $< 8\times$.

The largest source of error is now related to the memory latency and inter-process communication latency being too optimistic in the model. Large sources of error were also found relating to floating point operations and L1I cache accesses. While the TLB inaccuracies persist, the incorrectly sized L1 ITLB covers this from being a significant source of error in the model.

As well as affirming the identified errors in the BP (and their relationship with the ITLB), this also underlines a key motivation for a tool such as GemStone, that automatically evaluates a gem5 model against a fixed HW platform. In this case, a researcher would see very different results for their study depending on when they downloaded gem5. The GemStone tool can be run after a change has been made to the simulator to verify the model behaviour against the HW reference (i.e. ensuring no major bugs have been introduced). It can also be run by the user to ensure the model gives the required level of accuracy and is suitable for their use case. Furthermore, a model correction can cause a larger MAPE due to other errors present (e.g. increasing the L1 ITLB size), necessitating a tool that analyses sources of error. Remaining sources of error can be reduced by making changes and iteratively analysing the result with GemStone.

6.8 Conclusion

This work has presented a systematic methodology for comparing CPU performance models to reference hardware platforms and identifying sources of error, allowing such models to be improved, extended to other CPUs, validated after changes, and applicability tested for specific use cases. It employs statistical analysis and does not require detailed CPU specifications to be known. The GemStone open-source software tool has been presented, which automates the full methodology with gem5. Accurate energy analysis has been enabled in gem5 by developing and integrating empirical power models of an Arm Cortex-A7 and Arm Cortex-A15, achieving MAPEs of 6.6% and 3.3%, respectively. Furthermore, a tool that allows power and energy analysis to be retrospectively applied to gem5 simulations is presented. Additionally, the effect of errors in the gem5 models on the performance, power and energy has been analysed, including the scaling with DVFS levels and between core types. It was also shown how a low power MAPE can be achieved despite significant errors on certain model inputs, thus showing the importance of using a more detailed evaluation

of model quality. This work has also highlighted many aspects of the gem5 models to analyse in more detail, including the TLB hierarchy, classification of floating-point and SIMD operations, and how the L1I cache is accessed. The methodology identified significant errors in the existing Arm Cortex-A15 gem5 model and found the branch predictor to be the key source of error. A later version of gem5 included a fix for a branch predictor bug and the mean percentage error swung from -51% to $+10\%$, further motivating the use of the presented methodology to validate simulator changes.

Chapter 7

Conclusion and Future Work

Mobile devices have become ubiquitous in modern society, and are becoming increasingly more integrated with people's work and personal lives. The processors in modern devices are becoming more complex in order to attain the performance required to enable new applications, such as augmented reality, 3D vision, machine learning, and advanced photography and filmography. However, mobile processors must operate under strict energy budgets and thermal limits. Not only does improving the energy-efficiency increase the battery-life of such devices, but is required to allow them to achieve a higher peak performance while remaining within their thermal design power (TDP) and energy budgets. Mobile processors experience extreme diversity both in the type of workload and the level of computation required. To ensure they can provide a high peak performance when demanded and maintain maximum energy efficiency across all operating scenarios, modern mobile CPUs employ a large number of DVFS and DPM options, in addition to being able to dynamically migrate between HMP cores that have different micro-architectures aimed at different performance/power-efficiency trade-off points. More recently, Arm DynamIQ has been introduced, which supersedes Arm's big.LITTLE technology and enables cores of varying micro-architectures to reside in the same cluster, allowing them to be more tightly-coupled.

However, in order to exploit the vast number of energy-saving options available, these power and performance management mechanisms must be controlled effectively and efficiently at run-time. Many works have proposed run-time management systems that optimise device performance, power, energy, temperature and/or life-time reliability, while considering the current requirements and conditions. In order to make effective decisions, run-time knowledge of workload and core power consumption is required. The importance and applicability of this to modern systems has been highlighted by the recent development of the Energy-Aware Scheduler (EAS) that enhances Linux by enabling power management decisions to be made alongside scheduling decisions. Moreover the EAS requires a task load and energy model to make decisions.

Accurate performance and power models also underpin micro-architectural research, system-level design and optimisation, and the testing of future technologies. While valuable to research, high-level and flexible modelling tools inherently contain limitations and errors that can affect the results and conclusions drawn from research, if the limitations are not fully understood by the users.

Chapters 3-5 of this thesis presented several novel methodologies and insights for developing accurate and stable run-time CPU power models that are able to provide online energy estimates to RTM and

EAS system software. Chapter 6 identified significant errors in widely used performance simulation tools and presented methodologies for understanding and identifying the sources of such errors. The accurate power modelling techniques were then combined with the performance simulator calibration tools to enable accurate, hardware validated performance, power and energy simulation for design-space exploration.

More specifically, Chapter 3 developed empirical run-time power models for an Arm Cortex-A8 single-core device that used hardware performance monitoring counters (PMCs) as input features. While PMCs have long been shown to be effective for run-time CPU power estimation, only a handful of works focussed on mobile devices, despite energy-efficiency being of particular importance on such devices, and the fact that mobile processors and mobile workloads tend to have higher power variation, making power estimation more challenging. This was found to be largely due to the technical challenges involved in obtaining PMC events from mobile CPUs and accurately measuring the power consumption directly. Chapter 3 presented openly available software tools for extracting PMC events from Arm-based mobile processors, and presented a methodology for measuring CPU power consumption, combining it with the PMC event data, and post-processing. The first PMC-based power model for a mobile device that uses measured (as opposed to simulated) data and that has realistically high workload and power variation, was presented. It was found that existing works typically used a relatively small set of workloads to train and test the models. In real-life usage, the power models must estimate the power consumption of phases of the workload execution, as opposed to the whole workload itself. These workload phases are much more diverse and it is therefore more difficult to predict the power consumption. A method for extracting diverse workload phases for training and validation using a CPD algorithm was also presented. Another finding was that selecting events using correlation, as has been presented in existing works, is ineffective as it increases the intercorrelation in the model input features. A method for selecting model input features while reducing intercorrelation was presented. Later in Chapter 4, the effect of intercorrelation on coefficient stability and model error under different training and validation scenarios is analysed and event selection methods are investigated further. A separate model for each of the four DVFS levels on the platform considered in Chapter 3 was built and achieved a cross-fold validated MAPE of 1.91%, 1.93%, 1.49% and 1.40%, for 300 MHz, 600 MHz, 800 MHz and 1000 MHz clock frequencies, respectively. It was also demonstrated how a model developed using the presented methodology and diverse workload phases is able to accurately estimate the power consumption of the real-time power traces with a MAPE of less than 2.2% without a moving average filter and 1.3% with a moving average filter, demonstrating the responsiveness of the model.

Chapter 4 improved on the setup and methodology outlined in Chapter 3, identified and addressed significant shortcomings in existing methodologies, used a single model for all DVFS levels, and demonstrated the approach on an Arm big.LITTLE system containing a quad-core Arm Cortex-A7 InO CPU and an Arm Cortex-A15 OoO CPU. The methodology for developing PMC-based power models in this chapter uniquely considered coefficient stability, which allows the resulting model to make more accurate predictions across a wider range of observations, even if they are not well represented in the training data. This is an important property for real-world power models as the power consumption of individual, and diverse, phases needs to be estimated. The two key requirements for stable model coefficients were found to be:

1. Minimal multicollinearity between the input features;

2. Diverse training observations that exercise each input feature individually.

As identified in Chapter 3, existing works tend to train and test with a small set of workloads from typical benchmarking suites, such as MiBench and SPEC, and typically use the correlation of each event with power consumption to select PMCs, which leads to a PMC event selection with high multicollinearity. Chapter 4 proposed the use of the VIF to evaluate a PMC event selection and presented a novel methodology for selecting and transforming PMC events that captures as much information as possible while reducing coefficient errors. It was experimentally demonstrated how a model with a stable PMC event selection using the presented methodology makes more accurate predictions when tested on a larger set of diverse workloads. For example, a model with a typical selection of PMC events achieved a MAPE of $> 8\%$ and a maximum error of $> 45\%$ when trained with a small set of diverse workloads and tested on a large set of diverse workloads. An otherwise identical model using the proposed PMC event selection methodology and PMC event processing achieved a MAPE of $< 3.5\%$ and a maximum error of $< 15\%$ for the same training and testing scenario. It was also demonstrated how the same unstable model achieves an apparent low MAPE when trained and tested with typical benchmarks (i.e. from MiBench), showing that the common practice of reporting MAPE alone can be misleading, as it is highly dependent on the training and testing scenario, as well as the power variation in the system and workloads. Other input feature selection techniques were also considered, and insights from applying techniques, such as PCA and HCA, to PMC event data was presented.

The methodology presented in Chapter 4 creates a single model for all DVFS levels, which means it must correctly understand how voltage, frequency, activity, and temperature (which changes with voltage and frequency, as well as with activity and ambient conditions) relates to the power consumption. Furthermore, the experimental setup must provide consistent results when repeated to capture changing variables. The model specification was found to be critical in developing an accurate power model and significant errors in model equations presented in existing works were identified. Chapter 4 used knowledge of the various power components, calculated statistical significance, and the studentised residuals to develop the model equation. While the CPU thermal sensors were not used in this model, the effect of temperature due to switching activity and the voltage was considered. It was shown how it is possible to train the model by only running the full training set at a single DVFS level, and running only idle observations at the other DVFS levels, demonstrating the robustness of the model equation. This thesis identified the problem of heteroscedasticity in CPU power modelling, which impacts the reliability of reported statistics in existing works, including the confidence intervals, standard errors and p-values. Chapter 4 addressed this using a heteroscedasticity consistent standard error (HCSE) estimator, and also proposed Weighted Least Squares (WLS) regression as a method to investigate.

Another novel finding in PMC-based run-time power modelling was that the voltage supplied to the CPU from the voltage regular was not constant at a fixed DVFS level. In run-time power estimation applications, the voltage cannot usually be measured and is assumed to be fixed to a pre-determined value, which resulted in a MAPE of 2.6% (using the measured voltage) increasing to over 8% (using the idle voltage) in the case of the highest DVFS level considered. A novel voltage model was proposed that reduced the MAPE by up to 5%. Techniques were also presented to analyse how the specific workload observations in the training set influenced the fit, and which workloads were under-represented in the training data.

In addition to the methodology, Chapter 4 presented an Arm Cortex-A7 (InO) and Arm Cortex-

A15 (OoO) power model with MAPEs of 3.8% and 2.8%, respectively, and VIFs of under five. A simplified version of the Cortex-A15 model (to work at a single DVFS level, as most existing works consider separate models for each DVFS level) was compared with models presented in existing works, including the one presented in Chapter 3. All of the models in the comparison were re-trained with 20 diverse workloads and tested with 60 diverse workloads, using 10-fold cross validation. The model proposed in Chapter 4 achieved a superior MAPE and R^2 of 2.9% and 0.999, respectively. After the Cycle Count event was added to the existing models (as it is required on platforms implementing more advanced power management), the two closest performing models achieved MAPEs of over 6%. One of these was the model presented in Chapter 3, despite it only using four PMC events to the other model's six events.

While the methodology presented in Chapter 4 was demonstrated on an Arm mobile platform, it is applicable to CPUs implementing other architectures, and later works have developed power models for Intel x86 desktop and HPC platforms directly based on the approach presented in Chapter 4 [58, 234]. Furthermore, works have made use of the presented software tools and models to obtain experimental data and CPU power estimates for their investigations [215].

Few existing works consider the relationship between the static power consumption and the temperature. The model presented in Chapter 4 *absorbs* the temperature effects due to the voltage and switching activity by including extra polynomial terms (the statistical significance of model components as well as the leakage mechanism equations in Section 2.1 were used to derive these terms). Chapter 5 presented a novel methodology for modelling the effects of temperature on the static power consumption and incorporating this into the run-time power model, using data from the built-in CPU thermal sensors. On a testing set with large changes in ambient thermal conditions, the model MAPE was reduced from 6.0% to 3.7% by adding the thermal compensation. Furthermore, Chapter 5 modelled the effect of switching cores offline, decomposed the empirical power models, measured the energy required to access different levels of the memory hierarchy, and presented a methodology for estimating temperature, using the power model and a simple equivalent circuit model. As well as allowing the models to be used on platforms without thermal sensors, the temperature model enables thermally-compensated power estimation and temperature estimation in simulation tools, such as gem5. The model decomposition allows the models to be combined with bottom-up techniques, such as RTL simulation and CACTI for component-level energy analysis.

The power models presented in this thesis, and the majority of existing works, are built using linear regression. Chapter 3 discusses the benefits of using a linear modelling method over a non-linear approach, such as a neural network or k-nearest neighbours, and Chapters 4 and 5 demonstrate how a correctly specified linear modelling approach can accurately estimate the power consumption, including the effect of voltage, frequency and temperature. As illustrated in Chapter 3, non-linear models are more susceptible to overfitting, require larger amounts of training data, and are more difficult to interpret as the effect of each feature on the response is difficult to derive, p-values are not available, and the R^2 is invalid. While linear approaches require more knowledge of the system being modelled, they provide more insights into the underlying relationships being modelled. For example, Chapter 4 identified the presence, and estimated the magnitude, of a constant background dynamic power component using the model equation, R^2 value, and p-values. Chapter 5 verified this through measurement when the cores were switched offline. In another example, Chapter 5 used the residuals and p-values to model the relationship between the temperature and the leakage power. The linear model also allowed the power model to be decomposed into its constituting parts,

and the non-ideal voltage regulator behaviour to be understood. With a non-linear approach, these important observations would likely not be identified.

The residual plots in Chapters 3, 4 and 5 show that there are no clear non-linear relationships between the features and the dependent variable that are not being captured by the model. The use of a linear model is not a limiting factor for further improvements in accuracy, but the number of PMC events that can be used as inputs, the training data for modelling diverse thermal environments, modelling the non-ideal effects of the voltage regulation, remaining (albeit significantly reduced) presence of multicollinearity, and the number of diverse training workloads. By refining these aspects of the presented approach, an even higher accuracy could be obtained.

Computing research in both industry and academia heavily rely upon simulation tools for evaluating new ideas and design-space exploration. These simulation tools underpin the results and conclusions drawn from these works of research. Concerns in the research community regarding the accuracy and reliability of power and energy simulation tools motivated Chapter 6 to propose using the empirical power models developed in Chapters 3, 4 and 5 with widely used performance simulation tools, such as gem5, to enable accurate performance, power and energy modelling to the research community. However, Chapter 6 identified significant errors in the tested performance model and this motivated the development of a methodology for analysing and identifying sources of error. The methodology, which employs HCA and correlation analysis, enables existing CPU models to be improved, models for other micro-architectures to be developed, validation of changes made to the simulator, and users to understand how errors in the model may affect their specific use case and decide whether the models are suitable. For example, when the key error identified by the methodology was addressed, the MPE swung from -51% to +10%. In this case, the identified error was a bug that had been present for several months within the gem5 simulation framework, potentially impacting the results of many (published) investigations. It is inevitable that bugs are introduced in large, complex, open-source software tools with an active development community, and this further motivates the need for the presented methodology. Once the methodology of hardware-validating and calibrating the performance models had been developed, the empirical power models were optimised for use in the simulator, therefore enabling accurate performance, power and energy simulation. Furthermore, the GemStone software tools were released, which collect experimental data from both hardware platforms and the gem5 performance simulator; compare the two and utilise statistical and machine learning techniques to identify sources of error in the models; apply the empirical power models to both the performance model and hardware data; export power equations for online power estimations in gem5; and evaluate the performance, power and energy scaling across multiple DVFS levels and HMP core types, while simultaneously considering different types of workloads (clustered using unsupervised machine learning techniques). Finally, remaining sources of error in models were identified and discussed.

It is hoped that the techniques and insights presented in this thesis encourage greater statistical rigour when developing and validating models for system power and performance estimation, as well as in other fields.

7.1 Research Questions

This thesis aimed to answer the research questions outlined in Chapter 1. Direct answers to the questions, as well as the questions themselves, are listed below:

1. *How robust are the current state-of-the-art power modelling methodologies and how applicable are they to mobile CPUs?*

   Several shortcomings in existing approaches were identified that impact how well the model can make predictions across a wide range of scenarios, the validity of reported statistics, and the sensitivity of the model to changes in the input features. In particular, problems related to multicollinearity, coefficient stability, model specification, heteroscedasticity and training data diversity are highlighted and addressed in this thesis.

   Most existing approaches for run-time power estimation focus on desktop and server systems. In mobile devices, energy-efficiency is of particular importance and, due to the in-built power-saving techniques and diversity in workload intensity, there is greater power variation, making estimation more difficult. The few examples of PMC-based power models aimed at mobile devices had low power variation due to training workload selection and using early device implementations without key power management features enabled. Furthermore, they typically used simulated data, due to the difficulties in obtaining PMC data. Chapter 3 presented a methodology for identifying diverse workload phases using an unsupervised CPD algorithm, and Chapter 4 used a HMP system with more power-saving techniques enabled and a large set of diverse workloads, including manually tuned micro-benchmarks. Furthermore, existing approaches did not consider coefficient stability, which was experimentally demonstrated in Chapter 4 to be critical in developing a model that can make accurate and reliable predictions across a wide range of workload phases, which is of particular importance in mobile applications. In particular it was also shown that the validation techniques used in many existing works resulted in optimistic MAPEs and generally did not employ cross-validation, statistical significance tests, inspection of the residuals or tests for bias and variance.

   Modern mobile devices and operating systems put CPU cores into idle states when not in use, which is particularly important considering the growing number of HMP cores they include. However, the impact of switching cores offline was not considered in existing PMC-based mobile power models. In addition to this, most existing PMC-based power models do not consider temperature, which has a significant impact on the leakage power consumption, which is also of particular importance in modern mobile devices.

   Finally, Chapter 4 identified that the voltage being supplied to the CPU varied with the CPU load, due to the non-ideal characteristics of the voltage regulator, which increased the model error by up to 5%. Chapter 4 addressed this with a novel voltage model. While the complexity and power requirements of mobile devices is constantly increasing, the available physical space is reducing. Reducing the area and increasing the current handling capabilities of the voltage regulator means that its performance, i.e. its ability to provide a stable and fixed voltage, is reduced. This is likely to get worse over time, particularly as newer devices are using integrated voltage regulators (iVRs), also known as integrated on-chip voltage regulators (OCVRs), to further reduce the area and cost.

   This thesis has highlighted several areas where existing approaches are not robust and suitable for mobile devices and has presented methodologies that address these areas.

2. *How effective are PMC-based power models when applied to mobile CPUs in terms of accuracy and responsiveness?*

   After identifying and addressing the shortcomings described in the previous answer, linear PMC-based power models were found to provide accurate power estimations across a wide range of workloads and workload phases. A MAPE of less than 2% was achieved using the methodology presented in Chapter 3 when tested on a single-core mobile CPU using a diverse set of workload phases extracted using a CPD algorithm. Furthermore, the responsiveness was demonstrated by applying the same model to run-time power traces, achieving a MAPE of less than 2.2% on the 10 ms samples and less than 1.25% after applying a moving average with a window size of 20 samples.

   Chapter 4, which made further improvements to the methodology, used a single equation across all DVFS levels, and demonstrated the approach on an 8-core HMP system, achieved a cross-validated MAPE of 3.8% and 2.8% for the Arm Cortex-A7 quad-core cluster and the Arm Cortex-A15 quad-core cluster, respectively, when cross-validated with 60 diverse workloads. These models achieved significantly lower errors when compared directly with existing approaches, and were able to maintain their low accuracy when trained with a small number of workloads, and when the training workloads were only run at a single DVFS level.

   The proposed model in Chapter 5, which enhances the model presented in Chapter 4 by using the thermal sensors as input features, achieved a MAPE of 3.7% with ambient temperatures ranging from a minimum of 31°C to 91°C and across eight DVFS levels on the OoO quad-core Arm Cortex-A15.

3. *Which micro-architectural event statistics are most effective in estimating the power consumption across a wide range of workloads and workload phases, and how many input features are required?*

   This thesis has employed and evaluated several techniques for analysing and selecting model features (PMC Events) that extract the maximum amount of useful information for predicting power, while avoided repeated information that increases multicollinearity and gives rise to unstable coefficients.

   Previous works prescribed a selection of events (e.g. [244]) that work across multiple micro-architectures. However, this thesis found the event selection critical and specific for a particular CPU, and varied considerably between inO and OoO micro-architectures. Furthermore, even CPUs from the same manufacturer have different events implemented between different micro-architectures due to diverging micro-architectural features, and also had varying numbers of counters. This thesis therefore proposed an automated PMC event selection methodology that identifies a suitable set of PMC events for a specific micro-architecture, adapting to the available events and number of counters.

   An event indicating the number of instructions executed (e.g. Event `0x1B`) was found the most important in estimating the power consumption, as expected. In Chapter 3 and previous works, the cycle counter (which counts the number of *active* cycles) was not included, as it was not required on the platforms being considered. However, in devices with more advanced power management options enabled, the cycle counter is essential in estimating the power consumption accurately. A MAPE of less than 15% can be achieved with these two events alone with the methodology proposed in Chapter 4.

With the proposed PMC event selection methodology, over 95% of the power variation in the training data could be explained from just three events for the OoO Arm Cortex-A15 cluster. Without the proposed event transformations, only four events (including the cycle counter) could be used without the VIF exceeding an acceptable level. With all seven events selected and the inter-correlation reduced using transformations, the MAPE is reduced from over 6% to less than 3%. While the transformations would allow up to ten events to be selected in this example before the VIF rises too high, only seven events can be monitored simultaneously by the hardware. If more counters were available, the error could potentially be reduced much further using the presented methodology, providing that the effects of temperature and non-ideal voltage regulation are carefully accounted for. In addition to the events identified by the automated methodology, the residual plots in Chapter 4 showed where some *sub-events* would improve the accuracy. For example, accuracy would be improved if the UNALIGNED_LDST_SPEC:0x6A event (unaligned loads and stores) was replaced by both the UNALIGNED_ST_SPEC:0x69 (unaligned stores) and UNALIGNED_LD_SPEC:0x68 (unaligned loads).

4. *Do the existing run-time PMC-based power models address practical considerations and how suitable are they for real-world deployment?*

   Existing approaches did not take coefficient stability into account and typically validated the model on a small set of typical benchmark suite workloads, reporting the MAPE. In run-time management, the estimated power of workload phases, as opposed to the total power consumption of the whole workload duration, is required. Workload phases are significantly more diverse than the whole duration of the workload and this thesis experimentally demonstrates how training and testing with typical benchmarking suites produces an optimistic MAPE. Furthermore, it is shown how models built without considering coefficient stability are overly sensitive to small changes in the input features and have large errors when tested on a diverse set of workloads. This thesis uses CPD algorithms and tuned micro-benchmarks for extracting workload phases for diverse training and testing data, and the methodology reduces errors in the model coefficients to improve accuracy across a wider range of observations. They are also tested rigorously using diverse testing observations and cross-fold validation. This ensures that the models are more suitable for real world execution phases, and that the reported MAPE is realistic.

   Another finding of this thesis was that the voltage supplied to the CPU by the voltage regulator was not constant at a fixed DVFS level. This voltage variation, which is dependent on the CPU current consumption, significantly affects the power consumption. In real world applications, the CPU voltage cannot usually be measured and so this thesis presented a novel voltage model that models the non-idle voltage regulator behaviour.

   In real-world deployment, the ambient thermal conditions are not stable, particularly because mobile devices are carried with people in diverse thermal environments. Furthermore, the other components in the device impact the temperature of the CPU itself. Another consideration is that mobile devices make heavy use of CPU idle states, where unused cores are switched to a low power mode. The methodology presented in Chapter 5 uniquely considers both of these in mobile PMC-based power models.

   Another practical consideration is the time required to train the models. As well as allowing the input voltage, frequency, and temperature to be independently changed (unlike a per-frequency model), the presented model equation reduces the training data collection time

from over 40 hours to less than 20 minutes by only training the full set of workloads at a single DVFS level.

In order to implement run-time power models, a module in the OS kernel must sample the PMC registers on each CPU core at the sample frequency and calculate the linear equations. As well as the overhead, the maximum sampling frequency is affected by the event rate. For example, memory events occur significantly less frequently than CPU core events. Therefore the sampling frequency must be sufficiently low compared to the rate of the events. The overhead was verified to be negligible in Chapter 4, and the models were tested to work effectively at sample frequencies between 10 Hz and 100 Hz at the worst-case DVFS level of 200 MHz and with a non-optimised equation. Further optimisations for run-time estimations are possible, such as pre-computing fixed values. The linear equations derived from OLS provide models with minimal computational complexity. Furthermore, the PMC event selection methodology presented in Chapter 4 also provides an analysis of the trade-off between the number of events and the R^2, VIF, and the MAPE, and enables a smaller set of events to be chosen, for a lower overhead, if required. The coefficient stability and model specification make it straightforward to remove the events with a low rate at lower CPU clock frequencies, if a faster model sample frequency is required.

5. *Are current performance and power simulation frameworks accurate and reliable and, if not, how can they be improved?*

   A literature review in Chapter 2 identified significant sources of error in both performance simulation frameworks and, in particular, power modelling frameworks. The results from a performance simulator are used as the inputs into the power simulation framework and the errors therefore compound. Chapter 6 compared an existing gem5 model against the same hardware platform it was modelled on and identified errors much larger than those reported in existing works. For example, the Arm Cortex-A15 model had a MAPE of 59%.

   This motivated this thesis to present a methodology for identifying sources of error in simulator models by comparing them to a hardware platform and employing statistical and machine learning techniques. Several key sources of error were identified and a later version of the gem5 simulation framework had a bug fix for the one identified as being most significant, which reduced the MAPE to 18%.

   The methodology allows users of the simulation tools to understand how inevitable errors can impact their specific use case, as well as enabling the models to be periodically validated and improved.

6. *How applicable are empirical PMC-based power models to design-space exploration?*

   While high-level bottom-up power, area and timing models are flexible, they suffer from significant sources of error in all of the CPU sub-components. Although the PMC-based power models are characterised for a specific hardware platform, they have a high accuracy that has been validated against the hardware platform itself.

   In a typical scenario where a researcher or system designer is proposing a new idea, they first take a baseline performance and energy estimate of an example system, and then implement their changes to the models before evaluating the change in performance and power.

   A more accurate simulation approach would be to use gate-level or even SPICE simulations. However, due to the required amount of computation time, this is usually impractical for

design-space exploration of modern CPUs. Furthermore, the IP required to run such simulations is often not available to research teams.

A typical approach is to use a performance simulation framework, such as gem5, to model performance and the PMCs, and then use a power simulation framework, such as McPAT, to analyse the energy. This thesis proposes the use of a hardware-validated gem5 model, using the presented methodology for identifying sources of error, with a PMC-based empirical power model. After being validated against hardware, the gem5 model should respond in a representative fashion to the change being made, including how the change impacts different types of workloads. The empirical power model will more accurately estimate the power difference from the changes in the input PMC events. Any power changes due to hardware changes themselves (e.g. different size of a component, or extra logic added) must be accounted for separately, for example, by using empirical analysis, high-level bottom-up techniques (e.g. CACTI) or (as it is usually a small component) low level SPICE or gate-level simulations.

7.2 Future Work

This thesis has identified further challenges that remain to be addressed, as well as provide techniques, insights and methodologies that can be applied to solving other research problems. This section outlines avenues for future work that were unable to be completely addressed due to time constraints.

7.2.1 Adaptive Training Workloads and Enhanced Stability

Chapter 4 demonstrated the importance of training set diversity in developing stable power models and made use of manually tuned micro-benchmarks to activate specific micro-architectural activity. Chapter 5 developed a method of observing the power required to access various levels of the memory hierarchy and of memory prefetching. This results in four diverse observations that provide a model with memory access energies (Figure 7.1). However, the points at which these regions occur depend on the design of the memory hierarchy. The methodology in Section 5.7 could be extended to automatically identify these points (labelled A, B, C, and D in Figure 7.1) using the feedback from the PMC events, resulting in an adaptive training workload that identifies the most diverse observations for the specific CPU being characterised. This could then be extended to trigger other events such as branch mispredictions and instruction cache misses and be combined with CPU intensive micro-benchmarks to generate a full set of training data. Moreover, the experimental setup in Chapter 4 required the training workloads to be repeated many times in order to obtain accurate power readings from the sensors (which had a low sample rate). The proposed adaptive microbenchmark could address this as it can sustain specific (constant) operations, enabling accurate and repeatable power readings from fewer samples (this functionality was utilised in Chapter 6 to measure the memory latency at specific array sizes, to reduce simulation time).

It is expected that, in addition to reducing device characterisation time and increase the quality of the experimental data through the ability to sustain specific operating points, the diverse observations would significantly improve model stability, resulting in *ultra-stable* power models.

Research questions include:

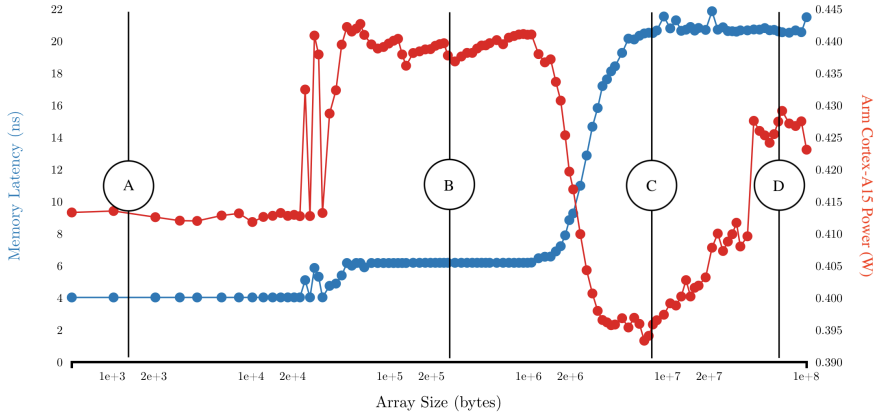


Figure 7.1: Diverse observation from memory latency experiments of Section 5.7

- How much does this technique reduce multicollinearity (considering a fixed set of features)?
- Is the adaptive training workload (including CPU micro-benchmarks) effective in training the model alone, or does the model benefit from including other workloads and workload micro-phases in the training data?
- How effective is this technique in identifying specific, diverse, operating points across a wide range of CPUs?
- Does using this adaptive training workload improve, and if so, by how much, the model accuracy when tested on a large set of diverse workloads?

7.2.2 Effects of Voltage Regulation on Power Consumption and RTM Decisions

Chapter 4 identified how the voltage supplied by the voltage regulator varies under different CPU loads. The relationship between the voltage deviation and current draw was not simply due to a voltage drop across a resistance; the voltage regulators (or power management integrated circuits, PMICs) use a feedback loop to control the supplied voltage. Assuming a fixed voltage at a DVFS level results in a significant increase in error. This is addressed in Section 4.6 with a voltage model. However, there is a large amount of further research to be conducted in this area. For example, a deeper analysis into how these non-ideal effects impact the CPU energy efficiency, whether an RTM can improve energy-efficiency by using the presented voltage models and considering the voltage regulator behaviour, and whether other modelling techniques or methodologies can improve on the accuracy of the presented voltage models.

Furthermore, to reduce physical space and cost, modern mobile devices include integrated on-chip voltage regulators (OCVRs) [99, 282, 207]. These consume both energy and area, and there are therefore trade-offs in their design which determines the maximum current they can provide to the CPU as well as the maximum voltage swing within a certain period of time. This should be considered when making run-time management decisions, such as the ones highlighted in Section 7.2.5. However, currently these problems are not the subject of much research, with the exception of a recent work by Sayadi *et al.* [255].

A related consideration is the battery characteristics of mobile devices [303, 262], which should also be considered when estimating run-time power consumption and guiding RTM decisions. For example, the charge reduction in a battery is not directly proportional to the power consumption and therefore battery capacitance can be saved by considering the battery characteristics and making RTM decisions accordingly.

7.2.3 Combining Top-Down and Low-Level Bottom-Up Techniques

The empirical power models developed in Chapters 3, 4 and 5 can be accurately decomposed further using information from RTL and gate-level simulation. This could be applied to the full design, or just large components, such as the caches and branch prediction tables. This would also enable the top-down power models to be more flexible when used with full-system simulation tools, such as gem5.

7.2.4 Power Estimation in Diverse Thermal Environments

Section 4 presented power models that compensate for thermal changes due to CPU voltage, frequency and activity changes. Chapter 5 added ambient temperature compensation by modelling the effect of the CPU temperature on the leakage and using the temperature sensors from each CPU as inputs to the model. The CPU thermal conditions were modulated by changing the settings of the on-board fan, resulting in observation temperatures ranging from 31°C to 91°C. However, the use of a temperature chamber to provide a more controlled thermal environment would likely improve both the model and the results. Chapter 5 also proposes a basic electrical circuit thermal model that combines a preliminary power estimation with an empirically-derived thermal capacitance. However, while the model itself could accommodate variable ambient temperature, the required data was not available. Therefore, by utilising a temperature chamber and repeating the experiments (automated in GemStone) at various temperatures, the heat flow between the CPU and external temperatures can be modelled. An extension would be to repeat the experiments (and re-calculate the thermal capacitances) with different heatsinks and fan settings. GemStone already has the ability to collect and combine data from external sources while running at various ambient thermal conditions. This would enable improved power modelling accuracy when running on systems without thermal sensors in the CPU (including simulation tools, such as gem5).

7.2.5 Event Forecasting for Performance, Power and Energy Prediction

Modern mobile processors are becoming increasingly complex and intelligent run-time management and control (RTM) of performance and power consumption is required to improve energy-efficiency, extend battery-life and increase peak performance, while respecting thermal and power constraints (see Section 2.3). For example, devices implementing Arm big.LITTLE technology (described in more detail in Section 2.3.3) contain CPUs of a common ISA that have differing micro-architectures that are optimised for different performance-energy trade-off points; higher performance cores allow the device to have a high peak performance when necessary while more power-efficient cores enable the device to be more energy efficient, when possible. Each type of CPU can be run at different DVFS levels or put into sleep states in order to make energy savings, if possible, while achieving the

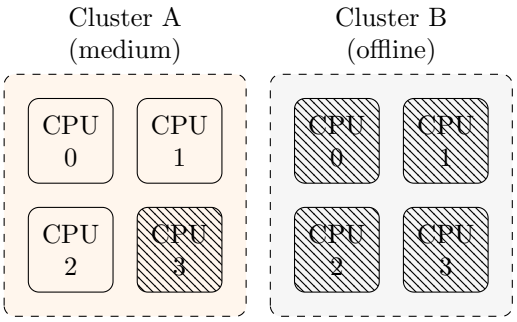
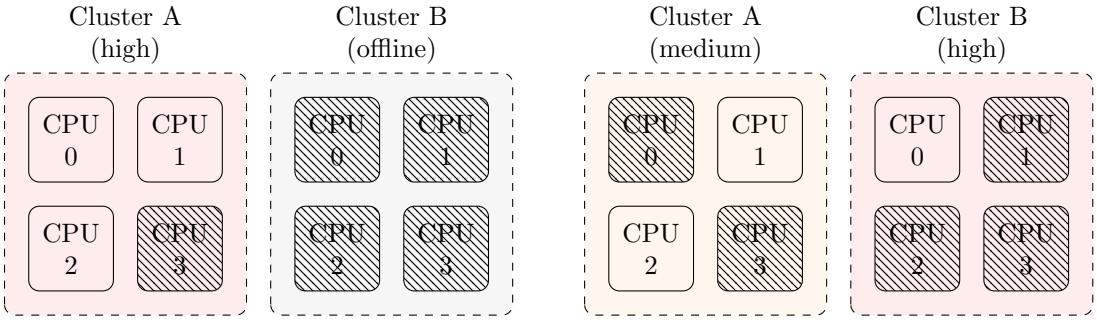


Figure 7.2: EAS Motivation: Cluster A at a medium DVFS level with three tasks running on three active cores, while Cluster B is offline.



(a) Cluster A at a high DVFS level with three tasks running on three active cores, while Cluster B is offline.

(b) Cluster A at a medium DVFS level with two tasks on two active cores, while Cluster B is at a high DVFS level with one task on one active core.

Figure 7.3: Two scheduling options to the situation where the task running on CPU0 in Cluster A requires a higher performance.

required level of performance. Chapters 3, 4, and 5 presented techniques for developing accurate and stable run-time power and thermal models for providing the RTM with the information required to make informed decisions.

However, the task of RTM is difficult and no ideal solution exists. Chapter 6 (see Figure 6.24) demonstrated how different workloads respond differently to a change in DVFS level or being run on a different micro-architecture. Several existing works have proposed techniques for estimating the impact of RTM decisions on performance or power consumption [260, 280, 238, 156]. This is particularly problematic in mobile devices, which must operate within a tight thermal budget, and where only a certain amount of power can be instantaneously provided to the CPU in a reliable manner (see Section 7.2.2). These limits must be proactively respected; reacting to exceeding the limits is not sufficient. A key challenge in RTM and scheduling is therefore predicting the performance, power, thermal and energy impact of making a decision in advance.

While many run-time management algorithms and approaches have been proposed (see Chapter 2.3) challenges in proactively maintaining thermal and power limits while considering shared voltage domains have not been sufficiently recognised or addressed. For example, it is usually not possible to independently change the DVFS level of each individual core of a CPU cluster; each core of a cluster in an Arm big.LITTLE configuration shares the same voltage domain and so the DVFS affects all cores of the cluster simultaneously. It is therefore important that energy-saving techniques are controlled in conjunction with scheduling decisions, which has motivated the recent efforts to

implement Energy-Aware Scheduling (EAS, see Section 2.3.4) into Linux [8]. For example, consider a CPU cluster *Cluster 1*, running at a medium DVFS level with 3 tasks, A, B, C running on its three active cores (Figure 7.2). If application A requires a higher level of performance, one option is to increase the DVFS level for the current cluster (affecting all the cores and applications, Figure 7.3a). However, if tasks B and C do not benefit from the high DVFS level, it may be more energy efficient to bring another cluster online (at a high DVFS level) and migrate task A it (Figure 7.3b). This decision is more complex when considering that there are different HMP core types available (e.g. big or LITTLE). The CPU power or thermal budget may mean that task A must be migrated (and the core it was previous running on switched offline) before the DVFS level is increased, therefore significantly increasing the latency of switching DVFS level (affecting the granularity at which the decisions should be made). Furthermore, Arm DynamIQ technology [7], which allows big and LITTLE cores to reside in the same cluster and some cores of the cluster to have separate DVFS domains, provides more energy-saving opportunities but requires more complex RTM to efficiently exploit them. It is not possible to effectively make accurate RTM decisions (and proactively respect power and thermal limits) unless the way the specific workloads will respond to a change in DVFS or different HMP core types is known in advance.

The insights and methodologies presented in this thesis can be combined and applied to run-time management approaches to aid the decision-making processes. Key to understanding how a workload will respond (in terms of performance, power and temperature) to a change in DVFS level, or being migrated to a different HMP core, is modelling the impact on the micro-architectural events for the type of workload being considered. Fundamental to achieving this are:

1. Techniques for accurately extracting and post-processing PMC event data (Chapter 3);
2. Methods for selecting and transforming PMC events to obtain optimal information while reducing multicollinearity (Chapter 4);
3. Change-point detection methods for identifying workload phases (Chapter 3);
4. Workload classification techniques (Chapter 6); and
5. Methods of identifying patterns between workload types and micro-architectural activity (Chapter 6).

Once the PMC events have been predicted, the predicted performance is known. These events can then be combined with power and temperature modelling methodologies to provide predicted power, energy and temperature.

7.2.6 Using Workload Classification in Power Estimation

The power modelling methodologies presented in Chapters 3, 4 and 6 identify a single set of PMC events that generalise well across all types of workload. A previous work by Mair *et al.* [185] uses workload classification to adapt the PMC event selection at run-time. However, the modelling techniques in this work are simple and the trade-offs involved in workload classification are not explored. The methodology presented in [185] uses correlation alone to select PMC events and does not consider coefficient stability, meaning that the coefficients in the models, including the model developed for the baseline comparison, will not necessarily generalise well. Furthermore, temperature, heteroscedasticity, and the residuals are not considered, and outliers are removed to

improve accuracy without justification. The classification technique employed is very primitive; classification is performed online using *if* statements to compare the magnitude of one PMC event against another, and branch accordingly.

The approach proposed by this thesis would first use the unsupervised machine learning techniques employed in Chapters 4 and 6, to identify workload classes. Separate PMC-based power models would then be developed for each class of workload, using the methodology presented in Chapters 4 and Chapter 5. The potential improvement in accuracy when using a specific set of PMC events depending on the type of workload (assuming an oracle workload classifier), compared to a model with a fixed set of PMC events, can be calculated. Supervised machine learning techniques would then be employed to develop an online workload classifier using the labels identified with unsupervised learning. The accuracy of the classifier, and the speed at which different phases can be identified and adapted to, affects how close the model accuracy is to the case considering an oracle workload classifier.

A key challenge in this approach is keeping variance low when training, and effective validation; once the full set of training data has been clustered, a model must be trained and validated on those clusters, and a sufficient number of workloads within those clusters must be used.

Another possible approach would be to train all of the models with all of the workloads, but use Weighted Least Squares (WLS) to focus accuracy on training observation of the cluster that the feature selection is optimised for.

Example research questions include:

1. How many classes are required?
2. What is the overhead of adding the workload classification and changing the monitored PMCs?
3. After a workload phase change, can any active PMC event selection identify the new correct selection, or must a specific set of 'classification' events first be used?
4. What are the trade-offs between allowing each PMC event set to contain any events versus constraining all sets to contain a subset of common events?
5. What is the prediction improvement when switching from a fixed set of PMC events to a variable set of PMC events?
6. How close can the power estimation error come to the case of an oracle workload classifier and instant PMC event selection switches?
7. What impact does a misclassification have on the power estimation?
8. How does the threshold of a PMC selection change effect the accuracy and overhead?
9. How can the problem of requiring many more workloads for training and validation be addressed?
10. Should each model optimised for a subset of the workloads be trained solely on the workloads in that subset, or should all of the models be trained on all of the data, perhaps using WLS to focus accuracy on the workloads in the subset?

7.2.7 Dynamic Micro-Architectural Feature Extraction

Only several of the many events can be simultaneously monitored, as described in Section 3.2. Chapter 4 identifies methods for selecting PMC events for use as input features into the power model. However, this thesis also proposes the use of PMC events for other tasks, such as workload phase detection, PMC event prediction, and workload classification. Therefore, there needs to be a mechanism for extracting the most useful set of PMC events that are most suitable for all purposes simultaneously and sharing the information between these units. As demonstrated in Chapter 4, it is essential to minimise multicollinearity by making careful event selections and making transformations to the selected events to remove repeated information. For example, if the RTM algorithm requires a specific PMC event that has a high correlation to events that the power model is already using, the power model must either:

- Not use the new event (worst-case scenario);
- Make linear transformations between the new selection to remove multicollinearity (if possible)
- Reselect the existing (unconstrained) events to ones that compliment (and do not correlate with) the new event;

The requirements from the RTM, or other units that require information from the PMCs, may change at run-time, and therefore this process must be dynamic.

7.2.8 Validating Newer CPU Performance and Energy Models

The current example models in the gem5 simulator for Arm-based platforms are for the Arm Cortex-A7 and Arm Cortex-A15, which implement the ARMv7 architecture. The methodology for comparing simulator models to hardware platforms and identifying the sources of error can be used to develop models of newer ARMv8 CPUs implementing Arm DynamIQ. Furthermore, an up-to-date comparison with McPAT (similar to that in [48]) could be conducted, with a direct comparison between employing top-down empirical power models and high-level bottom-up approaches.

7.2.9 Applying Power Modelling to New Systems

Mobile processors are developing rapidly, becoming more powerful and employing new features. As highlighted and addressed in Chapter 3, there are challenges in obtaining reliable PMC data (and power measurements) from mobile platforms. An interesting avenue of future work would be to apply the power modelling methodologies to newer, 64-bit mobile platforms with Arm DynamIQ technology and machine learning accelerators. Furthermore, while the models are agnostic to the operating system, applying the models to Android while running typical smartphone workloads would further demonstrate the capabilities of the models.

A proposed setup would use *Workload Automation* [271] for automating the process of running realistic workloads, and the new *Simpleperf* API [109] for accessing PMC events. Furthermore, this would provide a platform for interfacing the models with EAS.

References

[1] A. Akram and L. Sawalha. A comparison of x86 computer architecture simulators. *Computer Architecture and Systems Research Laboratory (CASRL)*, July 2016.

[2] S. Alawnah and A. Sagahyroon. Modeling of smartphones' power using neural networks. *EURASIP Journal on Embedded Systems*, 2017(1):22, Feb 2017.

[3] S. Albers and A. Antoniadis. Race to idle: New algorithms for speed scaling with a sleep state. *ACM Trans. Algorithms*, 10(2):9:1–9:31, Feb. 2014.

[4] F. A. Ali, P. Simoens, T. Verbelen, P. Demeester, and B. Dhoedt. Mobile device power models for energy efficient dynamic offloading at runtime. *Journal of Systems and Software*, 113:173 – 187, 2016.

[5] S. Aminikhanghahi and D. J. Cook. A survey of methods for time series change point detection. *Knowledge and Information Systems*, 51(2):339–367, May 2017.

[6] Arm. CycleModels: Implementation Accurate Models of Arm IP. `https://developer.arm.com/products/system-design/cycle-models`. [Online; accessed 20-Nov-2017].

[7] Arm Developer. DynamIQ. `https://developer.arm.com/technologies/dynamiq`, 2018. [Online; accessed 08-March-2018].

[8] Arm Developer. Energy aware scheduling. `https://developer.arm.com/open-source/energy-aware-scheduling`, 2018. [Online; accessed 08-March-2018].

[9] ARM Holdings. Arm processor architecture. `https://www.arm.com/products/processors/instruction-set-architectures/index.php`, 2017. [Online; accessed 12-June-2017].

[10] Arm Ltd. Cortex-A8 technical reference manual r3p2. `http://infocenter.arm.com/help/topic/com.arm.doc.ddi0344k/DDI0344K_cortex_a8_r3p2_trm.pdf`, May 2010. [Online; accessed 04-October-2014].

[11] Arm Ltd. Dhrystone and mips performance of arm processors, 2011.

[12] Arm Ltd. Cortex-A9 technical reference manual r4p1. `http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0388e/index.html`, June 2012. [Online; accessed 07-December-2014].

[13] Arm Ltd. Cortex-A15 technical reference manual r4p0. `http://infocenter.arm.com/help/topic/com.arm.doc.ddi0438i/DDI0438I_cortex_a15_r4p0_trm.pdf`, June 2013. [Online; accessed 04-January-2015].

[14] Arm Ltd. Cortex-A7 technical reference manual r0p5. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0464f/index.html, April 2013. [Online; accessed 04-January-2015].

[15] Arm Ltd. White paper: big.LITTLE technology, the future of mobile. Technical report, 2013.

[16] Arm Ltd. Cortex-A53 technical reference manual r0p4. http://infocenter.arm.com/help/topic/com.arm.doc.ddi0501f/DDI0501F_cortex_a53_cryptography_trm.pdf, December 2015. [Online; accessed 16-March-2017].

[17] Arm Ltd. Cortex-A72 technical reference manual r0p3. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.100095_0003_06_en/index.html, December 2016. [Online; accessed 16-March-2017].

[18] Arm Ltd. Cortex-A75 technical reference manual r3p0. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.100403_0200_00_en/woi1483519316976.html, June 2017. [Online; accessed 10-February-2018].

[19] M. Asri, A. Pedram, L. K. John, and A. Gerstlauer. Simulator calibration for accelerator-rich architecture studies. In *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, pages 88–95, July 2016.

[20] J. Bai. Least squares estimation of a shift in linear processes. *Journal of Time Series Analysis*, 15(5):453–472, 1993.

[21] 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.

[22] M. Basseville and I. V. Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.

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

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

[25] L. Benini and G. d. Micheli. *Dynamic Power Management: Design Techniques and CAD Tools*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.

[26] R. Bertran, M. Gonzalez, X. Martorell, N. Navarro, and E. Ayguade. Decomposable and responsive power models for multicore processors using performance counters. In *Proceedings of the 24th ACM International Conference on Supercomputing*, ICS '10, pages 147–158, New York, NY, USA, 2010. ACM.

[27] 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.

[28] K. Bhatti, C. Belleudy, and M. Auguin. Power management in real time embedded systems through online and adaptive interplay of dpm and dvfs policies. In *2010 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing*, pages 184–191, Dec 2010.

[29] C. Bienia. *Benchmarking Modern Multiprocessors*. PhD thesis, Princeton University, January 2011.

[30] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti, R. Sen, K. Sewell, M. Shoaib, N. Vaish, M. D. Hill, and D. A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, Aug. 2011.

[31] N. L. Binkert, R. G. Dreslinski, L. R. Hsu, K. T. Lim, A. G. Saidi, and S. K. Reinhardt. The m5 simulator: Modeling networked systems. *IEEE Micro*, 26(4):52–60, July 2006.

[32] W. L. Bircher and L. John. Predictive power management for multi-core processors. In A. L. Varbanescu, A. Molnos, and R. van Nieuwpoort, editors, *Computer Architecture*, pages 243–255, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

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

[34] W. L. Bircher and L. K. John. Complete system power estimation using processor performance events. *IEEE Transactions on Computers*, 61(4):563–577, April 2012.

[35] W. L. Bircher, J. Law, M. Valluri, and L. K. John. Effective use of performance monitoring counters for run-time prediction of power. Technical report, The University of Texas at Austin, 2004.

[36] W. L. Bircher, M. Valluri, J. Law, and L. K. John. Runtime identification of microprocessor energy saving opportunities. In *ISLPED '05. Proceedings of the 2005 International Symposium on Low Power Electronics and Design, 2005.*, pages 275–280, Aug 2005.

[37] S. Bischoff, A. Hansson, and B. M. Al-Hashimi. Applying of quality of experience to system optimisation. In *2013 23rd International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 91–98, Sept 2013.

[38] C. Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, 1 edition, 2006.

[39] B. Black and J. P. Shen. Calibration of microprocessor performance models. *Computer*, 31:59–65, 1998.

[40] E. Blem, J. Menon, and K. Sankaralingam. Power struggles: Revisiting the risc vs. cisc debate on contemporary arm and x86 architectures. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 1–12, Feb 2013.

[41] G. E. P. Box and D. R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society. Series B (Methodological)*, 26(2):211–252, 1964.

[42] D. Brand and C. Visweswariah. Inaccuracies in power estimation during logic synthesis. In *Proceedings of International Conference on Computer Aided Design*, pages 388–394, Nov 1996.

[43] L. Breiman and J. H. Friedman. Estimating optimal transformations for multiple regression and correlation. *Journal of the American statistical Association*, 80(391):580–598, 1985.

[44] D. Brooks, V. Tiwari, and M. Martonosi. Wattch: a framework for architectural-level power analysis and optimizations. In *Proceedings of 27th International Symposium on Computer Architecture (IEEE Cat. No.RS00201)*, pages 83–94, June 2000.

[45] M. Bruno, A. Macii, and M. Poncino. Rtl power estimation in an hdl-based design flow. *IEE Proceedings - Computers and Digital Techniques*, 152(6):723–730, Nov 2005.

[46] T. D. Burd, T. A. Pering, A. J. Stratakos, and R. W. Brodersen. A dynamic voltage scaled microprocessor system. *IEEE Journal of Solid-State Circuits*, 35(11):1571–1580, Nov 2000.

[47] D. Burger and T. M. Austin. The simplescalar tool set, version 2.0. *SIGARCH Comput. Archit. News*, 25(3):13–25, June 1997.

[48] A. Butko, F. Bruguier, A. Gamatié, G. Sassatelli, D. Novo, L. Torres, and M. Robert. Full-system simulation of big.little multicore architecture for performance and energy exploration. In *2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip (MCSOC)*, pages 201–208, Sept 2016.

[49] A. Butko, A. Gamatie, G. Sassatelli, L. Torres, and M. Robert. Design exploration for next generation high-performance manycore on-chip systems: Application to big.little architectures. In *2015 IEEE Computer Society Annual Symposium on VLSI*, pages 551–556, July 2015.

[50] A. Butko, R. Garibotti, L. Ost, and G. Sassatelli. Accuracy evaluation of gem5 simulator system. In *7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*, pages 1–7, July 2012.

[51] P. F. Butzen and R. P. Ribas. Leakage current in sub-micrometer cmos gates. Technical report, Universidade Federal do Rio Grande do Sul, 2006.

[52] B. H. Calhoun, F. A. Honore, and A. P. Chandrakasan. A leakage reduction methodology for distributed mtcmos. *IEEE Journal of Solid-State Circuits*, 39(5):818–826, May 2004.

[53] A. Carroll and G. Heiser. Unifying dvfs and offlining in mobile multicores. In *2014 IEEE 19th Real-Time and Embedded Technology and Applications Symposium (RTAS)*, pages 287–296, April 2014.

[54] S. A. Carvalho, D. C. Cunha, and A. G. Silva-Filho. Autonomous power management in mobile devices using dynamic frequency scaling and reinforcement learning for energy minimization. *Microprocessors and Microsystems*, 64:205 – 220, 2019.

[55] S. Catalan, R. Rodriguez-Sanchez, E. S. Quintana-Orti, and J. R. Herrero. Static versus dynamic task scheduling of the lu factorization on arm big. little architectures. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 733–742, May 2017.

[56] R. Cataldo, G. Korol, R. Fernandes, D. Matos, and C. Marcon. Architectural exploration of last-level caches targeting homogeneous multicore systems. In *2016 29th Symposium on Integrated Circuits and Systems Design (SBCCI)*, pages 1–6, Aug 2016.

[57] A. Cerioli and D. di Economia. K-means cluster analysis and mahalanobis metrics: A problematic match or an overlooked opportunity? In *Statistica Applicata*, volume 17, 2005.

[58] M. Chadha, T. Ilsche, M. Bielert, and W. E. Nagel. A statistical approach to power estimation for x86 processors. In *2017 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, pages 1012–1019, May 2017.

[59] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen. Low-power cmos digital design. *IEEE Journal of Solid-State Circuits*, 27(4):473–484, Apr 1992.

[60] K. Chandrasekar, B. Akesson, and K. Goossens. Improved power modeling of ddr sdrams. In *2011 14th Euromicro Conference on Digital System Design*, pages 99–108, Aug 2011.

[61] L. Chang, Z. Wang, Y. Gao, W. Kang, Y. Zhang, and W. Zhao. Evaluation of spin-hall-assisted stt-mram for cache replacement. In *2016 IEEE/ACM International Symposium on Nanoscale Architectures (NANOARCH)*, pages 73–78, July 2016.

[62] T. Chantem, Y. Xiang, X. S. Hu, and R. P. Dick. Enhancing multicore reliability through wear compensation in online assignment and scheduling. In *2013 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1373–1378, March 2013.

[63] S. Chatterjee and B. Price. *Regression Analysis by Example*. John Wiley and Sons, 2 edition, 1991.

[64] A. Chatzidimitriou and D. Gizopoulos. Anatomy of microarchitecture-level reliability assessment: Throughput and accuracy. In *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 69–78, April 2016.

[65] K. Chen, B. Varghese, P. Kilpatrick, and D. S. Nikolopoulos. Power modelling for heterogeneous cloud-edge data centers. *CoRR*, abs/1710.10325, 2017.

[66] Q. Chen, H. Yi, Y. Hu, X. Xu, and X. Li. A new method of selecting k-means initial cluster centers based on hotspot analysis. In *2018 26th International Conference on Geoinformatics*, pages 1–6, June 2018.

[67] Z. Chen, C. Diaz, J. D. Plummer, M. Cao, and W. Greene. 0.18 um dual vt mosfet process and energy-delay measurement. In *International Electron Devices Meeting. Technical Digest*, pages 851–854, Dec 1996.

[68] K. Chronaki, M. Moretó, M. Casas, A. Rico, R. M. Badia, E. Ayguadé, J. Labarta, and M. Valero. Poster: Exploiting asymmetric multi-core processors with flexible system software. In *2016 International Conference on Parallel Architecture and Compilation Techniques (PACT)*, pages 415–417, Sept 2016.

[69] S. Clement, D. McKee, and J. Xu. A service-oriented co-simulation: Holistic data center modelling using thermal, power and computational simulations, December 2017.

[70] G. Contreras and M. Martonosi. Power prediction for intel xscale/spl reg/ processors using performance monitoring unit events. In *ISLPED '05. Proceedings of the 2005 International Symposium on Low Power Electronics and Design, 2005.*, pages 221–226, Aug 2005.

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

[72] H. J. Curnow and B. A. Wichmann. A synthetic benchmark. In *The Computer Journal*, volume 19, pages 43–49, 1976.

[73] A. Das, B. M. Al-Hashimi, and G. V. Merrett. Adaptive and hierarchical runtime manager for energy-aware thermal management of embedded systems. *ACM Trans. Embed. Comput. Syst.*, 15(2):24:1–24:25, Jan. 2016.

[74] A. Das, G. V. Merrett, and B. M. Al-Hashimi. The slowdown or race-to-idle question: Workload-aware energy optimization of smt multicore platforms under process variation. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 535–538, March 2016.

[75] A. Das, R. A. Shafik, G. V. Merrett, B. M. Al-Hashimi, A. Kumar, and B. Veeravalli. Reinforcement learning-based inter- and intra-application thermal optimization for lifetime improvement of multicore systems. In *2014 51st ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2014.

[76] A. Das, M. J. Walker, A. Hansson, B. M. Al-Hashimi, and G. V. Merrett. Hardware-software interaction for run-time power optimization: A case study of embedded linux on multicore smartphones. In *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 165–170, July 2015.

[77] R. E. W. David A. Belsley, Edwin Kuh. *Regression Diagnostics: Identifying Influential Data and Sources of Collinearity*. John Wiley and Sons, new edition edition, July 2004.

[78] R. de Jong and A. Sandberg. Nomali: Simulating a realistic graphics driver stack using a stub gpu. In *2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 255–262, April 2016.

[79] B. S. Deepaksubramanyan and A. Nunez. Analysis of subthreshold leakage reduction in cmos digital circuits. In *2007 50th Midwest Symposium on Circuits and Systems*, pages 1400–1404, Aug 2007.

[80] A. S. Dhodapkar and J. E. Smith. Comparing program phase detection techniques. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 36, pages 217–, Washington, DC, USA, 2003. IEEE Computer Society.

[81] R. P. Dick. Reliability, thermal, and power modeling and optimization. In *2010 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 181–184, Nov 2010.

[82] K. E. Dierckens, A. B. Harrison, C. K. Leung, and A. V. Pind. A data science and engineering solution for fast k-means clustering of big data. In *2017 IEEE Trustcom/BigDataSE/ICESS*, pages 925–932, Aug 2017.

[83] W. Ding, D. Guttman, and M. Kandemir. Compiler support for optimizing memory bank-level parallelism. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 571–582, Dec 2014.

[84] O. Djedidi, M. Djeziri, N. M'Sirdi, and N. Aziz. A novel easy-to-construct power model for embedded and mobile systems - using recursive neural nets to estimate power consumption of arm-based embedded systems and mobile devices. In *15th International Conference on Informatics in Control, Automation and Robotics*, pages 541–545, 01 2018.

[85] O. Djedidi, M. A. Djeziri, N. K. M'Sirdi, and A. Naamane. Constructing an accurate and a high-performance power profiler for embedded systems and smartphones. In *Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, MSWIM '18, pages 79–82, New York, NY, USA, 2018. ACM.

[86] X. Dong, C. Xu, Y. Xie, and N. P. Jouppi. Nvsim: A circuit-level performance, energy, and area model for emerging nonvolatile memory. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 31(7):994–1007, July 2012.

[87] B. Donyanavard, T. Mück, S. Sarma, and N. Dutt. Sparta: Runtime task allocation for energy efficient heterogeneous manycores. In *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 1–10, Oct 2016.

[88] R. C. Dorf. *The Technology Management Handbook*. CRC Press, 1 edition, July 1998.

[89] N. R. Draper and H. Smith. *Applied Regression Analysis*. Wiley Series in Probability and Statistics. Wiley-Blackwell, 3 edition, May 1998.

[90] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern Classification*. Wiley-Interscience, New York, 2 edition, 2001.

[91] F. A. Endo, D. Couroussé, and H. P. Charles. Micro-architectural simulation of in-order and out-of-order arm microprocessors with gem5. In *2014 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS XIV)*, pages 266–273, July 2014.

[92] D. Ernst, N. S. Kim, S. Das, S. Pant, R. Rao, T. Pham, C. Ziesler, D. Blaauw, T. Austin, K. Flautner, and T. Mudge. Razor: a low-power pipeline based on circuit-level timing speculation. In *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, pages 7–18, Dec 2003.

[93] X. Fan, Y. Sui, and J. Xue. Contention-aware scheduling for asymmetric multicore processors. In *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*, pages 742–751, Dec 2015.

[94] C. Fang and J. Ma. A novel k'-means algorithm for clustering analysis. In *2009 2nd International Conference on Biomedical Engineering and Informatics*, pages 1–5, Oct 2009.

[95] M. A. Faruque, J. Jahn, T. Ebi, and J. Henkel. Runtime thermal management using software agents for multi- and many-core architectures. *IEEE Design Test of Computers*, 27(6):58–68, Nov 2010.

[96] S. Feng, S. Gupta, A. Ansari, and S. Mahlke. Maestro: Orchestrating lifetime reliability in chip multiprocessors. In *Proceedings of the 5th International Conference on High Performance Embedded Architectures and Compilers*, HiPEAC'10, pages 186–200, Berlin, Heidelberg, 2010. Springer-Verlag.

[97] P. Flach. *Machine Learning: The Art and Science of Algorithms that Make Sense of Data*. Cambridge University Press, 2012.

[98] K. Flautner, D. Flynn, D. Roberts, and D. I. Patel. Iem926: An energy efficient soc with dynamic voltage scaling. In *DATE*. IEEE Computer Society, 2004.

[99] E. J. Fluhr, S. Baumgartner, D. Boerstler, J. F. Bulzacchelli, T. Diemoz, D. Dreps, G. English, J. Friedrich, A. Gattiker, T. Gloekler, C. Gonzalez, J. D. Hibbeler, K. A. Jenkins, Y. Kim, P. Muench, R. Nett, J. Paredes, J. Pille, D. Plass, P. Restle, R. Robertazzi, D. Shan, D. Siljenberg, M. Sperling, K. Stawiasz, G. Still, Z. Toprak-Deniz, J. Warnock, G. Wiedemeier, and V. Zyuban. The 12-core power8$^{\text{TM}}$ processor with 7.6 tb/s io bandwidth, integrated voltage regulation, and resonant clocking. *IEEE Journal of Solid-State Circuits*, 50(1):10–23, Jan 2015.

[100] Forbes. Arm holdings and qualcomm: The winners in mobile. `https://www.forbes.com/sites/greatspeculations/2016/08/29/intels-agreement-with-arm-to-boost-its-foundry-business-for-leading-edge-products/#250fd62d539f`, February 2013. [Online; accessed 12-June-2017].

[101] C. Ford. Understanding q-q plots. `https://data.library.virginia.edu/understanding-q-q-plots/`, August 2015. [Online; accessed 22-July-2018].

[102] J. Fox. *Applied Regression Analysis, Linear Models, and Related Methods*. SAGE Publications, 1997.

[103] E. W. Frees. *Regression Modeling with Actuarial and Financial Applications*. Feb 2010.

[104] R. J. Freund and P. D. Minton. *Regression Methods: A Tool for Data Analysis*, volume 30 of *Statistics, textbooks and monographs*. Marcel Dekker Inc., New York, 1979.

[105] F. Gaspar, L. Tanica, P. Tomas, A. Ilic, and L. Sousa. Attaining performance fairness in big.little systems. In *2015 12th International Workshop on Intelligent Solutions in Embedded Systems (WISES)*, pages 67–72, Oct 2015.

[106] gem5. The gem5 Simulator. `http://www.gem5.org`. [Online; accessed 16-Nov-2017].

[107] B. Goel and S. A. McKee. A methodology for modeling dynamic and static power consumption for multicore processors. In *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 273–282, May 2016.

[108] R. Gonzalez and M. Horowitz. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits*, 31(9):1277–1284, Sep 1996.

[109] Google. SimplePerf. `https://android.googlesource.com/platform/system/extras/+/master/simpleperf/`, 2018. [Online; accessed 8-July-2018].

[110] P. E. Green, J. Kim, and F. J. Carmone. A preliminary study of optimal variable weighting in k-means clustering. *Journal of Classification*, 7(2):271–285, Sep 1990.

[111] P. J. GREEN. Reversible jump markov chain monte carlo computation and bayesian model determination. *Biometrika*, 82(4):711–732, 1995.

[112] P. Greenhalgh. Big.LITTLE processing with ARM Cortex-A15 and Cortex-A7. Technical report, Arm Ltd, September 2011.

[113] D. Gujarati. *Basic econometrics*. Economic series. McGraw Hill, 2003.

[114] Y. Guo, P. Narayanan, M. A. Bennaser, S. Chheda, and C. A. Moritz. Energy-efficient hardware data prefetching. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 19(2):250–263, Feb 2011.

[115] U. Gupta, C. Patil, G. Bhat, P. Mishra, and U. Ogras. Dypo: Dynamic pareto-optimal configuration selection for heterogeneous mpsocs. *ACM Transactions on Embedded Computing Systems*, 16(5s), 9 2017.

[116] 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.

[117] A. Gutierrez, J. Pusdesris, R. Dreslinski, T. Mudge, C. Sudanthi, C. Emmons, M. Hayenga, and N. Paver. Sources of error in full-system simulation, 03 2014.

[118] M. Gutierrez, S. Rahman, D. Tamir, and A. Qasem. Neural network methods for fast and portable prediction of CPU power consumption. In *2015 Sixth International Green and Sustainable Computing Conference (IGSC)*, pages 1–4, Dec 2015.

[119] H. Hanson, S. W. Keckler, S. Ghiasi, K. Rajamani, F. Rawson, and J. Rubio. Thermal response to dvfs: analysis with an intel pentium m. In *Low Power Electronics and Design (ISLPED), 2007 ACM/IEEE International Symposium on*, pages 219–224, Aug 2007.

[120] A. Hansson, N. Agarwal, A. Kolli, T. Wenisch, and A. N. Udipi. Simulating dram controllers for future system architecture exploration. In *2014 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 201–210, March 2014.

[121] Hardkernel. ODROID-XU3. `http://www.hardkernel.com`, 2013. [Online; accessed 30-Nov-2017].

[122] A. F. Hayes and L. Cai. Using heteroskedasticity-consistent standard error estimators in ols regression: An introduction and software implementation. *Behavior Research Methods*, 39(4):709–722, Nov 2007.

[123] M. A. Hoque, M. Siekkinen, K. N. Khan, Y. Xiao, and S. Tarkoma. Modeling, profiling, and debugging the energy consumption of mobile devices. *ACM Computing Surveys (CSUR)*, 48(3):39, 2016.

[124] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of educational psychology*, 24(6):417, 1933.

[125] Z. Hu, A. Buyuktosunoglu, V. Srinivasan, V. Zyuban, H. Jacobson, and P. Bose. Microarchitectural techniques for power gating of execution units. In *Proceedings of the 2004 International Symposium on Low Power Electronics and Design*, ISLPED '04, pages 32–37, New York, NY, USA, 2004. ACM.

[126] C. X. Huang, B. Zhang, A.-C. Deng, and B. Swirski. The design and implementation of powermill. In *Proceedings of the 1995 International Symposium on Low Power Design*, ISLPED '95, pages 105–110, New York, NY, USA, 1995. ACM.

[127] W. Huang, S. Ghosh, S. Velusamy, K. Sankaranarayanan, K. Skadron, and M. R. Stan. Hotspot: a compact thermal modeling methodology for early-stage vlsi design. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 14(5):501–513, May 2006.

[128] H. Im, T. Inukai, H. Gomyo, T. Hiramoto, and T. Sakurai. Vtcmos characteristics and its optimum conditions predicted by a compact analytical model. In *Low Power Electronics and Design, International Symposium on, 2001.*, pages 123–128, 2001.

[129] C. Imes, D. H. K. Kim, M. Maggio, and H. Hoffmann. Poet: a portable approach to minimizing energy under soft real-time constraints. In *21st IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 75–86, April 2015.

[130] Intel Corporation. White paper: Measuring processor power, TDP vs. ACP. Technical report, April 2011.

[131] Intel Corporation. Intel 64 and IA-32 architectures software developer's manual. `https://software.intel.com/en-us/download/intel-64-and-ia-32-architectures-sdm-combined-volumes-1-2a-2b-2c-2d-3a-3b-3c-3d-and-4`, May 2018. [Online; accessed 19-July-2018].

[132] S. M. Z. Iqbal, Y. Liang, and H. Grahn. Parmibench - an open-source benchmark for embedded multiprocessor systems. *IEEE Comput. Archit. Lett.*, 9(2):45–48, July 2010.

[133] A. Iranfar, S. N. Shahsavani, M. Kamal, and A. Afzali-Kusha. A heuristic machine learning-based algorithm for power and thermal management of heterogeneous mpsocs. In *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 291–296, July 2015.

[134] d. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '03, pages 37–46, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

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

[136] C. Isci and M. Martonosi. Phase characterization for power: evaluating control-flow-based and event-counter-based techniques. In *The Twelfth International Symposium on High-Performance Computer Architecture, 2006.*, pages 121–132, Feb 2006.

[137] R. Jagtap, S. Diestelhorst, A. Hansson, M. Jung, and N. When. Exploring system performance using elastic traces: Fast, accurate and portable. In *2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS)*, pages 96–105, July 2016.

[138] S. Jain, H. Navale, U. Ogras, and S. Garg. Energy efficient scheduling for web search on heterogeneous microservers. In *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 177–182, July 2015.

[139] B. Jeff. Enabling mobile innovation with the cortextm-a7 processor. `https://community.arm.com/processors/b/blog/posts/witepaper-enabling-mobile-innovation-with-the-cortex-a7-processor`, October 2011. [Online; accessed 13-June-2017].

[140] B. Jeff. big.LITTLE technology moves towards fully heterogeneous Global Task Scheduling. `https://www.arm.com/files/pdf/big_LITTLE_technology_moves_towards_fully_heterogeneous_Global_Task_Scheduling.pdf`, November 2013. [Online; accessed 13-June-2017].

[141] J. Jeffers, J. Reinders, and A. Sodani. *Intel Xeon Phi Processor High Performance Programming: Knights Landing Edition*. Elsevier Science, 2016.

[142] J. Jeong, C. H. Park, J. Huh, and S. Maeng. Efficient hardware-assisted logging with asynchronous and direct-update for persistent memory. In *The 51st Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 51, 2018.

[143] Jianchang Mao and A. K. Jain. A self-organizing network for hyperellipsoidal clustering (hec). *IEEE Transactions on Neural Networks*, 7(1):16–29, Jan 1996.

[144] I. Jolliffe. *Principal Component Analysis*, pages 1094–1096. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[145] R. Joseph and M. Martonosi. Run-time power estimation in high performance microprocessors. In *ISLPED'01: Proceedings of the 2001 International Symposium on Low Power Electronics and Design (IEEE Cat. No.01TH8581)*, pages 135–140, Aug 2001.

[146] J. F. H. Jr, W. C. Black, B. J. Babin, and R. E. Anderson. *Multivariate Data Analysis*. Prentice Hall, 7 edition, February 2009.

[147] Karthik Chandrasekar and Christian Weis and Yonghui Li and Sven Goossens and Matthias Jung and Omar Naji and Benny Akesson and Norbert Wehn and Kees Goossens. DRAMPower: Open-source DRAM Power & Energy Estimation Tool. `http://www.drampower.info`. [Online; accessed 16-Nov-2017].

[148] S. Kaxiras and M. Martonosi. *Computer Architecture Techniques for Power-Efficiency*. Morgan and Claypool Publishers, 1st edition, 2008.

[149] M. Keating, D. Flynn, R. Aitken, A. Gibbons, and K. Shi. *Low Power Methodology Manual: For System-on-Chip Design*. Springer Publishing Company, Incorporated, 2007.

[150] G. Kestor, R. Gioiosa, D. J. Kerbyson, and A. Hoisie. Quantifying the energy cost of data movement in scientific applications. *2013 IEEE International Symposium on Workload Characterization (IISWC)*, 0:56–65, 2013.

[151] D. H. K. Kim, C. Imes, and H. Hoffmann. Racing and pacing to idle: Theoretical and empirical analysis of energy optimization heuristics. In *2015 IEEE 3rd International Conference on Cyber-Physical Systems, Networks, and Applications*, pages 78–85, Aug 2015.

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

[153] N. Kim, T. Austin, D. Baauw, T. Mudge, K. Flautner, J. Hu, M. Irwin, M. Kandemir, and V. Narayanan. Leakage current: Moore's law meets static power. *Computer*, 36(12):68–75, Dec 2003.

[154] S. Kim, S. Kim, R. M. Kil, and H. Y. Youn. Cgroup-aware load balancing in heterogeneous multi-processor scheduler. In *2016 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 21–26, Oct 2016.

[155] T. Kim, Z. Sun, H. B. Chen, H. Wang, and S. X. D. Tan. Energy and lifetime optimizations for dark silicon manycore microprocessor considering both hard and soft errors. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 25(9):2561–2574, Sept 2017.

[156] Y. Kim, P. Mercati, A. More, E. Shriver, and T. Rosing. P4: Phase-based power/performance prediction of heterogeneous systems via neural networks. In *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pages 683–690, Nov 2017.

[157] Y. Kim, F. Paterna, S. Tilak, and T. S. Rosing. Smartphone analysis and optimization based on user activity recognition. In *Proceedings of the IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '15, pages 605–612, Piscataway, NJ, USA, 2015. IEEE Press.

[158] E. Kofman and R. de Simone. A formal approach to the mapping of tasks on an heterogenous multicore, energy-aware architecture. In *2016 ACM/IEEE International Conference on Formal Methods and Models for System Design (MEMOCODE)*, pages 153–162, Nov 2016.

[159] R. Kumar, K. I. Farkas, N. P. Jouppi, P. Ranganathan, and D. M. Tullsen. Single-isa heterogeneous multi-core architectures: the potential for processor power reduction. In *Proceedings. 36th Annual IEEE/ACM International Symposium on Microarchitecture, 2003. MICRO-36.*, pages 81–92, Dec 2003.

[160] T. Kuroda, T. Fujita, S. Mita, T. Nagamatsu, S. Yoshioka, K. Suzuki, F. Sano, M. Norishima, M. Murota, M. Kako, M. Kinugawa, M. Kakumu, and T. Sakurai. A 0.9-v, 150-mhz, 10-mw, 4 mm2, 2-d discrete cosine transform core processor with variable threshold-voltage (vt) scheme. *IEEE Journal of Solid-State Circuits*, 31(11):1770–1779, Nov 1996.

[161] M. Kutner. *Applied Linear Statistical Models*. McGraw-Hill international edition. McGraw-Hill Irwin, 2005.

[162] M. Kutner, C. Nachtsheim, and J. Neter. *Applied Linear Regression Models*. McGraw-Hill Education, 4 edition, January 2004.

[163] T. Lanier. Exploring the design of the cortex-a15 processor. `https://www.arm.com/files/pdf/AT-Exploring_the_Design_of_the_Cortex-A15.pdf`, 2013. [Online; accessed 4-January-2017].

[164] C. Lasheng and L. Yuqiang. Improved initial clustering center selection algorithm for k-means. In *2017 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pages 275–279, Sep. 2017.

[165] J. Lau, S. Schoemackers, and B. Calder. Structures for phase classification. In *IEEE International Symposium on - ISPASS Performance Analysis of Systems and Software, 2004*, pages 57–67, March 2004.

[166] M. Lavielle. Detection of change points in a time series. `http://sia.webpopix.org/changePoints.html`, March 2017. [Online; accessed 19-September-2017].

[167] M. Lavielle and E. Lebarbier. An application of mcmc methods for the multiple change-points problem. *Signal Processing*, 81(1):39 – 53, 2001. Special section on Markov Chain Monte Carlo (MCMC) Methods for Signal Processing.

[168] M. Lavielle and G. Teyssière. Detection of multiple change-points in multivariate time series. *Lithuanian Mathematical Journal*, 46(3):287–306, Jul 2006.

[169] C. Lee, M. Potkonjak, and W. Mangione-Smith. Mediabench: a tool for evaluating and synthesizing multimedia and communications systems. In *Microarchitecture, 1997. Proc., 13th Annu. IEEE/ACM Int. Symp.*, pages 330–335, Dec 1997.

[170] K. Lee and K. Skadron. Using performance counters for runtime temperature sensing in high-performance processors. In *19th IEEE International Parallel and Distributed Processing Symposium*, pages 8 pp.–, April 2005.

[171] W. Lee, Y. Kim, J. H. Ryoo, D. Sunwoo, A. Gerstlauer, and L. K. John. Powertrain: A learning-based calibration of mcpat power models. In *2015 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, pages 189–194, July 2015.

[172] M. J. Li, M. K. Ng, Y. Cheung, and J. Z. Huang. Agglomerative fuzzy k-means clustering algorithm with selection of number of clusters. *IEEE Transactions on Knowledge and Data Engineering*, 20(11):1519–1534, Nov 2008.

[173] M.-L. Li, R. Sasanka, S. V. Adve, Y.-K. Chen, and E. Debes. The alpbench benchmark suite for complex multimedia applications. In *Workload Characterization Symposium, 2005. Proceedings of the IEEE International*, pages 34–45, Oct 2005.

[174] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi. Mcpat: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In *Proceedings of the 42Nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO 42, pages 469–480, New York, NY, USA, 2009. ACM.

[175] X. Li, G. Chen, and W. Wen. Energy-efficient execution for repetitive app usages on big.little architectures. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2017.

[176] Y. Li, W. Shi, C. Jiang, J. Zhang, and J. Wan. Energy efficiency analysis of heterogeneous platforms: Early experiences. In *2016 Seventh International Green and Sustainable Computing Conference (IGSC)*, pages 1–6, Nov 2016.

[177] C. Lively, X. Wu, V. Taylor, S. Moore, H.-C. Chang, C.-Y. Su, and K. Cameron. Power-aware predictive models of hybrid (mpi/openmp) scientific applications on multicore systems. *Computer Science - Research and Development*, 27(4):245–253, Nov 2012.

[178] J. S. Long and L. H. Ervin. Using heteroscedasticity consistent standard errors in the linear regression model. *The American Statistician*, 54(3):pp. 217–224, 2000.

[179] R. Longbottom. Roy longbottom's pc benchmark collection. `http://www.roylongbottom.org.uk`, Sept 2014. [Online; accessed 2-Jun-2015].

[180] A. Ltd. *CortexTM-A15 MPCore*, r2p0 edition, 2011.

[181] Y. Lu, J. Shu, L. Sun, and O. Mutlu. Loose-ordering consistency for persistent memory. In *2014 IEEE 32nd International Conference on Computer Design (ICCD)*, pages 216–223, Oct 2014.

[182] R. MacAusland. The moore-penrose inverse and least squares. 2014.

[183] J. G. MacKinnon and H. White. Some heteroskedasticity-consistent covariance matrix estimators with improved finite sample properties. *Journal of Econometrics*, 29(3):305–325, Sept. 1985.

[184] J. Mair, Z. Huang, and D. Eyers. Manila: Using a densely populated pmc-space for power modelling within large-scale systems. *Parallel Computing*, 82:37 – 56, 2019. Special Issue on Parallel Programming Models and Systems Software.

[185] J. Mair, Z. Huang, D. Eyers, and H. Zhang. Pmc-based power modelling with workload classification on multicore systems. In *2014 43rd International Conference on Parallel Processing Workshops*, pages 129–138, Sept 2014.

[186] N. Mankiw. A quick refresher course in macroeconomics. *Journal of Economic Literature*, 28(Dec):1645–1660, 1990.

[187] M. M. K. Martin, D. J. Sorin, B. M. Beckmann, M. R. Marty, M. Xu, A. R. Alameldeen, K. E. Moore, M. D. Hill, and D. A. Wood. Multifacet's general execution-driven multiprocessor simulator (gems) toolset. *SIGARCH Comput. Archit. News*, 33(4):92–99, Nov. 2005.

[188] A. Martino, A. Ghiglietti, F. Ieva, and A. M. Paganoni. A k-means procedure based on a mahalanobis type distance for clustering multivariate functional data. *Statistical Methods & Applications*, pages 1–22, 2017.

[189] L. McVoy and C. Staelin. LAT MEM RD Manual Page. `http://lmbench.sourceforge.net/man/lat_mem_rd.8.html`, 1996. [Online; accessed 5-July-2015].

[190] L. McVoy and C. Staelin. Lmbench: Portable tools for performance analysis. In *Proc. of the 1996 Annu. Conf. on USENIX Annual Technical Conference*, ATEC '96, pages 23–23, Berkeley, CA, USA, 1996. USENIX Association.

[191] M. Meixner and T. G. Noll. Limits of gate-level power estimation considering real delay effects and glitches. In *2014 International Symposium on System-on-Chip (SoC)*, pages 1–7, Oct 2014.

[192] I. Melnykov and V. Melnykov. On k-means algorithm with the use of mahalanobis distances. *Statistics and Probability Letters*, 84:88 – 95, 2014.

[193] U. Memory. Unaligned data access. `https://www.heyrick.co.uk/armwiki/Unaligned_data_access`, December 2012. [Online; accessed 14-June-2017].

[194] P. Mercati, F. Paterna, A. Bartolini, L. Benini, and T. Š. Rosing. Warm: Workload-aware reliability management in linux/android. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(9):1557–1570, Sept 2017.

[195] F. J. Mesa-Martinez, E. K. Ardestani, and J. Renau. Characterizing processor thermal behavior. In *Proceedings of the Fifteenth Edition of ASPLOS on Architectural Support for Programming Languages and Operating Systems*, ASPLOS XV, pages 193–204, New York, NY, USA, 2010. ACM.

[196] Micron. System Power Calculators. `https://www.micron.com/support/tools-and-utilities/power-calc`, 2007. [Online; accessed 06-Nov-2017].

[197] Minitab Blog Editor. Why you need to check your residual plots for regression analysis. `http://blog.minitab.com/blog/adventures-in-statistics-2/why-you-need-to-check-your-residual-plots-for-regression-analysis`, April 2012. [Online; accessed 19-May-2016].

[198] T. M. Mitchell. *Machine Learning*. McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[199] C. Möbius, W. Dargie, and A. Schill. Power consumption estimation models for processors, virtual machines, and servers. *IEEE Transactions on Parallel and Distributed Systems*, 25(6):1600–1614, June 2014.

[200] A. Mohammad, U. Darbaz, G. Dozsa, S. Diestelhorst, D. Kim, and N. S. Kim. dist-gem5: Distributed simulation of computer clusters. In *2017 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 153–162, April 2017.

[201] D. Montgomery, E. Peck, and G. Vining. *Introduction to Linear Regression Analysis*. Wiley Series in Probability and Statistics. Wiley, 2015.

[202] Mu-Chun Su and Chien-Hsing Chou. A modified version of the k-means algorithm with a distance based on cluster symmetry. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(6):674–680, June 2001.

[203] T. R. Muck, Z. Ghaderi, N. D. Dutt, and E. Bozorgzadeh. Exploiting heterogeneity for aging-aware load balancing in mobile platforms. *IEEE Transactions on Multi-Scale Computing Systems*, 3(1):25–35, Jan 2017.

[204] S. Mukhopadhyay, A. Raychowdhury, and K. Roy. Accurate estimation of total leakage in nanometer-scale bulk cmos circuits based on device geometry and doping profile. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 24(3):363–381, March 2005.

[205] T. S. Muthukaruppan, M. Pricopi, V. Venkataramani, T. Mitra, and S. Vishin. Hierarchical power management for asymmetric multi-core in dark silicon era. In *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–9, May 2013.

[206] S. Mutoh, T. Douseki, Y. Matsuya, T. Aoki, S. Shigematsu, and J. Yamada. 1-v power supply high-speed digital circuit technology with multithreshold-voltage cmos. *IEEE Journal of Solid-State Circuits*, 30(8):847–854, Aug 1995.

[207] J. Myers, A. Savanth, R. Gaddh, D. Howard, P. Prabhat, and D. Flynn. A subthreshold arm cortex-m0+ subsystem in 65 nm cmos for wsn applications with 14 power domains, 10t sram, and integrated voltage regulator. *IEEE Journal of Solid-State Circuits*, 51(1):31–44, Jan 2016.

[208] J. Nagler. Notes on ordinary least squares estimates. `https://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/13/lecture-13.pdf`, January 2001. [Online; accessed 14-August-2018].

[209] S. Narendra, S. Borkar, V. De, D. Antoniadis, and A. Chandrakasan. Scaling of stack effect and its application for leakage reduction. In *ISLPED'01: Proceedings of the 2001 International Symposium on Low Power Electronics and Design (IEEE Cat. No.01TH8581)*, pages 195–200, Aug 2001.

[210] R. Nau. Statistical forecasting: notes on regression and time series analysis. `http://people.duke.edu/~rnau/411home.htm`, June 2018. [Online; accessed 22-July-2018].

[211] C. Neau and K. Roy. Optimal body bias selection for leakage improvement and process compensation over different technology generations. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design, 2003. ISLPED '03.*, pages 116–121, Aug 2003.

[212] A. Ng. Coursera: Machine learning. `https://www.coursera.org/learn/machine-learning`, 2018. [Online; accessed 10-Mar-2018].

[213] A. Ng, J. Ngiam, C. Y. Foo, Y. Mai, C. Suen, A. Coates, A. Maas, A. Hannun, B. Huval, T. Wang, and S. Tandon. Stanford UFLDL: PCA whitening. `http://ufldl.stanford.edu/tutorial/unsupervised/PCAWhitening/`, 2013. [Online; accessed 22-May-2019].

[214] K. Nikov, J. L. Nunez-Yanez, and M. Horsnell. Evaluation of hybrid run-time power models for the arm big.little architecture. In *2015 IEEE 13th International Conference on Embedded and Ubiquitous Computing*, pages 205–210, Oct 2015.

[215] M. Novaes, V. Petrucci, A. Gamatié, and F. M. Q. Pereira. Compiler-assisted adaptive program scheduling in big.little systems. *CoRR*, abs/1903.07038, 2019.

[216] T. Nowatzki, J. Menon, C. han Ho, and K. Sankaralingam. gem5, gpgpusim, mcpat, gpuwattch, "your favorite simulator here" considered harmful, 2014.

[217] 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.

[218] T. Odajima, Y. Kodama, and M. Sato. Power performance analysis of arm scalable vector extension. In *2018 IEEE Symposium in Low-Power and High-Speed Chips (COOL CHIPS)*, pages 1–3, April 2018.

[219] F. M. of Education and Research. likwid: Performance monitoring and benchmarking suite. `https://github.com/RRZE-HPC/likwid/`, Nov 2017. [Online; accessed 28-Nov-2017].

[220] Open Virtual Platforms. Technology OVPsim. `http://www.ovpworld.org/technology_ovpsim`. [Online; accessed 16-Nov-2017].

[221] S. Padmanabha, A. Lukefahr, R. Das, and S. Mahlke. Dynamos: Dynamic schedule migration for heterogeneous cores. In *2015 48th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 322–333, Dec 2015.

[222] D. Pandiyan and C. Wu. Quantifying the energy cost of data movement for emerging smart phone workloads on mobile platforms. In *2014 IEEE International Symposium on Workload Characterization (IISWC)*, pages 171–180, Oct 2014.

[223] S. Park, J. Park, D. Shin, Y. Wang, Q. Xie, M. Pedram, and N. Chang. Accurate modeling of the delay and energy overhead of dynamic voltage and frequency scaling in modern microprocessors. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 32(5):695–708, May 2013.

[224] A. Patel, F. Afram, S. Chen, and K. Ghose. MARSS: A full system simulator for multicore x86 CPUs. In *2011 48th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1050–1055, June 2011.

[225] A. Pathania, A. E. Irimiea, A. Prakash, and T. Mitra. Power-performance modelling of mobile gaming workloads on heterogeneous mpsocs. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2015.

[226] D. Pelleg and A. W. Moore. X-means: Extending k-means with efficient estimation of the number of clusters. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 727–734, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[227] T. Pering, T. Burd, and R. Brodersen. Dynamic voltage scaling and the design of a low-power microprocessor system. In *Power Driven Microarchitecture Workshop, attached to ISCA98*, pages 96–101, 1998.

[228] N. Peters, S. Park, S. Chakraborty, B. Meurer, H. Payer, and D. Clifford. Web browser workload characterization for power management on hmp platforms. In *2016 International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, pages 1–10, Oct 2016.

[229] K. B. Petersen and M. S. Pedersen. The matrix cookbook, January 2005.

[230] D. T. Pham, S. S. Dimov, and C. D. Nguyen. Selection of k in k-means clustering. *Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science*, 219(1):103–119, 2005.

[231] F. Picard. An introduction to process segmentation. Technical report, March 2007.

[232] F. Picard. A not so short introduction to process segmentation, 2010.

[233] F. Picard, E. Lebarbier, E. Budinskà, and S. Robin. Joint segmentation of multivariate gaussian processes using mixed linear models. *Computational Statistics & Data Analysis*, 55(2):1160 – 1170, 2011.

[234] F. Pittino, F. Beneventi, A. Bartolini, and L. Benini. A scalable framework for online power modelling of high-performance computing nodes in production. In *2018 International Conference on High Performance Computing Simulation (HPCS)*, pages 300–307, July 2018.

[235] J. Pouwelse, K. Langendoen, and H. Sips. Dynamic voltage scaling on a low-power microprocessor. In *Proc. 7th Annu. Int. Conf. on Mobile computing and networking*, pages 251–259. ACM, 2001.

[236] J. Power, J. Hestness, M. S. Orr, M. D. Hill, and D. A. Wood. gem5-gpu: A heterogeneous cpu-gpu simulator. *IEEE Computer Architecture Letters*, 14(1):34–36, Jan 2015.

[237] A. Prakash, S. Wang, A. E. Irimiea, and T. Mitra. Energy-efficient execution of data-parallel applications on heterogeneous mobile platforms. In *2015 33rd IEEE International Conference on Computer Design (ICCD)*, pages 208–215, Oct 2015.

[238] M. Pricopi, T. S. Muthukaruppan, V. Venkataramani, T. Mitra, and S. Vishin. Power-performance modeling on asymmetric multi-cores. In *2013 International Conference on Compilers, Architecture and Synthesis for Embedded Systems (CASES)*, pages 1–10, Sept 2013.

[239] P. Péneau, R. Bouziane, A. Gamatié, E. Rohou, F. Bruguier, G. Sassatelli, L. Torres, and S. Senni. Loop optimization in presence of stt-mram caches: A study of performance-energy tradeoffs. In *2016 26th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 162–169, Sept 2016.

[240] M. S. Rahim and T. Ahmed. An initial centroid selection method based on radial and angular coordinates for k-means algorithm. In *2017 20th International Conference of Computer and Information Technology (ICCIT)*, pages 1–6, Dec 2017.

[241] P. Ramachandrany, S. V. Adve, P. Bose, and J. A. Rivers. Metrics for architecture-level lifetime reliability analysis. In *ISPASS 2008 - IEEE International Symposium on Performance Analysis of Systems and software*, pages 202–212, April 2008.

[242] J. Ren, L. Gao, H. Wang, and Z. Wang. Optimise web browsing on heterogeneous mobile platforms: A machine learning based approach. In *IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pages 1–9, May 2017.

[243] 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.

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

[245] G. Rodríguez. Lecture notes in generalized linear models. `http://data.princeton.edu/wws509/notes/c2s9.html`, 2017. [Online; accessed 28-Aug-2018].

[246] A. Roelke, R. Zhang, K. Mazumdar, K. Wang, K. Skadron, and M. R. Stan. Pre-rtl voltage and power optimization for low-cost, thermally challenged multicore chips. In *2017 IEEE International Conference on Computer Design (ICCD)*, pages 597–600, Nov 2017.

[247] E. Rotem, A. Naveh, A. Ananthakrishnan, E. Weissmann, and D. Rajwan. Power-management architecture of the intel microarchitecture code-named sandy bridge. *IEEE Micro*, 32(2):20–27, March 2012.

[248] K. Roy, S. Mukhopadhyay, and H. Mahmoodi-Meimand. Leakage current mechanisms and leakage reduction techniques in deep-submicrometer cmos circuits. *Proceedings of the IEEE*, 91(2):305–327, Feb 2003.

[249] Sakshi, Dhariwal, Sandeep, and Singh, Amandeep. Analyzing the effect of gate dielectric on the leakage currents. *MATEC Web of Conferences*, 57:01028, 2016.

[250] M. E. Salehi, M. Samadi, M. Najibi, A. Afzali-Kusha, M. Pedram, and S. M. Fakhraie. Dynamic voltage and frequency scheduling for embedded processors considering power/performance tradeoffs. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 19(10):1931–1935, 2011.

[251] A. L. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3(3):210–229, July 1959.

[252] A. Sandberg, N. Nikoleris, T. E. Carlson, E. Hagersten, S. Kaxiras, and D. Black-Schaffer. Full speed ahead: Detailed architectural simulation at near-native speed. In *2015 IEEE International Symposium on Workload Characterization*, pages 183–192, Oct 2015.

[253] S. Sankaran and R. Sridhar. Energy modeling for mobile devices using performance counters. In *2013 IEEE 56th International Midwest Symposium on Circuits and Systems (MWSCAS)*, pages 441–444, Aug 2013.

[254] S. Sarma, T. Muck, L. A. D. Bathen, N. Dutt, and A. Nicolau. Smartbalance: A sensing-driven linux load balancer for energy efficiency of heterogeneous mpsocs. In *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, June 2015.

[255] H. Sayadi, D. Pathak, I. Savidis, and H. Homayoun. Power conversion efficiency-aware mapping of multithreaded applications on heterogeneous architectures: A comprehensive parameter tuning. In *2018 23rd Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 70–75, Jan 2018.

[256] R. R. Schaller. Moore's law: past, present and future. *IEEE Spectrum*, 34(6):52–59, June 1997.

[257] K. F. Schuegraf and C. Hu. Hole injection sio2 breakdown model for very low voltage lifetime extrapolation. *IEEE Transactions on Electron Devices*, 41(5):761–767, May 1994.

[258] J. Setoain, A. Chacon, and F. Spiga. Simulating sve-optimised genomics workloads on gem5. In *2018 IEEE International Conference on Cluster Computing (CLUSTER)*, pages 637–637, Sept 2018.

[259] C. Shalizi. Lecture 13: Simple linear regression in matrix format. `https://www.stat.cmu.edu/~cshalizi/mreg/15/lectures/13/lecture-13.pdf`, October 2016. [Online; accessed 14-August-2018].

[260] J. Sharkey, A. Buyuktosunoglu, and P. Bose. Evaluating design tradeoffs in on-chip power management for cmps. In *Proceedings of the 2007 international symposium on Low power electronics and design (ISLPED '07)*, pages 44–49, Aug 2007.

[261] H. Shi and M. Xu. A data classification method using genetic algorithm and k-means algorithm with optimizing initial cluster center. In *2018 IEEE International Conference on Computer and Communication Engineering Technology (CCET)*, pages 224–228, Aug 2018.

[262] D. Shin, K. Kim, N. Chang, W. Lee, Y. Wang, Q. Xie, and M. Pedram. Online estimation of the remaining energy capacity in mobile systems considering system-wide power consumption and battery characteristics. In *2013 18th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 59–64, Jan 2013.

[263] P. Shivakumar, N. P. Jouppi, and P. Shivakumar. Cacti 3.0: An integrated cache timing, power, and area model. Technical report, 2001.

[264] J. Shuja, A. Gani, A. Naveed, E. Ahmed, and C.-H. Hsu. Case of arm emulation optimization for offloading mechanisms in mobile cloud computing. *Future Generation Computer Systems*, 76:407 – 417, 2017.

[265] A. Shye, B. Scholbrock, and G. Memik. Into the wild: Studying real user activity patterns to guide power optimizations for mobile architectures. In *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 168–178, Dec 2009.

[266] L. Simon, D. Young, and I. Pardoe. Detecting multicollinearity using variance inflation factors. `https://onlinecourses.science.psu.edu/stat501/node/347/`, May 2013. [Online; accessed 17-May-2015].

[267] 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.

[268] G. Singla, G. Kaur, A. K. Unver, and U. Y. Ogras. Predictive dynamic thermal and power management for heterogeneous mobile platforms. In *2015 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 960–965, March 2015.

[269] K. Skadron, M. R. Stan, K. Sankaranarayanan, W. Huang, S. Velusamy, and D. Tarjan. Temperature-aware microarchitecture: Modeling and implementation. *ACM Trans. Archit. Code Optim.*, 1(1):94–125, Mar. 2004.

[270] T. Smejkal, M. Hähnel, T. Ilsche, M. Roitzsch, W. E. Nagel, and H. Härtig. E-team: Practical energy accounting for multi-core systems. In *2017 USENIX Annual Technical Conference (USENIX ATC 17)*, pages 589–601, Santa Clara, CA, 2017. USENIX Association.

[271] A. Software. Workload Automation. `https://github.com/ARM-software`, 2018. [Online; accessed 5-April-2018].

[272] E. D. Sozzo, G. C. Durelli, E. M. G. Trainiti, A. Miele, M. D. Santambrogio, and C. Bolchini. Workload-aware power optimization strategy for asymmetric multiprocessors. In *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 531–534, March 2016.

[273] V. Spiliopoulos, A. Bagdia, A. Hansson, P. Aldworth, and S. Kaxiras. Introducing dvfs-management in a full-system simulator. In *2013 IEEE 21st International Symposium on Modelling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 535–545, Aug 2013.

[274] V. Spiliopoulos, S. Kaxiras, and G. Keramidas. Green governors: A framework for continuously adaptive dvfs. In *2011 International Green Computing Conference and Workshops*, pages 1–8, July 2011.

[275] V. Spiliopoulos, A. Sembrant, and S. Kaxiras. Power-sleuth: A tool for investigating your program's power behavior. In *2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, pages 241–250, Aug 2012.

[276] V. Spiliopoulos, A. Sembrant, G. Keramidas, E. Hagersten, and S. Kaxiras. A unified DVFS-cache resizing framework. Technical Report 2016-014, Department of Information Technology, Uppsala University, Aug 2016.

[277] J. Srinivasan, S. Adve, P. Bose, J. Rivers, C.-K. Hu, P. Emma, B. Linder, and E. Y Wu. Ramp: A model for reliability aware microprocessor design. 29:312–122, 01 2004.

[278] J. Srinivasan, S. V. Adve, P. Bose, and J. A. Rivers. Lifetime reliability: toward an architectural solution. *IEEE Micro*, 25(3):70–80, May 2005.

[279] K. R. Stokke, H. K. Stensland, P. Halvorsen, and C. Griwodz. Why race-to-finish is energy-inefficient for continuous multimedia workloads. In *2015 IEEE 9th International Symposium on Embedded Multicore/Many-core Systems-on-Chip*, pages 57–64, Sept 2015.

[280] B. Su, J. Gu, L. Shen, W. Huang, J. L. Greathouse, and Z. Wang. Ppep: Online performance, power, and energy prediction framework and dvfs space exploration. In *2014 47th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 445–457, Dec 2014.

[281] V. Sundararajan and K. K. Parhi. Low power synthesis of dual threshold voltage cmos vlsi circuits. In *Proceedings. 1999 International Symposium on Low Power Electronics and Design (Cat. No.99TH8477)*, pages 139–144, Aug 1999.

[282] M. Takamiya, Y. K. Ramadass, K. Bowman, G. V. Pique, S. Nagai, and D. Sylvester. F1: Integrated voltage regulators for soc and emerging iot systems. In *2017 IEEE International Solid-State Circuits Conference (ISSCC)*, pages 500–502, Feb 2017.

[283] H. Takase, K. Aono, Y. Matsubara, K. Takagi, and N. Takagi. An evaluation framework of os-level power managements for the big.little architecture. In *2016 14th IEEE International New Circuits and Systems Conference (NEWCAS)*, pages 1–4, June 2016.

[284] I. Takouna, W. Dawoud, and C. Meinel. Accurate mutlicore processor power models for power-aware resource management. In *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, pages 419–426, Dec 2011.

[285] J. Tang, C. Liu, S. Liu, and J.-L. Gaudiot. Practical models for energy-efficient prefetching in mobile embedded systems. *Microprocess. Microsyst.*, 37(8):1173–1182, Nov. 2013.

[286] J. Tang, S. Liu, Z. Gu, C. Liu, and J. Gaudiot. Prefetching in embedded mobile systems can be energy-efficient. *IEEE Computer Architecture Letters*, 10(1):8–11, Jan 2011.

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

[288] The kernel development community. CPU performance scaling. `https://www.kernel.org/doc/html/v4.14/admin-guide/pm/cpufreq.html`, August 2018. [Online; accessed 26-Aug-2018].

[289] G. Theodorou, N. Kranitis, A. Paschalis, and D. Gizopoulos. Power-aware optimization of software-based self-test for l1 caches in microprocessors. In *2014 IEEE 20th International On-Line Testing Symposium (IOLTS)*, pages 154–159, July 2014.

[290] M. Thulasidas. A quality metric for k-means clustering. In *2018 14th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pages 752–757, July 2018.

[291] J. Treibig, G. Hager, and G. Wellein. Likwid: A lightweight performance-oriented tool suite for x86 multicore environments. In *Proceedings of the 2010 39th International Conference on Parallel Processing Workshops*, ICPPW '10, pages 207–216, Washington, DC, USA, 2010. IEEE Computer Society.

[292] C. Truong, L. Oudre, and N. Vayatis. A review of change point detection methods. *CoRR*, abs/1801.00718, 2018.

[293] I. Ukhov, M. Bao, P. Eles, and Z. Peng. Steady-state dynamic temperature analysis and reliability optimization for embedded multiprocessor systems. In *DAC Design Automation Conference 2012*, pages 197–204, June 2012.

[294] I. Ukhov, P. Eles, and Z. Peng. Temperature-centric reliability analysis and optimization of electronic systems under process variation. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 23(11):2417–2430, Nov 2015.

[295] University of Vienna. Machine learning: Dimensionality reduction. `http://vda.univie.ac.at/Teaching/ML/14s/LectureNotes/02_Dimensionality_Reduction.pdf`, 2014. [Online; accessed 22-May-2019].

[296] K. Usami and M. Horowitz. Clustered voltage scaling technique for low-power design. In *Proceedings of the 1995 International Symposium on Low Power Design*, ISLPED '95, pages 3–8, New York, NY, USA, 1995. ACM.

[297] S. Valle, W. Li, and S. J. Qin. Selection of the number of principal components: The variance of the reconstruction error criterion with a comparison to other methods. *Industrial and Engineering Chemistry Research*, 38(11):4389–4401, 1999.

[298] A. Vassighi and M. Sachdev. Thermal runaway in integrated circuits. *IEEE Transactions on Device and Materials Reliability*, 6(2):300–305, June 2006.

[299] H. J. M. Veendrick. Short-circuit dissipation of static cmos circuitry and its impact on the design of buffer circuits. *IEEE Journal of Solid-State Circuits*, 19(4):468–473, Aug 1984.

[300] V. Vijay, V. P. Raghunath, A. Singh, and S. N. Omkar. Variance based moving k-means algorithm. In *2017 IEEE 7th International Advance Computing Conference (IACC)*, pages 841–847, Jan 2017.

[301] B. Vikas and B. Talawar. On the cache behavior of splash-2 benchmarks on arm and alpha processors in gem5 full system simulator. In *2014 3rd International Conference on Eco-friendly Computing and Communication Systems*, pages 5–8, Dec 2014.

[302] T. Vogelsang. Understanding the energy consumption of dynamic random access memories. In *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*, pages 363–374, Dec 2010.

[303] M. I. Wahyuddin, P. S. Priambodo, and H. Sudibyo. State of charge (soc) analysis and modeling battery discharging parameters. In *2018 4th International Conference on Science and Technology (ICST)*, pages 1–5, Aug 2018.

[304] M. J. Walker, S. Bischoff, S. Diestelhorst, G. V. Merrett, and B. M. Al-Hashimi. Hardware-validated CPU performance and energy modelling. In *2018 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*, pages 44–53, April 2018.

[305] M. J. Walker, A. K. Das, G. V. Merrett, and B. Hashimi. Run-time power estimation for mobile and embedded asymmetric multi-core cpus. In *HIPEAC Workshop on Energy Efficiency with Heterogenous Computing*, January 2015.

[306] M. J. Walker, S. Diestelhorst, S. Bischoff, G. Merrett, and B. Al-Hashimi. Gemstone. `http://gemstone.ecs.soton.ac.uk/`, 2017. [Online; accessed 30-Jan-2018].

[307] M. J. Walker, S. Diestelhorst, A. Hansson, B. M. Al-Hashimi, and G. V. Merrett. Powmon: Run-time CPU power modelling. `http://www.powmon.ecs.soton.ac.uk/powermodeling`, October 2017. [Online; accessed 06-Nov-2017].

[308] M. J. Walker, S. Diestelhorst, A. Hansson, D. Balsamo, G. V. Merrett, and B. M. Al-Hashimi. Thermally-aware composite run-time CPU power models. In *2016 26th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, pages 17–24, Sept 2016.

[309] M. J. Walker, S. Diestelhorst, A. Hansson, A. K. Das, S. Yang, B. M. Al-Hashimi, and G. V. Merrett. Accurate and stable run-time power modeling for mobile and embedded cpus. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 36(1):106–119, Jan 2017.

[310] M. J. Walker, S. Diestelhorst, G. V. Merrett, and B. Al-Hashimi. Accurate and stable empirical CPU power modelling for multi- and many-core systems. In *Adaptive Many-Core Architectures and Systems Workshop (15/06/18)*, June 2018.

[311] S. Wang, H. Chen, and W. Shi. Span: A software power analyzer for multicore computer systems. *Sustainable Computing: Informatics and Systems*, 1(1):23 – 34, 2011.

[312] X. Wang, Y. Jiao, and S. Fei. Estimation of clusters number and initial centers of k-means algorithm using watershed method. In *2015 14th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)*, pages 505–508, Aug 2015.

[313] Wind River. Simics. `https://www.windriver.com/products/simics/`. [Online; accessed 16-Nov-2017].

[314] M. Witkowski, A. Oleksiak, T. Piontek, and J. Weglarz. Practical power consumption estimation for real life hpc applications. *Future Generation Computer Systems*, 29(1):208 – 217, 2013. Including Special section: AIRCC-NetCoM 2009 and Special section: Clouds and Service-Oriented Architectures.

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

[316] E. P. Xing, A. Y. Ng, M. I. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. In *Proceedings of the 15th International Conference on Neural Information Processing Systems*, NIPS'02, pages 521–528, Cambridge, MA, USA, 2002. MIT Press.

[317] H. Yang. Visual assessment of residual plots in multiple linear regression: A model-based simulation perspective. *Multiple Linear Regression Viewpoints*, 2012.

[318] J. Yang and S. Cao. An accurate power and temperature simulation framework for network-on-chip. In *2016 International Conference on Integrated Circuits and Microsystems (ICICM)*, pages 166–171, Nov 2016.

[319] S. Yang, M. D. Powell, B. Falsafi, K. Roy, and T. N. Vijaykumar. An integrated circuit/architecture approach to reducing leakage in deep-submicron high-performance i-caches. In *Proceedings HPCA Seventh International Symposium on High-Performance Computer Architecture*, pages 147–157, Jan 2001.

[320] G. Yao, H. Yun, Z. P. Wu, R. Pellizzoni, M. Caccamo, and L. Sha. Schedulability analysis for memory bandwidth regulated multicore real-time systems. *IEEE Transactions on Computers*, 65(2):601–614, Feb 2016.

[321] H. Yasuura, T. Ishihara, and M. Muroyama. *Energy Management Techniques for SOC Design*, pages 177–223. Springer Netherlands, Dordrecht, 2006.

[322] C. Yoon, S. Lee, Y. Choi, R. Ha, and H. Cha. Accurate power modeling of modern mobile application processors. *Journal of Systems Architecture*, 81(Supplement C):17 – 31, 2017.

[323] M. T. Yourst. Ptlsim: A cycle accurate full system x86-64 microarchitectural simulator. In *2007 IEEE International Symposium on Performance Analysis of Systems Software*, pages 23–34, April 2007.

[324] K. Yu, D. Han, C. Youn, S. Hwang, and J. Lee. Power-aware task scheduling for big.LITTLE mobile processor. In *2013 International SoC Design Conference (ISOCC)*, pages 208–212, Nov 2013.

[325] L. Yu and C. Zhou. Determining the best clustering number of k-means based on bootstrap sampling. In *2018 2nd International Conference on Data Science and Business Analytics (ICDSBA)*, pages 78–83, Sep. 2018.

[326] V. Zaccaria, M. Sami, D. Sciuto, and C. Silvano. *Power Estimation and Optimization Methodologies for VLIW-Based Embedded Systems*. Springer, Boston, MA, 2004.

[327] R. Zamani and A. Afsahi. A study of hardware performance monitoring counter selection in power modeling of computing systems. In *2012 International Green Computing Conference (IGCC)*, pages 1–10, June 2012.

[328] H. Zhang and X. Zhou. A novel clustering algorithm combining niche genetic algorithm with canopy and k-means. In *2018 International Conference on Artificial Intelligence and Big Data (ICAIBD)*, pages 26–32, May 2018.

[329] J. Zhang, D. Li, S. Fan, Z. Guo, W. Hu, and L. Geng. Theoretical model of endp to achieve energy-efficient sram. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 64(8):2049–2062, Aug 2017.

[330] L. Zhang. Linear regression in matrix form. `http://www.stat.purdue.edu/~lingsong/teaching/2018spring/topic3.pdf`, 2018. [Online; accessed 14-August-2018].

[331] M. Zhang and K. Asanovic. Fine-grain cam-tag cache resizing using miss tags. In *Proceedings of the International Symposium on Low Power Electronics and Design*, pages 130–135, Aug 2002.

[332] X. Zhang, J. Lu, and X. Qin. BFEPM: Best fit energy prediction modeling based on CPU utilization. In *2013 IEEE Eighth International Conference on Networking, Architecture and Storage*, pages 41–49, July 2013.

[333] Y. Zhang, Y. Liu, X. Liu, and Q. Li. Enabling accurate and efficient modeling-based CPU power estimation for smartphones. In *2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pages 1–10, June 2017.

[334] Y. Zhang, Y. Liu, L. Zhuang, X. Liu, F. Zhao, and Q. Li. Accurate CPU power modeling for multicore smartphones. Technical report, February 2015.

[335] Y. Zhu and V. J. Reddi. High-performance and energy-efficient mobile web browsing on big/little systems. In *2013 IEEE 19th International Symposium on High Performance Computer Architecture (HPCA)*, pages 13–24, Feb 2013.