WASHINGTON UNIVERSITY IN ST. LOUIS

McKelvey School of Engineering
Department of Computer Science and Engineering

Dissertation Examination Committee:
Jeremy Buhler, Chair
Roger Chamberlain
Christopher Gill
Christine O'Brien
Ning Zhang

Understanding Performance Modeling in a Heterogeneous World

by
Steven Harris

A dissertation presented to
the McKelvey School of Engineering
of Washington University in
partial fulfillment of the
requirements for the degree
of Doctor of Philosophy

December 2026
St. Louis, Missouri

© 2026, Steven Harris

Table of Contents

List of Figures . . . . . **viii**

List of Tables . . . . . **xi**

Acknowledgments . . . . . **xiv**

Abstract . . . . . **xvi**

Chapter 1: Introduction **1**
1.1 Contributions 3
1.2 Outline 5

Chapter 2: Background and Related Work **7**
2.1 Performance Modeling 7
2.1.1 Performance Classification 7
2.1.2 Performance Abstraction 9
2.1.3 Performance Representation 11
2.1.4 Performance Convergence 12
2.2 An Introduction to Structural Equation Modeling 14
2.3 Boltzmann Uniqueness: Recontextualizing Performance Dependencies 20
2.4 Real-Time Maternal Hemorrhage Monitoring 21
2.5 Experimental Platforms 24
2.5.1 Workload 26
2.5.2 Platform Attributes 28

Chapter 3: The Selection, Assessment, Classification Methodology **29**
3.1 The Selection, Assessment, Classification Framework 30
3.1.1 Stage 1: Selection (S_h) 30
3.1.2 Stage 2: Assessment (A_h) 31
3.1.3 Stage 3: Classification (C_h) 32
3.1.4 Reverse Mapping and Preimage Optimization 33
3.1.5 Architectural Consistency and Reduced Latent Variance 34
3.1.6 Industry Application: Predictability vs. Complexity 34
3.2 Homogeneous Performance Insights 36

3.2.1 Stability of the Parameter Vector (β) . . . 37
3.2.2 Elimination of the Mediation Gap . . . 38
3.2.3 Industrial Application: Deterministic Scheduling and DVFS . . . 38
3.2.4 Efficient Inverse Mapping . . . 39
3.2.5 Technical Insight . . . 39
3.3 Heterogeneous Performance Insights . . . 40
3.3.1 Performance Model Portability . . . 40
3.3.2 Configuration Scenarios . . . 40
3.3.3 Problem Formulation: The Brute-Force Trap . . . 42
3.4 Experimental Setup . . . 43
3.4.1 Feature Vector Construction and Dimensionality . . . 44
3.4.2 Forward Predictive Mapping Pipeline . . . 44
3.4.3 Inverse Diagnostic Mapping Pipeline . . . 45
3.4.4 Validation Metrics and Diagnostic Protocol . . . 45
3.5 Experimental Results . . . 46
3.5.1 Analysis of Scalar Mapping (The Baseline) . . . 47
3.5.2 Dimensional Feature Expansion . . . 48
3.5.3 Key Observations . . . 51
3.5.4 Motivation for Structural Equation Modeling (SEM) . . . 52
3.6 Cross-Disciplinary Promise and Future Domains . . . 53
3.7 Chapter Conclusion . . . 54

Chapter 4: Performance Models for Non-Uniform Heterogeneous Platforms and their Validation Using Structural Equation Modeling . . . 57
4.1 Introduction . . . 57
4.1.1 Chapter Overview . . . 59
4.1.2 Problem Statement & Objectives . . . 60
4.1.3 Chapter Research Questions . . . 60
4.2 Research Hypotheses . . . 61
4.3 Methodology . . . 62
4.3.1 Performance Model . . . 62
4.4 Experiments . . . 66
4.4.1 Parameters: Island + Matrix Dimension . . . 66
4.4.2 Parameters: Island + Matrix Dimension + Cores . . . 67
4.4.3 Performance Normalization . . . 69
4.4.4 Parameter Characterization . . . 71
4.4.5 System Parameter Correlation . . . 73
4.4.6 Re-parametrization . . . 76
4.4.7 Can we validate the behavior? . . . 76
4.5 Structural Equation Modeling . . . 78
4.5.1 SEM Measurement Model . . . 79
4.5.2 SEM Structural Model . . . 81
4.5.3 Performance Simulation for SEM Structural Model Confirmation . . . 86

4.6 Discussion . . . 90
4.6.1 First Principles Model . . . 90
4.6.2 Structural Equation Modeling . . . 93
4.6.3 Insights . . . 94
4.7 Chapter Conclusions . . . 95
4.7.1 Evolution: From Foundational Vectors into the Tripartite SAC Methodology . . . 96

Chapter 5: Understanding Asymmetric Multicores Using Structural Equation Modeling . . . 98
5.1 Introduction . . . 98
5.1.1 Chapter Overview . . . 100
5.1.2 Problem Statement & Objectives . . . 100
5.1.3 Chapter Research Questions . . . 101
5.2 SEM Measured Variable Path Analysis Models . . . 101
5.2.1 Parameter Impacts . . . 106
5.2.2 Extension to Radxa Cubie A7A . . . 106
5.3 Performance Modeling as SEM Latent Variables . . . 108
5.3.1 Methodological Synthesis and Model Selection . . . 110
5.4 Combining SEM with First Principles Modeling . . . 114
5.4.1 Polynomial Performance Modeling & Analysis . . . 115
5.4.2 Multi-Parameter Interaction Modeling . . . 120
5.4.3 Performance Articulation and Results Synthesis . . . 124
5.5 New Performance Indicators . . . 126
5.5.1 Complexity Sensitivity . . . 126
5.5.2 Inter-Island Scaling Friction . . . 126
5.5.3 Multi-Core Saturation . . . 127
5.5.4 Stochastic Coupling Significance . . . 127
5.5.5 Critical Complexity Index . . . 127
5.6 Chapter Conclusions . . . 128
5.6.1 Formative vs. Reflective SEM Specification . . . 129
5.6.2 Structural Bottlenecks, Inter-Island Friction, and the Critical Complexity Index . . . 130
5.6.3 Evolution: From The Tripartite SAC Operator Chain into Entanglement . . . 131

Chapter 6: Structural Equation Modeling for Compute Diversity Articulation . . . 133
6.1 Asymmetry and Platform Variance Mitigation . . . 134
6.1.1 Simultaneous Estimation and Full-Information Maximum Likelihood . . . 134
6.1.2 Modeling the Covariance Structure of Indicators . . . 135
6.1.3 Resolving the Platform-Specific OS Scheduling Bias . . . 135
6.1.4 Analysis Insights . . . 136

6.2 Microarchitectural Asymmetric Characterization . . . 139
6.2.1 Theoretical Overview of the Latent Space . . . 139
6.2.2 The Measurement Model (Inner Model) . . . 140
6.2.3 The Structural Model (Outer Model) . . . 142
6.2.4 The Covariance Structure Equation . . . 144
6.2.5 Mitigation of the Errors-in-Variables (EIV) Bias . . . 145
6.3 Empirical Validation . . . 147

Chapter 7: Measure-Theoretic Diagnostics of Architectural Entanglement in Asymmetric Multiprocessing Systems: A Boltzmann Uniqueness Approach . . . 149
7.1 Introduction . . . 149
7.1.1 Chapter Overview . . . 151
7.1.2 Problem Statement and Objective . . . 152
7.1.3 Chapter Research Questions . . . 153
7.2 Measure-Theoretic Framework . . . 154
7.2.1 Architectural Mapping . . . 154
7.2.2 Matrix-Based Relational Model . . . 155
7.2.3 Ghost Power Coefficient . . . 156
7.3 Experimental Methodology . . . 157
7.3.1 Workload Construction . . . 157
7.3.2 Platform Analysis . . . 158
7.4 Empirical Results . . . 160
7.4.1 Cluster Independence . . . 161
7.4.2 Measurement States . . . 163
7.5 Discussion: Addressing the Physics of Entanglement on ARM Platforms . . . 164
7.5.1 Architectural Entanglement . . . 165
7.5.2 Power-Domain Entanglement . . . 166
7.5.3 Empirical Inter-Cluster Entanglement . . . 168
7.6 Extending the Investigation to the Intel i7-12800H . . . 171
7.6.1 Introduction and Platform Overview . . . 171
7.6.2 Power Management Envelopes: PL1 and PL2 . . . 172
7.6.3 Hardware and Software Specifications . . . 173
7.6.4 Progression of Experiments . . . 174
7.6.5 Comparative Analysis: Intel x86 vs. ARM DynamIQ . . . 175
7.6.6 Novelty and Modeling Importance . . . 175
7.7 Intel i7-12800H Architectural Entanglement . . . 176
7.7.1 Technical Analysis and Contextual Evaluation . . . 178
7.8 Discussion: Intel vs. ARM Platforms . . . 179
7.8.1 The 90-Second Result: Why It Showed Entanglement . . . 180
7.8.2 Pulse-Width Modulation (PWM) and Peak Interaction . . . 180
7.8.3 Shift from Electrical to Algorithmic Coupling . . . 181
7.8.4 Ghost Power on the Intel i7-12800H . . . 182

7.9 Chapter Conclusions . . . 182
7.9.1 Evolution: From Entanglement to Biomedical Engineering Applications 184

Chapter 8: An Entropy-Bounded Thermodynamic Approach for Real-Time Maternal Hemorrhage Monitoring . . . 185
8.1 Introduction . . . 185
8.1.1 Chapter Overview . . . 186
8.1.2 Problem Statement & Objectives . . . 187
8.1.3 Chapter Research Questions . . . 188
8.2 Clinical Mechanics of LSCI and Peripheral Vasoconstriction . . . 189
8.2.1 Environmental Resources and Portability Boundaries . . . 189
8.2.2 The Throughput-Latency Paradigm Conflict . . . 190
8.3 Research Hypotheses . . . 190
8.4 Platform and Experiments . . . 191
8.4.1 Real-Time Embedded System Benchmarks . . . 192
8.4.2 Raspberry Pi Platform . . . 193
8.4.3 System Performance Experiments . . . 193
8.4.4 Storage & I/O Assessment Experiments . . . 194
8.4.5 Hardware Telemetry Experiments . . . 195
8.4.6 Performance Bottlenecks . . . 195
8.5 Paradigm Conflict in Embedded Performance Modeling . . . 196
8.5.1 Technical Challenges (TMP) . . . 198
8.5.2 Clinical Challenges (LMP) . . . 198
8.6 Performance Modeling . . . 199
8.6.1 Cross-Disciplinary Infusion: Boltzmann Uniqueness Theorem . . . 199
8.6.2 Entropy-Bounded Performance Model . . . 201
8.6.3 Validation Against Platform Data . . . 203
8.7 Interdisciplinary Convergence . . . 205
8.7.1 Computer Science Contributions . . . 205
8.7.2 Computer Engineering Contributions . . . 206
8.7.3 Biomedical Engineering Contributions . . . 206
8.8 Chapter Conclusions . . . 206

Chapter 9: Conclusions and Future Work . . . 210
9.1 Conclusions . . . 210
9.2 Future Work . . . 211
9.2.1 Future Research Roadmap . . . 213

References . . . 215

Appendix A: Boltzman Theorem and Distribution . . . 228
A.1 Boltzmann Uniqueness Theorem . . . 228
A.2 Boltzmann Distribution . . . 229

A.3 Boltzman AMP Applicability 230

Appendix B: Workload Construction 234

Appendix C: Execution and Telemetry 236

Appendix D: Platform Deep Dive 237

List of Figures

Figure 2.1: SEM Model Structure 15

Figure 2.2: SEM Path Elements 17

Figure 2.3: Orange Pi 5 Core Configuration 24

Figure 2.4: Cubie A7A Core Configuration 24

Figure 2.5: Intel i7-12800H Core Configuration 24

Figure 3.1: Potential Scenarios 40

Figure 3.2: Scenario Connections 41

Figure 3.3: Selection, Assessment, Classification Stages 55

Figure 3.4: Applied Selection, Assessment, Classification in Action 56

Figure 4.1: Orange Pi 5 Matrix Multiplication by Island 67

Figure 4.2: Orange Pi 5 Matrix Multiplication by Configuration 68

Figure 4.3: Static Big-to-Little Ratio 70

Figure 4.4: Normalized Execution Time Ratio 72

Figure 4.5: Correlations – 768 × 768 Matrices 74

Figure 4.6: Correlations – 1216 × 1216 Matrices 75

Figure 4.7: Re-parameterized Big-to-Little Ratio 77

Figure 4.8: Complete Model 77

Figure 4.9: Initial Full SEM Model 83

Figure 4.10: Specified SEM Structural Model 84
Figure 4.11: Bootstrapped Specified Model 85
Figure 4.12: IPC Distribution 91
Figure 4.13: L1 DCache Loads Distribution 91
Figure 4.14: LLC Loads Distribution 92
Figure 4.15: Parallel Time Distribution 92

Figure 5.1: SEM Path Analysis Model 1 103
Figure 5.2: SEM Path Analysis Model 4 103
Figure 5.3: Parameter Impacts for Orange Pi 5 107
Figure 5.4: Parameter Impacts for Cubie A7A 109
Figure 5.5: Orange Pi SEM Reflective Model 111
Figure 5.6: Cubie SEM Reflective Model 111
Figure 5.7: Orange Pi SEM Reflective Model 111
Figure 5.8: Cubie SEM Reflective Model 111
Figure 5.9: Orange Pi SEM Formative Model 112
Figure 5.10: Cubie SEM Formative Model 112
Figure 5.11: Orange Pi SEM Formative Model 113
Figure 5.12: Cubie SEM Formative Model 113
Figure 5.13: Orange Pi Model Visualization 116
Figure 5.14: Cubie Model Visualization 116

Figure 6.1: Theoretical SEM Model 134
Figure 6.2: 4-Parameter Architectural Intensity SEM 136
Figure 6.3: 9-Parameter Architectural Intensity SEM 137

Figure 7.1: Measure-theoretic Diagnostic Flow 151

Figure 8.1: Laser Speckle Contrast Imaging 191

Figure 8.2: Multi-Stage High-Frequency Recording Architecture for LSCI Capture . 194

Figure 8.3: Entropy vs. Frame Rate 202

Figure 8.4: Jitter vs. CPU Utilization 204

Figure 9.1: The Post-Von Neumann Co-Design Landscape. 213

Figure 9.2: Structural Equation-Variational Autoencoder (SE-VAE) Architecture. . 214

List of Tables

Table 2.1: ARM big.LITTLE Specifications 26

Table 2.2: Platform Attributes 28

Table 3.1: Scenarios by Configuration 43

Table 3.2: 1-Parameter Little Core Mapping 47

Table 3.3: 1-Parameter Big Core Mapping 47

Table 3.4: 4-Parameter Big Core Mapping 49

Table 3.5: 9-Parameter Big Core Mapping 49

Table 3.6: 9-Parameter Inverse Big Core Mapping 50

Table 4.1: Notation and Description 65

Table 4.2: Big-to-Little Normalized Ratios 69

Table 4.3: Big-to-Little Normalized Ratios 78

Table 4.4: Big-to-Little Normalized Ratio Tests 78

Table 4.5: Anticipated Metric Shift 87

Table 4.6: Model Parameters and Investigated Metric Shift 88

Table 4.7: Bootstrapped Memory-Bound Model Regressions and Variances 90

Table 5.1: Orange Pi 5 SEM Path Analysis Across Successive Models 105

Table 5.2: Comparative Model Regressions and Variances 108

Table 5.3: Total Impacts . . . 109
Table 5.4: Latent Performance Variable R^2 Values Across SEM Models . . . 110
Table 5.5: Tested Polynomial Regression Models . . . 118
Table 5.6: Model Fit Information Criteria Comparison . . . 118
Table 5.7: Comparative Regression and Variance Results . . . 119
Table 5.8: Notation and Description . . . 121
Table 5.9: Comparative Regression Models . . . 123
Table 5.10: Architectural Metrics Derived from SEM Estimates . . . 126
Table 5.11: Methodological Evolution . . . 129

Table 6.1: Stage 1 Individual Predictive Performance (4-Parameter Configuration) . 145
Table 6.2: Stage 1 Individual Predictive Performance (9-Parameter Configuration) . 145

Table 7.1: Summary of Correlation Contrast . . . 161
Table 7.2: Rockchip RK3588 (Orange Pi 5) Power Island Analysis . . . 161
Table 7.3: Allwinner A733 (Cubie A7A) Power Island Analysis. . . . 161
Table 7.4: Ghost Power Coefficient and Architectural Entanglement Metrics . . . 162
Table 7.5: Orange Pi 5 Entanglement Metrics . . . 168
Table 7.6: Model Paradigm Shift . . . 171
Table 7.7: Hardware Architectural Specifications . . . 173
Table 7.8: Software and Telemetry Environment . . . 173
Table 7.9: Systemic Power and Statistical Significance . . . 177
Table 7.10: Pulse-width Modulation Diagnostic Results . . . 178
Table 7.11: Comparative Ghost Power Mechanisms . . . 182

Table 8.1: Hardware Comparison for LSCI Performance Modeling 192
Table 8.2: Metrics Tested in LSCI Platform Experiments 197
Table 8.3: Platform Performance Metrics vs. Targets 197
Table 8.4: Mapping Platform Data to Boltzmann Variables 201
Table 8.5: Input Parameters for Multi-Core Resource Allocation 203
Table 8.6: Optimal vs. Current Distribution 203
Table 8.7: Mean Measurement Values for 80 FPS Goal 205
Table 8.8: Interdisciplinary Convergence Points for PPH Detection 207

Table A.1: Measure-Theoretic Diagnostic of Architectural Entanglement in AMPs. . 231

Acknowledgments

I would like to thank the following for financial support of my work: the National Science Foundation through grants CNS-1763503, CNS-1814739, and CNS-2229290; a gift from BECS Technology, Inc.; and a grant through the Big Ideas Program of the Washington University Institute for Informatics, Data Science and Biostatistics. I would like to thank the Washington University Department of Biomedical Engineering, Department of Mathematics and Statistics, and Department of Computer Science and Engineering, as well as the Department of Statistical Genomics at the Washington University School of Medicine. These departments have fostered, sustained, and nurtured my research, providing the occasion to advance knowledge and make cross-disciplinary contributions throughout this dissertation. The scholars within these departments could have chosen any number of talented students to work in their labs, but they chose me. For that, my life is forever changed, and I am eternally grateful for the opportunities they have so richly afforded me. I want to thank my primary mentors, Dr. Roger Chamberlain and Dr. Christopher Gill, who served as the catalyst behind my research. Their guidance provided the rich, fertile soil in which my ideas grew to maturity. I am humbled by Dr. Christine O'Brien and Francesca Bonnetta-Misteli's generosity, who gave me the privilege of joining them in the development of life-saving maternal technologies, along with Lleyton Martin, who supported my efforts. Dr. Mladen Wickerhauser is solely responsible for my ability to comprehend and apply complex mathematical techniques. Dr. Ning Zhang and Dr. Jeremy Buhler provided invaluable insights that sharpened and clarified not only my work, but my investigative processes. I thank Dr. Michael Province, without whom I could never have supported my family as a single father while pursuing doctoral research. Finally, I thank my first teacher, my mother Maria De Shields, who sacrificed many years of her life along with countless opportunities to home-school me. Words cannot do justice to the kindness you, beloved mother, have lavished upon me. To you and the other esteemed scholars, thank you.

Steven Harris

Washington University in St. Louis
December 2026

To My Companions

Having run headlong through the dense jungle of my master's degree, my thesis marked an arrival at a clearing. On that mountain ridge I stood—anchored by my mother and first teacher, Maria De Shields, my son, Yochannan, and my sister, Salina Greene, my sole companions. I found myself gazing down at the steep descent toward doctoral studies. Then Salina, the steady hand who brought me to WashU, passed into the quiet of the ancestors. Words failed her loss. She was an irreplaceable compass. In the stillness that followed, the vision for my PhD vanished. Not knowing where to turn, I chose the most radical direction: inward.

I held the memory of the legendary swordsmith Amakuni and his son Amakura, who sealed themselves inside their forge to pray seven days and seven nights to the Shinto deities, emerging after thirty-one days with a blade of immortal craftsmanship. I thought of Srinivasa Ramanujan, the mathematician who received whole formulas in his sleep through the grace of the Goddess Namagiri. I recalled the proverb of King Solomon—that She, Wisdom, cries out in the public square, waiting only for us to pause and listen. Recognized by endless names and manifested in infinite forms, the limitless Divine waits patiently for our stillness. When I yielded inward, Wisdom answered.

The cornerstone of my doctoral work was laid when I uncovered a mathematical framework to support high-performance detection of postpartum hemorrhage in mothers. That single class of mathematics became the bedrock of my research, my publications, and the dissertation before you. Every inquiry in this manuscript traces its lineage back to that initial mandate: to protect life at its origin. Wisdom is anciently rendered as feminine, and every solution forged in these pages was inspired by that first impulse to save mothers—the primary sustainer of our species. Beyond the mathematical insights, algorithms, and technical applications recorded here, I offer this work to you—my dear companions—as a call. May it serve as an invitation to quiet the noise, turn within, and commune with that infinite Presence waiting to reveal your own path.

ABSTRACT OF THE DISSERTATION

Understanding Performance Modeling in a Heterogeneous World

by

Steven Harris

Doctor of Philosophy in Computer Engineering

Washington University in St. Louis, 2026

Professor Jeremy Buhler, Chair

Modern edge platforms have fundamentally shifted from uniform processor designs to asymmetric multicore architectures, combining high-performance and energy-efficient cores on a single chip. In these heterogeneous environments, standard performance models fall short as execution behavior is dictated more by nuanced core types, memory hierarchies, and workload interactions rather than core counts alone. By utilizing structural equation modeling alongside first-principles analysis on ARM big.LITTLE platforms, we show that workload-to-hardware relationships follow a structured, predictable pattern where high-performance cores can act as scaled equivalents of efficiency cores.

Moving beyond static core counts, path analysis across diverse hardware topologies reveals that runtime dynamics, including thread migrations and cache misses, dictate execution behavior, uncovering interaction thresholds where efficiency cores saturate shared memory pathways and degrade overall throughput. Challenging the legacy assumption that core clusters operate independently, we apply the Boltzmann uniqueness theorem to demonstrate architectural entanglement across both ARM and Intel hybrid architectures. This physical coupling, where shared caches and power delivery networks cause activity in one cluster to

induce parasitic power floor elevation or starve the performance of adjacent idle cores, refutes additive energy models and motivates relational, power-rail-aware scheduling.

Finally, applying these microarchitectural realities to a real-time clinical wearable device for early maternal postpartum hemorrhage monitoring demonstrates that while low-cost hardware can achieve high imaging frame rates, best-effort software scheduling causes catastrophic latency spikes and processing delays under heavy CPU loads. By modeling processing delay as a fluid via the Boltzmann transport equation, deterministic task distribution eliminates heuristic scheduling randomness to enforce strict clinical latency bounds within mandatory clinical safety limits to protect patient health. As systems engineering confronts the nonlinear realities of heterogeneous architectures, these mathematically grounded, hardware-in-the-loop methodologies provide a novel navigational framework for mastering next-generation silicon.

Chapter 1

Introduction

The rapid transition of modern edge-cloud and embedded computing platforms from homogeneous microarchitectures to non-uniform, asymmetric multiprocessing (AMP) architectures represents a fundamental paradigm shift in computing. For decades, software developers operated within an "isotropic grove" of symmetric silicon, relying on the single-core induction approach where resource scaling behaves linearly (most of the time) and execution units act as passive, independent resources. However, on modern heterogeneous substrates—such as ARM's big.LITTLE and Intel's hybrid architectures—this assumption of orthogonal execution falls apart. Asymmetric core hierarchies, non-uniform memory hierarchies, and shared on-chip interconnects introduce severe non-linear bottlenecks, thermal cross-talk, and memory bus contention that render classical symmetric-core performance models and scalar additive heuristics fundamentally underspecified.

This dissertation addresses this predictive crisis by pioneering a hardware-aware, relational modeling paradigm that unifies statistical mechanics, silicon diagnostics, and multivariate statistical analysis. Utilizing tools that have been used extensively in other disciplines (e.g., social science, business, etc.), we apply these tools to the performance analysis of AMP architectures. A defining characteristic of these tools is that they are primarily data driven, relying on empirical measurements to draw conclusions about the underlying system being modeled. This is in contrast to many performance models that are primarily guided by first-principles understanding of the engineered system being modeled. When we succeed, we combine the benefits of both modeling approaches.

First, we formulate the **Selection, Assessment, Classification (SAC) methodology**, a tripartite operator chain ($G = C_h \circ A_h \circ S_h$) that maps high-level workloads to quantitative metrics and qualitative regimes across distinct memory and scheduling boundaries. To bypass the "compound error explosion" of standard Ordinary Least Squares (OLS) regressions, we introduce the application of **Structural Equation Modeling (SEM)** [78] to edge

systems engineering. By treating hardware performance counters as reflective or formative indicators of unobserved latent variables—such as *Core Performance* and *Cache Efficiency*—we isolate low-level microarchitectural bottlenecks directly from runtime telemetry. Under this framework, we derive the **Critical Complexity Index (χ)**, a new architectural metric that identifies the structural thresholds where inter-island coherency traffic negates parallel scaling, signaling when task schedulers should prioritize core isolation over active parallelism to prevent memory bus poisoning.

Second, we formulate a disproof of the "sum of parts" power assumption by deploying the **Boltzmann Uniqueness Theorem** [13] as a silicon diagnostic framework. By treating cluster power profiles as continuous probability distributions of energy states, we establish that a system is truly uncoupled if and only if its joint resource measurement is strictly the mathematical convolution of its isolated subsystem measures. Across three distinct topologies—the balanced 4+4 ARM-based Orange Pi 5, the asymmetric 2+6 ARM Radxa Cubie A7A, and the high-performance x86-based Intel i7-12800H—we document a systemic failure of convolution commutativity. To quantify this physical-electrical coupling, we introduce a **Ghost Power Coefficient (γ)** and propose a matrix-based relational power model. We isolate and model the physical mechanisms of this coupling, including **"Fabric Drag"** ($\gamma \approx 0.5$) in the DynamIQ Shared Unit, **"Monolithic Voltage Locking"** ($\gamma \approx 0.8$) on cost-optimized shared power rails, and **"Inverse LLC Coupling"** ($\gamma \approx -0.08$) mediated by cache line starvation and power delivery network transients.

Finally, we demonstrate the predictive fidelity and real-world applicability of our relational models across a pair of high-impact edge workloads:

1. Deep Learning and Transformer Primitives (GEMM): We trace the non-linear "congestion collapse" that occurs when highly coupled asymmetric core clusters compete for limited Last-Level Cache residency under speculative, out-of-order execution.

2. Real-Time Clinical Engineering (LSCI): To address early-stage maternal post-partum hemorrhage monitoring, we implement a Laser Speckle Contrast Imaging pipeline on a low-cost Raspberry Pi 5. We show how standard scheduling triggers "entropy collapse" events at 85% CPU utilization, resulting in catastrophic 280 ms latency spikes. Utilizing our **Entropy-Bounded Performance Model (EBPM)**, we prove that frame rates are bounded by the Shannon entropy of processing delays

($H(S) \leq 2.64$ bits), and deploy a Fermi-Dirac-like resource allocation function that stabilizes latency jitter below the critical ± 2 ms threshold.

Ultimately, this dissertation establishes that logical software intent remains subservient to the thermodynamic, electrical, and bus-level realities of the silicon substrate. By mapping the physical entanglements of asymmetric hardware, this work proposes mathematical baselines and "power-rail-aware" approaches to scheduling policy required to master the complex, non-uniform co-design landscapes of next-generation edge intelligence.

1.1 Contributions

This dissertation advances heterogeneous platform execution modeling, microarchitectural power isolation, and theoretical limits of embedded sensing systems. The specific scientific contributions are organized and enumerated below by theoretical, systems-engineering, and cross-disciplinary impact.

Theoretical & Mathematical Foundations

C1. Parameterized First-Principles Execution Model: Derived and validated a first-principles performance model relating heterogeneous core island behavior (Big $\approx$ Little) and non-linear execution parameter interactions on asymmetric SoCs [49, 50, 53] [Ch. 4, 5].

C2. Derivation of Novel Path Metrics: Introduced and empirically verified structural equation parameters for performance degradation tracking [48, 51, 52][Ch. 5, 7]:

- *Complexity Sensitivity* ($\sum \beta_i M_c^j$) and *Critical Complexity Index* (χ).
- *Inter-Island Scaling Friction* ($M_c \times I_L$) and *Multi-Core Saturation* ($\Delta\beta_i C_3 \to C_4$).
- *Ghost Power Coefficient* (γ), quantifying dynamic static leakage coupling ($\gamma \approx 0.8$ on monolithic rails vs. $\gamma \approx 0.5$ under pseudo-isolation).

C3. Measure-Theoretic entangling Baseline: Formulated a mathematical baseline for "architectural entanglement" by evaluating uncoupled systems through convolution

commutativity ($\Phi[\mu_1 * \mu_2] = \Phi[\mu_1] * \Phi[\mu_2]$), bridging statistical physics and hardware diagnostics [52][Ch. 7].

C4. Entropy-Bounded Performance Model (EBPM): Applied the Boltzmann Uniqueness Theorem to establish the EBPM framework, quantifying fundamental throughput-latency bounds and explaining hardware jitter and entropy collapse in embedded sensing architectures [55][Ch. 8].

Systems Architecture & Empirical Discoveries

C5. Microarchitectural Bottleneck Isolation via SEM: Integrated low-level hardware performance counters (`perf`) with Structural Equation Modeling (SEM) measurement and structural models to validate first-principles assumptions, quantifying microarchitectural starvation and LLC cache-level bottlenecks [49, 50, 53][Ch. 4, 5].

C6. Disproof of Software-Defined Power Isolation: Demonstrated empirically that software-level thread affinity fails to isolate electrical power domains on modern SoCs, invalidating the traditional additive power assumption ($P_{\text{total}} = \sum P_{\text{core}} + P_{\text{uncore}}$) in favor of relational matrix models ($P_{\text{total}} = V^T M f$) [52, 54][Ch. 7].

C7. Characterization of Inter-Cluster Parasitic Mechanisms: Identified and quantified physical coupling phenomena across target silicon platforms [52][Ch. 7]:

- Fabric Drag: Quantified systemic parasitic load (e.g., active Big core stress inducing 1.275 W draw on an idle Little cluster via shared interconnect frequency elevation).
- Monolithic Voltage Locking: Quantified DVFS failure modes on single-rail architectures where active core voltage pins shared rails at peak levels.
- Inverse Coupling: Identified LLC starvation-mediated coupling under PL1/PL2 power envelopes.

Cross-Disciplinary Impact

C8. Power-Rail Aware OS Scheduling: Formulated empirical directives for operating system schedulers to co-locate threads on shared physical voltage domains, minimizing parasitic power draw in idle domains [52][Ch. 7].

C9. Generative AI Structural Priors (SE-VAE Integration): Derivation and empirical validation of measurement-aligned structural equation models that formulate the domain-specific latent priors and error covariance structures required for Structural Equation Variational Autoencoders (SE-VAE) [134]. This provides the theoretical foundation to isolate construct-specific execution variance from high-amplitude microarchitectural nuisance noise in deep generative latent spaces [52][Ch. 7].

C10. Embedded Biomedical Sensing Paradigm Synthesis: Resolved throughput-latency conflicts in embedded sensing, synthesizing computer engineering principles with biomedical application constraints [55][Ch. 8].

1.2 Outline

The outline of the remainder of the dissertation is as follows. Chapter 2 gives background and related work. Chapter 3 introduces the SAC methodology, providing evidence of the complexity of the modeling problem and a few clues as to how we will address it. Chapter 4 provides our first-principles performance model for the Orange Pi platform and uses SEM techniques to validate the choices made in its development. Chapter 5 goes further by including the Cubie A7A platform into the investigation to explore and articulate the direct and quantifiable performance implications of workload and platform interactions. Chapter 6 delves into Structural Equation Modeling and its usage for articulating performance for homogeneous in-order processors, followed by an investigation of its application to heterogeneous out-of-order processors and draws conclusions on how SEM models may unify performance modeling for both processor types. Chapter 7 explores heterogeneous platforms with respect to power consumption and using the Boltzmann Uniqueness Theorem, demonstrates that modern asymmetric multi-processors exhibit parasitic power consumption and are fundamentally entangled at the silicon level. Chapter 8 expands our application of the Boltzmann Uniqueness Theorem to applications in biomedical engineering, wherein we

discover that the theorem can be extended to balance the polarized and contrasting technical (throughput-based) and clinical (latency-based) requirements simultaneously. Finally, in Chapter 9 we discuss the overlap and distinctions between workloads, platforms, and applications that we have explored in this work and draw conclusions that may direct future work.

Chapter 2

Background and Related Work

2.1 Performance Modeling

2.1.1 Performance Classification

Given the variety of multiprocessor systems available, it is important to consider the inherent parallelism execution units deliver for high performance workloads at scale. The practice of mapping workload threads onto uniform cores can be considered as an instance of the knapsack problem [23, 127]. Unfortunately, there is no known polynomial-time algorithm that can determine if a valid workload mapping is optimal. Typically, heuristics can determine good (though possibly sub-optimal) configurations, however, and simply considering the behaviors of each device can be crucial to determine the most appropriate mapping. Workloads can be sensitive to changes in data types, data sizes, and (processor- or memory-) bounds on their execution. Consequently, a great deal of investigation into platform architectures has enabled characterization of performance impacts of specific design choices on workloads.

MICROARCHITECTURE: The microarchitectural properties of a processor core clearly have a profound impact on performance assessment, as investigated in [2, 20, 31, 82, 87, 104], including instruction pipelining, cache hierarchy, branch prediction, and data path widths.

SOFTWARE ORCHESTRATION: The heterogeneous multi-core landscape presents unique challenges in task orchestration and scheduling. Some of the more prominent challenges include: task mapping and granularity, dynamic workload management, overhead management, and security and isolation [83, 91, 110, 119, 132].

MODELING APPROACHES: The techniques used for performance modeling cover a wide range. Novel approaches suggested in [3, 29, 63, 64, 79, 80] include: reducing the cost of simulations, statistical parameter modeling, sampling, and application-specific modeling.

Elaborating on some of the literature cited above, Pagani et al. [91] emphasize the importance of a management layer between the system software and the acceleration platform that takes task-specific considerations into account to facilitate shared-memory communication, improve hardware reconfiguration, and support scheduling.

Shirazi et al. [110] identify that the numbers of instructions for workloads are increasing and particular operations in the workloads take considerable time. Instead of using generic execution units for these work-intensive instructions, it would be worthwhile to architect custom execution units that exclusively operate on these sub-tasks of the workload.

Trimberger et al. [119] clearly outline how the physical hardware is the very first line of defense for securing any platform and expose several weaknesses and fortifications that impact platform security. Yet, few if any of the considerations are taken into account to assess the impact that each design choice may have on performance, scheduling, and timing behavior.

Yue et al. [132] consider the affinity of workloads for available execution units. Instead of tailoring a platform to a specific workload, the authors seek to understand how the workload operates on the given platform and classify workloads according to their affinity to the specific processor features and capabilities. Such affinity becomes a necessary gauge for identifying the appropriate platform with a given set of constraints.

Moving from classification schemes with emphasis on real-valued performance metrics (e.g., execution time, IPC, cache-misses, etc.) to qualitative performance scales (e.g., capable/incapable, slow/fast) supports resource planning and workload scheduling. Given the pervasiveness of energy-constrained computing environments at the edge, the identification of workload affinity becomes ever more important. Obtaining sufficient measurements in a polymorphic design space can be an intractable problem, but regression modeling as proposed in [80] is a powerful tool to assess performance accurately for arbitrary processor configurations in large micro-architectural spaces.

It also is becoming popular to leverage statistical inference and machine learning for approximating solutions in large intractable design spaces. Gaining traction in computer performance modeling, statistical models have been shown in [63] to detect trends in performance metrics, which can be used to establish baseline performance, automate anomaly detection, and discover sources of bottlenecks. These findings suggest that any behavior both undesirable and constrained is reflected by the platform performance data.

Methods to explore exponentially increasing design spaces efficiently remain an open problem. As supported by the work of [64], design space exploration can be accelerated by identifying important design parameters, utilizing targeted sampling, and leveraging fast and accurate predictions afforded by artificial neural networks and other structures. Such methods applied to these design spaces demonstrate broad applicability to relevant workload modeling challenges.

The tasks we consider here are compact (e.g., many of the empirical results we present are for a matrix-matrix multiply kernel). There has been considerable work in modeling the performance of applications that are composed from compact tasks. For streaming data applications, queuing theory [38], flow analysis [8], and network calculus [37] have been applied to heterogeneous systems.

Looking specifically to the ARM big.LITTLE platform, Wang et al. [123] develop a model for convolutional neural network inference that they then use to guide placement of tasks onto both types of cores. Zhu and Reddi [136] use statistical inference to model mobile web browsing performance with the goal of meeting latency constraints while consuming the least amount of energy. Butko et al. [18] developed a simulation model of the ARM big.LITTLE platform that predicts both performance and power consumption.

2.1.2 Performance Abstraction

Hierarchical performance attribution provides a structural link between raw measurement data and actionable system insights. Across microarchitectural execution, parallel software execution, machine learning models, and analytical queueing frameworks, that work establishes that aggregate system outcomes—such as overall latency, scaling bottlenecks, or model outputs—cannot be properly understood by inspecting isolated components.

Organizing performance data into tree structures, calling contexts, feature clusters, or decomposed queuing networks, enables recursive isolation of exact root causes of inefficiency or inaccurate prediction without getting overwhelmed by system complexity. Furthermore, we may be able to bridge empirical attribution with formal system evaluation, and thereby demonstrate that hierarchical decomposition may prove to be a powerful abstraction for analytical queuing models as it is for physical measurement tool-chains. We outline key efforts that apply hierarchical abstractions to common domains as follows:

Hardware Microarchitecture: Yasin et al. [131] define the Top-Down Microarchitecture Analysis Method (TMAM) which establishes a hierarchical classification leveraging hardware performance counters to partition microarchitectural execution bottlenecks systematically.

Execution Context & Software Profiling: Tallent et al. [115] address the attribution of indirect runtime overheads in parallel computations by using call-path profiling to quantify parallel idleness and overhead in work-stealing computations, pinpointing the application regions responsible for scalability bottlenecks. Adhianto et al. [1] present the architecture of HPCToolkit, an asynchronous sampling suite, with their work detailing the attribution of performance metrics across hierarchical program representations, mapping execution measurements to calling contexts, program regions, and source-level constructs.

Machine Learning Interpretability: Murdoch et al. [90] introduce Agglomerative Contextual Decomposition (ACD) for hierarchical interpretation of deep neural network predictions. ACD produces a hierarchy of input-feature clusters and estimates each cluster's contribution to the prediction, enabling interpretation of feature interactions rather than isolated variables.

Analytical System Modeling: Thomasian [118] surveys hierarchical performance evaluation and analytical modeling (rather than empirical measurement attribution), and reviews the decomposition of complex queueing network models, continuous-time Markov chains, and multi-tier transaction systems into solvable hierarchical subsystems.

These works illustrate that connecting direct attribution to formal performance modeling is vital for building predictive, next-generation computing systems. While empirical attribution explains *where* time or resources were spent in a past execution, performance modeling leverages those hierarchical abstractions to predict *how* a system will behave under future workloads, hardware alterations, or scaling regimes. As modern architectures grow increasingly heterogeneous and complex the incorporation of multi-chiplet modules, non-uniform memory topologies, and deep AI accelerator stacks can make flat performance models intractable. Future modeling paradigms may rely on hierarchical attribution to construct modular, scalable representations where individual subsystems can be evaluated, replaced, or simulated in isolation before being composed back into a full-system model.

2.1.3 Performance Representation

Traditional performance analysis in computer engineering typically aggregates core power and execution latency using simple scalar addition. This particular approach is known as the "sum of parts" methodology that is commonly used for performance modeling [50, 17, 81, 99, 66]. However, empirical diagnostic data from heterogeneous platforms like the Rockchip RK3588 and Allwinner A733 reveal that this additive heuristic is fundamentally underspecified due to "architectural entanglement." When core islands share critical microarchitectural fabric, such as LLCs or primary voltage rails, their operational states are not computationally orthogonal but instead are interactively coupled. To capture these non-linear interactions accurately, we shift the investigation toward measure-theoretic representations, where system behavior is modeled as a distribution of states rather than isolated variables. By utilizing the mathematical properties of convolutions, engineers can evaluate cluster independence formally: a system is truly "uncoupled" only if its joint resource measurement adheres strictly to the convolution of its subsystem measures.

Consequently, performance modeling in heterogeneous systems requires an algebraic shift from scalar heuristics to measure-theoretic representations, specifically utilizing Probability Generating Functions (PGFs) to encode execution states. The convolution of discrete probability measures is isomorphic to polynomial multiplication. However, the semiring of polynomials with non-negative coefficients, representing physically realizable energy states, is not a Unique Factorization Domain (UFD). Therefore, the resulting structural constraints are required to parameterize the Boltzmann map uniquely. These mathematical foundations

are grounded in *support-preserving endomorphisms*, which verify formally that the measure-theoretic support of a workload remains invariant under state transitions. Such mappings ensure that any energy level architecturally defined in the hardware remains reachable under a given execution profile, bridging *combinatorial sampling theory* with formal *reachability analysis*. The most pertinent formalisms related to our work include:

Polynomials and PGFs: Probability measures are often represented as PGFs, where convolution of measures is equivalent to the *multiplication of polynomials*. The theorem relies on the fact that the set of polynomials with non-negative coefficients (representing energy states) is not a unique factorization domain, which provides the constraints needed to fix the Boltzmann map [32, 105].

Support-Preserving Endomorphisms: Specifically in hardware, the "support" of a measure refers to the set of energy levels a system can actually attain. A support-preserving map ensures that if a state is physically available in the hardware, it must be reachable under any execution profile [5, 56].

Microarchitectural Bottlenecks: System components like the LLC connect all cores, creating the physical path for the "starvation bottlenecks" that the Boltzmann theorem detects. The theorem proves that because these clusters share the LLC, their distributions cannot be convolved independently, making linear performance models (like those based on volume alone) underspecified [19, 67].

2.1.4 Performance Convergence

Investigations into heterogeneous ARM processors present new challenges and opportunities for extracting both performance and understanding of the characteristics of workload execution on these platforms. Yet, the precise characterization of performance elicited from nuanced heterogeneous edge architectures remains fundamentally constrained by the limitations of prevailing analytical paradigms. Classical performance bounds and portability abstractions establish operational baselines but systematically fail to resolve the stochastic non-linearities, memory contention, and opaque driver behaviors inherent in modern edge deployments. Consequently, critical operations such as General Matrix-Matrix Multiplication (GEMM) within Transformer self-attention mechanisms are inadequately modeled by

standard throughput or linear scaling assumptions. To bridge the deficit between theoretical performance proxies and the volatile execution dynamics of edge intelligence, this work introduces a unified analytical framework. By reformulating bottleneck evaluation through Structural Equation Modeling (SEM) which will be expanded on in the next section, this architecture simultaneously quantifies the multi-variate, latent dependencies between workload parameters and hardware saturation points. This approach synthesizes previously disjointed methodologies into a cohesive model, a subset of which are outlined below, to establish a high-fidelity alternative to traditional predictive approximations as follows.

Performance Models: Traditional performance modeling relies on asymptotic analysis for the PRAM model [41] and the Roofline Model [126]. While the Roofline Model identifies bounds based on operational intensity and peak bandwidth, it often fails to capture the non-linear performance "valleys" and "plateaus" inherent in heterogeneous edge hardware. Existing research into execution time models [98] for GEMM frequently assumes linear scaling with respect to FLOPs, overlooking the stochastic nature of memory contention and cache misses in non-uniform memory access (NUMA) architectures. *Analytical models* are rooted in theoretical frameworks such as queuing theory, fundamental physics, and direct simulation of microarchitecture features [50, 65, 73, 75, 121, 129]. In contrast, *empirical models* derive their structure, parameters, and prediction capabilities from direct measurements of performance, power, resource usage, etc., requiring statistical techniques including regression analysis, clustering, and neural networks [6, 57, 97, 128].

Heterogeneous Edge Intelligence: Current literature on AI/ML at the edge focuses on hardware-specific optimizations for Transformers and CNNs [130]. Research into performance portability (e.g., SYCL [39], Kokkos [120]) attempts to abstract hardware layers, yet performance prediction remains difficult due to the "black-box" nature of proprietary drivers and varying compute-to-memory ratios [120]. Recent studies have identified that transformer inference latency is dominated by GEMM operations, specifically within the self-attention mechanism, necessitating high-fidelity proxies that go beyond simple throughput metrics [135]. With the advent of ML tools, including K-means clustering, neural network classifiers, support vector machines, and nearest neighbor with generalized examplars, researchers can scale behavior trained on hardware performance counters and infer predictive models of GPU speedup based on profiles collected from CPU execution [6, 128]. First-principles analytical models, such

as those used in cycle-level simulation tools [75], provide the highest microarchitectural fidelity, as they operate on fundamental timing and resource models. Similarly, classical queuing networks for multicores [73] can offer pertinent theoretical insight into system behavior based on mathematical laws. Heterogeneous platforms require a hybrid bridge between the modalities. There is significant research that uses analytical equations derived from physics (e.g., DVFS power models [75]), algorithmic complexity (e.g., HPC matrix-matrix multiply [57]), or network topology (e.g., LogP/GPH for MPI communication [97]) while relying on empirical measurements along with regression or quadratic optimizations to determine parameters and calibration factors. Hybrid models are essential for the interpretibility of analytical models without the complexity of modeling every microarchitectural non-linearity manually.

2.2 An Introduction to Structural Equation Modeling

Structural Equation Modeling (SEM) is a set of multivariate statistical analysis techniques that includes multiple regression, factor analysis, and path analysis [61, 78]. SEM has been widely used in a number of fields, including the social sciences, psychology, education, and business [44, 60]. SEM techniques allow for the direct estimation of complex relationships between multiple variables. These SEM techniques are distinct from traditional statistical methods because SEM can model and test complex relationships between observed and latent variables simultaneously.

Model parameters serve as the fundamental building blocks for constructing a SEM model. The observed variables are denoted as rectangles, and the latent variables are defined as circles, with double arrow lines indicating variance/covariance, and single arrow lines showing the regression path. A basic SEM model example, which defines the latent variable Y regressed on the latent variable X, is shown in Figure 2.1. The latent variable Y has three factors $\{y_1, y_2, y_3\}$ with the relationship between the latent variable Y and each measured variable $\{y_i\}$ defining its measurement model. Similarly, the latent variable X has three factors $\{x_1, x_2, x_3\}$ and the relationship between the latent variable X and each measured variable $\{x_i\}$ defines the X measurement model. Finally, the relationship between latent variables X and Y defines the structural model. The SEM model framework is a powerful technique that allows for the simultaneous estimation of all relationships in the model.

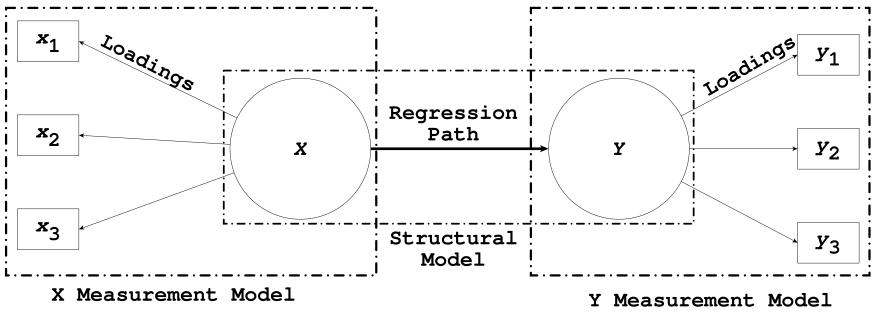


Figure 2.1: SEM Model Structure

SEM is focused on variations in, and relationships between, quantities themselves (e.g., variance, covariance, correlation, autocorrelation, etc.) rather than for example the means of those quantities. As described by Hoyle [60], often in SEM the goal of estimation is "the minimization of the difference between the observed covariance matrix and the covariance matrix implied by the model." This is a clear distinction from models which are trying to, e.g., minimize the sum of squared differences between model output and empirical observation.

Some key characteristics of SEM:

Modeling Complex Relationships: SEM gives researchers the flexibility to test hypothesized causal relationships among a set of observed and unobserved (latent) variables simultaneously. It can model direct and indirect effects, and accommodate situations where variables serve as both predictors and outcomes.

Latent Variables: One notable distinguishing feature of SEM is its ability to incorporate "latent variables." These are theoretical constructs or concepts that cannot be measured directly (e.g., intelligence, customer satisfaction, motivation). On the contrary, they are inferred from multiple directly observed "indicator" variables (e.g., scores on different tests, survey questions, etc.). SEM explicitly accounts for measurement error in these observed indicators, providing more accurate estimates of the relationships between the underlying latent constructs.

Graphical Representation: SEM models are often represented visually using path diagrams (similar to flowcharts). Circles and/or ovals typically represent latent variables, squares or rectangles represent observed variables, single-headed arrows indicate

hypothesized causal effects, and double-headed arrows indicate correlations or covariances.

Model Fit Assessment: A critical aspect of SEM involves evaluating the "goodness-of-fit" for the hypothesized model to the observed data. There are various fit indices (e.g., Chi-square (χ^2), Root Mean Square Error of Approximation (RMSEA), Comparative Fit Index (CFI), Tucker-Lewis Index (TLI)) that are used to determine how well the proposed model accounts for the relationships (covariances) observed in the dataset.

Essentially, SEM assists researchers in confirming or falsifying theories by testing how the relationships among variables, including abstract concepts, align with the empirical data. In our work, we will use SEM measurement models to examine the parameter selections made in our first principles model above and then follow that using SEM structural models to examine the refinement.

SEM Constructs: Reflective vs. Formative

Though numerous methods are available to model platform performance in SEM, we will focus on the two most prominent methodologies:

Reflective – Covariance-Based (CB-SEM): Optimized for theory confirmation and hypothesis testing. It focuses on sample covariance matrix reproduction and requires large sample sizes alongside multivariate normal distributions.

Formative – Partial Least Squares (PLS-SEM): Utilized primarily for exploratory research and predictive objectives. It focuses on maximizing the explained variance (R^2) of endogenous constructs and exhibits robustness with smaller sample sizes and unknown data distributions.

SEM makes assumptions regarding the causal relationships between endogenous (dependent) and exogenous (independent) constructs, which have dramatic implications for model selection. In the case of a *reflective model*, the latent variables are the cause of the indicator

variables, with all indicator variables inferred from a latent construct assumed to be correlated and interchangeable. This follows, as the indicator variables are considered mere reflections of the latent construct that exists in an absolute sense, independently of measures. The *formative model* assumes that the indicator variables are the cause of the latent construct. The set of indicator variables forms a composite that defines the latent construct. The formative indicator variables are assumed to be uncorrelated and non-interchangeable as the latent construct is sensitive to both the quantity (number) and constitution (type) of the indicator variables forming the latent construct.

SEM Path Diagrams

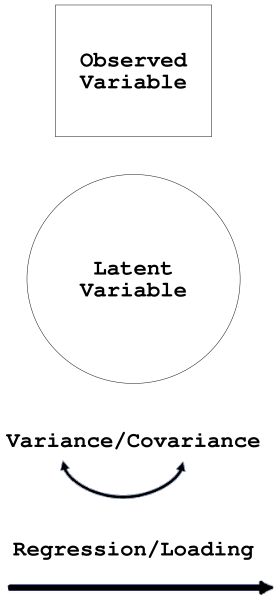


Figure 2.2: SEM Path Elements

With regard to the graphical summaries provided by SEM as depicted in Figure 2.2, there are several fundamental elements that we outline as follows:

Squares or Rectangles (□): Represent observed variables (also called manifest variables, indicators, or measured variables). These are variables for which we have direct data (from measurements, experiments, or literature).

Circles or Ovals (○): Represent latent variables or error terms (i.e., residual variance). In contrast to observed variables, latent variables are constructs that cannot

be measured directly but are inferred from *observed variables* with the *error terms* accounting for *leftover variance* not explained by other measurements and variables.

Single-Headed Arrows ($\rightarrow$): Represent *regression paths* or factor loadings, also known as *direct effects* which are used to indicate a hypothesized causal relationship or a directional influence from one variable to another.

Double-Headed Arrows ($\leftrightarrow$): Are used to represent covariances or correlations within individual SEM elements (i.e., error terms) or across elements (i.e., between latent variables), but the relation does not imply a direct causal direction.

Graphical representations of SEM models are powerful tools for both defining and analyzing complex relationships between variables. SEM is unique in that it combines aspects of factor analysis and regression analysis into one single analysis allowing for the estimation of multiple, potentially interrelated dependent variables and structures. It consists of two main parts: the measurement model and the structural model:

Measurement Model: The measurement model in SEM defines how latent variables (unobserved constructs) are measured by their observed variables which is used to assess the validity and reliability of the measurement of these latent constructs. Essentially testing whether the observed variables adequately capture the underlying latent variable they are intended to represent. Confirmatory Factor Analysis (CFA) is the most common method used to test a measurement model. It confirms whether the observed data fits a pre-specified theoretical model of how indicators load onto latent factors.

Structural Model: The structural model in SEM defines the hypothesized causal relationships and direct/indirect effects between the latent variables themselves to test theoretical relationships and pathways among these unobserved constructs by examining how changes in one latent variable are hypothesized to cause changes in another. After a measurement model's validity is established, the structural model is tested to see how well the hypothesized relationships among the latent variables fit the observed data which often involves examining path coefficients, R-squared values, and overall model fit indices.

SEM Measured Variable Path Analysis

A complementary resource to both CB-SEM and PLS-SEM methodologies is Measured Variable Path Analysis (MVPA) [89], which tests casual relationships among observed variables, enabling precise insights into parameter impacts. More specifically, as MVPA focuses exclusively on directly measured variables, this allows investigation into networks of hypothesized causal effects, including both direct and indirect pathways to understand the complexity with which variables influence each other and enables quantitative delineation of both the magnitude and strength of that influence, which is of critical importance where the impact of collections of heterogeneous components needs to be quantified. Starting with generic path analysis, we iterate through the parameters until we obtain a collection of variables meeting principled criteria for performance modeling with emphasis on measured variables, path coefficients, and direct effects [47, 103].

SEM Model Articulation

We will explore several models to understand these tools and associated workload-to-platform relationships better, revealing how these techniques articulate platform behaviors, model heterogeneous performance characteristics, and assist in identifying pertinent parameters for specific performance goals. The power of SEM comes from its ability to measure and assess the relationships between both latent variables (unobserved constructs) and indicator variables (measured constructs) simultaneously, using statistical analysis to model the variance or covariance between them. Given a model composed of one or more exogenous (indicator) variables regressed on one or more endogenous (latent) constructs, changes in the latent variables can be detected or influenced through the indicator variables. These relationships can be analyzed using either CB-SEM or PLS-SEM, both of which we will explore in our analysis. As the models we develop are heavily data-dependent, our goal is to characterize and understand the relationships between platform parameters and workload performance, rather than only on developing an opaque predictive model to serve as a performance estimator.

2.3 Boltzmann Uniqueness: Recontextualizing Performance Dependencies

The synthesis of the performance formalisms (i.e. classification, representation, and convergence) addresses the inherent limitations of linear performance models, which remain underspecified for modern systems-on-chip (SoCs) incorporating big.LITTLE clusters, GPUs, NPUs, and FPGAs. In many of these architectures, the LLC functions as a central coupling point, inducing microarchitectural "starvation bottlenecks" that manifest as stochastic dependencies between otherwise disjoint accelerators. The Boltzmann theorem demonstrates that the shared residency in the LLC precludes the independent convolution of cluster-specific energy distributions, necessitating an orchestration framework capable of detecting *non-linear resource contention*. Consequently, a Boltzmann-centric modeling approach is essential for future heterogeneous acceleration, providing the analytical rigor to optimize resource allocation in environments where non-orthogonal resource sharing invalidates traditional volume-based throughput estimations.

As stated above, the Linux EAS scheduler [95] makes independence assumptions about the state of Big and Little cores. The following well-known computer engineering models utilize additive summation or independent domain assumptions that are rendered underspecified by the coupled-domain characteristics of modern heterogeneous SoCs:

McPAT (Multicore Power, Area, and Timing) [81] is a widely utilized framework that models power by aggregating the dissipation of independent hardware structures (ALUs, caches, branch predictors). It treats "uncore" components as a constant or a simple linear function of memory traffic.

Wattch [17] is a framework which provides architectural-level power analysis by summing struct-by-struct estimates based on conditional clock gating and activity factors. Wattch assumes that if a hardware structure is gated (inactive), its contribution to the power sum is negligible.

RAPL (Running Average Power Limit) [99] assumes domain separability, reporting energy for discrete domains such as Package (PKG), Power Plane 0 (PP0/Cores), and DRAM. RAPL assumes that core-level activity (PP0) can be treated independently from uncore/package overhead.

The Roofline Model [126] identifies performance bounds based on operational intensity and peak bandwidth, assuming that compute-bound and memory-bound states are strictly demarcated by the physical limits of the core and bus. The model fails to account for the non-linear interaction between workload parameters (e.g., matrix dimension M) and core island configuration.

BCMP Queuing Networks [7] traditionally are used in multicore performance evaluation to model CPU and memory subsystems as independent queuing servers. This approach relies on the assumption that components are partitioned and operate as decoupled service nodes.

To support performance modeling for these platforms, we utilize the Boltzmann uniqueness theorem [13], which serves as a rigorous mathematical bridge between *statistical mechanics* and the *architectural orchestration* of heterogeneous computing systems.

2.4 Real-Time Maternal Hemorrhage Monitoring

Postpartum hemorrhage (PPH) constitutes a leading cause of maternal mortality worldwide. Mitigating PPH has profound regional disparities and systemic determinants [27]. With respect to the clinical setting, early recognition and intervention are critical for effective management and morbidity reduction [34, 125]. Currently, the traditional diagnostic paradigms, both in the United States and abroad, have relied heavily on the visual assessment of blood loss. This approach has been documented to underestimate hemorrhage volume systematically when compared to alternative quantitative techniques such as drape estimation [92]. To address the limitations of these assessment techniques, digital devices (such as wearable apparatuses) are being developed to streamline clinical workflows and improve response times [116]. Despite these advancements, there is a persistent and critical demand for autonomous, continuous, and non-invasive monitoring systems which are capable of detecting physiological deterioration prior to the onset of overt hemorrhagic shock.

Consequently, efforts to transition away from tethered clinical instrumentation to continuous maternal monitoring increasingly are driven by wearable technologies that employ soft, flexible, and wireless sensor networks [100]. Furthermore, optical sensing methodologies have emerged as a promising modality for such applications, while offering non-invasive probing

of peripheral hemodynamics. The recent literature has reported the development of wearable optical devices specifically engineered to detect postpartum hemorrhage, specifically by measuring peripheral vascular compensation [33]. Most notable and revelvant to our investigation, is the Maternal-aRMOR system which integrates optical sensors into a wearable form factor, enabling early detection of compensatory vasoconstriction mechanisms that are indicative of central blood volume depletion [15]. The success of this optical approach in detecting early-stage hemorrhage has been evaluated rigorously in controlled in vivo studies that have demonstrated reliable tracking of peripheral vascular changes in a swine model of hemorrhage studies [14]. Additionally, the translation of continuous monitoring devices into practical clinical tools requires the mitigation of strict hardware and performance constraints. More specifically, the devices require reliable energy storage and power management in implantable or wearable biomedical devices [11].

Laser Speckle Constrast Imaging (LSCI) has been determined to be a promising technique frequently leveraged for assessing microvascular blood flow in these wearable systems. The LSCI approach enables fluid dynamics quantification via analysis of the spatial and temporal statistics of interference patterns generated by coherent light scattered from moving particles within the body such as erythrocytes (red blood cells) [106, 107]. When coherent laser light illuminates tissue, moving red blood cells cause intensity fluctuations that blur the captured speckle pattern. Spatial or temporal blurring is quantified as speckle contrast K:

$$K = \frac{\sigma}{\langle I \rangle} \tag{2.1}$$

where σ and $\langle I \rangle$ represent the standard deviation and mean of pixel intensity I, respectively. Compute platforms process this raw statistical measurement to compute the Laser Speckle Flow Index (LSFI) [16]. Mathematically, K maps to blood flow via decorrelation time τ_c and camera exposure time T:

$$K = \frac{1}{\sqrt{1 + \frac{2T}{\tau_c}}} \tag{2.2}$$

Increased flow velocity accelerates decorrelation (smaller τ_c), resulting in reduced speckle contrast K. Capturing pulsatile waveforms and rapid transients requires targeted exposure times (0.1 ms to 1 ms for vessels, $\sim$3 ms for parenchyma) [107]. Validation in swine models demonstrates a 94.0% correlation between LSCI flow metrics and actual blood loss volume [14]. As will be developed later in this work, there exist theoretical frameworks which govern both particle kinetics and interaction probabilities derived from foundational statical mechanics, including adaptions of the Botlzman distribution and the associated H-theorem to model enable behaviors observed in scattering media [101, 117]. Though LSCI has a robust theoretical basis, it remains susceptible to some practical limitations, including constrained penetration depth and susceptibility to motion artifacts [16]. There also exists advanced configurations, including dual-wavelength LSCI (dwLSCI), which has been introduced to address these constraints and facilitate improved chronic measurement of superficial blood flow dynamics [133].

The challenges associated with the translation of high-bandwidth spatial and temporal LSCI data into intuitive, responsive, and actionable clinical information necessitates robust computational frameworks at the network edge [111]. Specifically, edge-based AI paradigms minimize latency, preserve data privacy, and ensure platform functionality in low-resource environments that are completely independent of existing cloud infrastructure. With the adoption of commercial off-the-shelf (COTS) single board computers, such as the Raspberry Pi, these platforms have demonstrated substantial capabilities in terms of executing complex medical diagnostic tasks locally. Such examples of deployment scenarios include the use of deep learning architectures on Raspberry Pi platforms for analyzing COVID-19 CT scans and chest X-rays [58, 114], along with real-time processing of single-channel electroencephalogram (EEG) signals for traumatic brain injury detection [28]. Adapting COTS platforms to edge-based challenges has enabled the use of optical wearable sensors that support continuous, real-time risk stratification necessary for predicting PPH.

By viewing these challenges from a computational systems perspective, the process of ingesting and responding to high-fidelity optical signals on wearable platforms introduces strict constraints regarding timing, thermal dissipation, and power consumption. Prior analysis of biomedical signals [96] under continuous monitoring conditions has been shown to require deterministic execution which necessitates hard real-time scheduling algorithms specifically tailored for heterogeneous multiprocessor architectures [26]. Furthermore, rigorous algorithmic optimization of both thermal and power management is mandatory in the prevention

of computational throttling, support to extend battery life, and assurance of patient safety by limiting localized device heating [10]. Finally, advanced spatio-temporal modeling will be required in order to translate raw optical contrast matrices into predictive clinical indicators. Utilizing techniques adapted from collaborative filtering and multi-dimensional state estimation [86] offers alternative computational pathways to synergize temporal hemodynamic trends with spatial vascular maps, enabling critical optimization of early detection capabilities of the hardware.

2.5 Experimental Platforms

For this dissertation, the empirical experimentation utilizes two octacore platforms, specifically the Orange Pi 5 and Cubie A7A, which are illustrated in Figure 2.3 and Figure 2.4, respectively. We also utilize an Intel platform that uses an entirely different ISA, compute resources, and memory management techniques, yet has fundamentally the same underlying heterogeneous characteristic (i.e., AMP platform). It is illustrated in Figure 2.5.

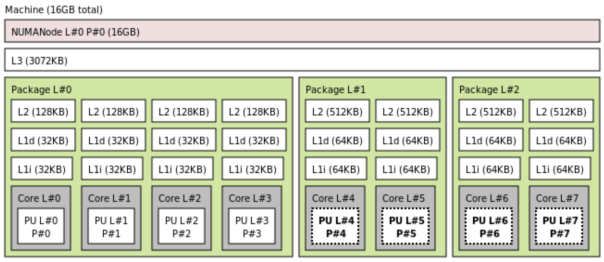


Figure 2.3: Orange Pi 5 Core Configuration

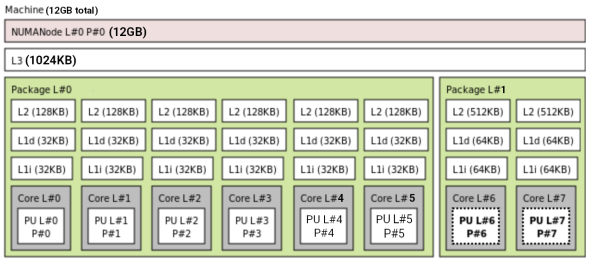


Figure 2.4: Cubie A7A Core Configuration

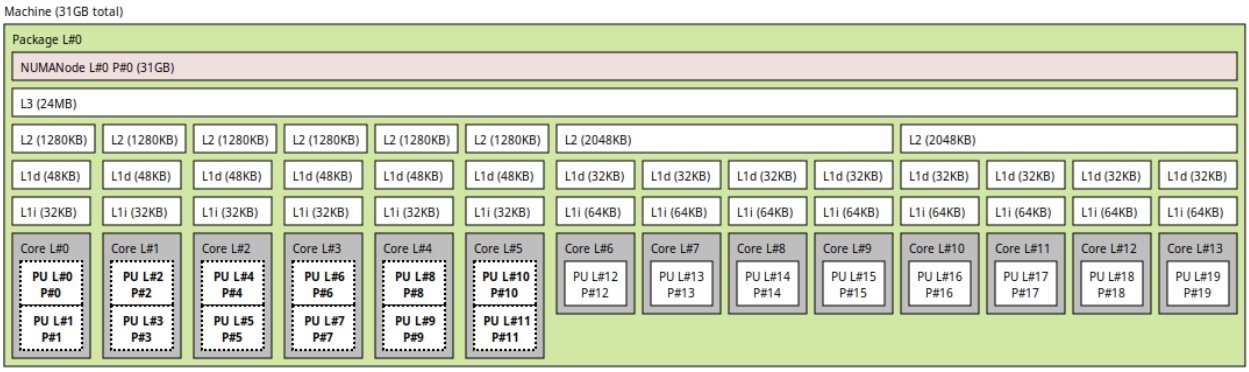


Figure 2.5: Intel i7-12800H Core Configuration

Irrespective of their differences, these platforms offer a variety of compute nodes ranging from high-performance ("Big") cores to energy-efficient ("Little") cores, each with unique performance implications. We outline the architecture for each as follows:

Orange Pi 5 - Rockchip RK3588S (4 Big Cortex-A76, 4 Little A55): This device represents a modern, high-performance AMP exhibiting balanced big.LITTLE cluster performance. As shown in Figure 2.3, the platform utilizes a Rockchip RK3588S big.LITTLE architecture, which consists of physical (Big and Little) islands, each with private L1/L2 caches, and one logical (Dual) island composed of at least one core from each physical island. All cores are connected via the Last-Level Cache, which comprises a quad-core Arm Cortex-A55 in Package L#0 and a quad-core Arm Cortex-A76 across Package L#1 and Package L#2. We denote the two Big islands as Big0 and Big1, respectively.

Cubie A7A - Allwinner A733 (2 Big Cortex-A76, 6 Little Cortex-A55): While architecturally symmetric in core types to the Orange Pi 5, this board exposes asymmetric big.LITTLE cluster performance. This device, like its Orange Pi counterpart, also utilizes a heterogeneous multicore configuration. In particular, the Allwinner A733 SoC is a modern octa-core SoC designed with a highly asymmetric big.LITTLE configuration consisting of two distinct physical islands: a power-optimized cluster of six energy-efficient Arm Cortex-A55 ("Little") cores, and a high-capacity cluster of two high-performance Arm Cortex-A76 ("Big") cores, denoted Package L#0 and Package L#1, respectively, in Figure 2.4. These clusters are integrated by a unified interconnect with shared access to high-bandwidth LPDDR5 memory.

Intel i7-12800H - 12th Generation Alder Lake-H (6 P-Cores, 8 E-Cores): This platform represents a transition from embedded edge SoCs to the high-performance x86 mobile workstation domain, utilizing a complex hybrid topology of 14 physical cores supporting 20 hardware threads. As depicted in Figure 2.5, the high-performance cluster (P-cores) employs hyper-threading to support two threads per physical core (CPUs 0–11), while the energy-efficient cluster (E-cores) executes single-threaded workloads (CPUs 12–19). The microarchitecture features 14 private L1 cache instances and 8 shared L2 instances, all integrated via a single, massive 24 MiB unified Last-Level Cache (LLC) shared across the entire 20-thread package. Unlike the fixed power-island strategies of DynamIQ Shared Units (DSUs) found in ARM platforms, the Intel architecture utilizes sophisticated Running Average Power Limit (RAPL) envelopes (PL1 and PL2), where the unified LLC and

shared package (PKG) power planes function as the primary physical media for architectural entanglement.

We will limit our discussion to the ARM platforms for now. Notice that these two platforms utilizes ARM big.LITTLE architecture and are both comprised of two physical (Big & Little) islands, with private L1/L2 caches, and one logical (Dual) island composed of at least one core from *each* physical island. All cores are connected via the last-level-cache. A few relevant performance variables and associated components applicable to both platforms are listed in Table 2.1 offering a plethora of configurations.

Table 2.1: ARM big.LITTLE Specifications

Variables	**Performance Components**
Number of Cores	ARM Quad-core A76 + Quad-core A55
Island Selection	High-Performance (Big) + Energy-Efficient (Little)
Core Frequencies	2.4 GHz down to 1.8 GHz
Power Levels	Conservative, Power Save, Performance

2.5.1 Workload

For the following workload, we consider the Orange Pi 5 and Radxa Cubie A7A architectures as a set of baseline frames of reference for investigation into the larger AMP design space (e.g., datacenters via Intel Xeon 6 [94], desktops via Intel Alder Lake, Raptor Lake [24], and mobile platforms via Apple M-series chips [62]). We develop our models from data measured on the Orange Pi 5 and then subsequently use that model structure to explore the Radxa Cubie A7A.

To exercise these platforms, we utilize matrix-matrix computations of the form: $C \leftarrow \gamma AB + \delta D$, where $\gamma, \delta \in \mathbb{R}$, $C, D \in \mathbb{R}^{M \times N}$, $A \in \mathbb{R}^{M \times K}$, and $B \in \mathbb{R}^{K \times N}$. The workload, authored in C++, is a common OpenMP parallelized "three-loop" naïve matrix multiply implementation. It does not make use of any optimizations that leverage prior hardware knowledge. This workload reflects a common general matrix multiply (GEMM) computation, parallelized with OpenMP for threads in the range $\{1, ..., 2P\}$, where P is the number of processor cores. The dimensions ($M = K = N$) of the square matrices are in the range $\{64, 128, ..., 8192\}$, which is the `matrix_dimension` attribute in the SEM models below and is also denoted by M. This workload is representative of computationally intensive requests that are common

for many edge-based applications, in particular signal processing (a historical task on the edge) and machine learning (a newly emerging task on the edge). We collected measurements using `perf` for each workload configuration.

To characterize execution behavior across the memory hierarchy, we adopt a formalized *cache-boundary taxonomy* derived from the specific workload dimensions of the experiments. By mapping matrix dimensions M to their double-precision data footprints ($F \approx 3 \times M^2 \times 8$ bytes), we identify three discrete execution regimes dictated by on-chip cache capacities and the associated interconnect topology:

The Instruction-Bound Regime ($T_{L1 \to L2}$): For scales $M \in \{64, 128\}$, the workload footprint (96 KB to 384 KB) remains largely within the private L2 cache capacity (512 KB) for the high-performance Big cores. While $M = 64$ immediately saturates the 32 KB L1 data cache, $M = 128$ begins to spill from the 128 KB L2 cache of the energy-efficient Little cores. Performance in this regime is primarily dictated by instruction-level parallelism (ILP) and core clock speed. In this state, the system is highly sensitive to *cpu migrations*, as moving a thread between clusters destroys "cache warmth" at the L2 level, causing a measurable latency penalty.

The Interconnect-Bound Regime ($T_{L2 \to L3}$): This transition occurs as the workload scales to $M = 256$ (1.5 MB), exceeding the private L2 caches of both core types and requiring residence in the Shared Last-Level Cache (LLC). On the Orange Pi 5, the 3 MB L3 cache effectively captures this footprint, maintaining parallel efficiency. Conversely, the Cubie A7A, with a smaller 1 MB LLC, encounters saturation at this scale, forcing premature DRAM access. This regime is the primary site of coefficient inversion, where the overhead of maintaining coherency across the shared interconnect begins to overwhelm the computational contribution of the Little cores, leading to a critical complexity index (χ), which we define in Section 5.5, that signals the threshold of utility for asymmetric scheduling.

The Memory Wall Regime $T_{LLC \to RAM}$:At $M \geq 512$ (6 MB+), the workload footprint exceeds all available on-chip cache for both architectures, hitting the "Memory Wall". At these scales, performance is no longer compute-bound but is strictly limited by main memory bandwidth and thermal throttling. This regime represents the "stress-case" where stochastic coupling between OS jitter and shared-resource contention dictates parallel time. On the asymmetric Cubie (2+6), the "funnel effect"

to the L3 cache triggers congestion collapse; the six Little cores act as a "net liability," generating excessive coherency traffic that "poisons" the memory bus for the two high-performance Big cores. Both of these will be outlined in Section 5.5.

2.5.2 Platform Attributes

Given a general set of parameters as shown in Table 2.2, we seek to understand how these parameters affect application performance. Most of the parameters are self-explanatory, and are measured using the `perf` utility. The exception is the *island* parameter (e.g., big, little, dual) that we relabel as `island_numeric` by re-specifying the parameter using ordinal encoding.

Table 2.2: Platform Attributes

`threads`	`instructions`
`cores`	`ipc`
`matrix_dimension`	`branch_misses`
`cpuid`	`l1_dcache_loads`
`cpu_utilization`	`l1_dcache_load_misses`
`context_switches`	`llc_loads`
`cpu_migrations`	`llc_load_misses`
`page_faults`	`island_numeric`
`cycles`	

To support our investigation, we will look at these parameters from many different perspectives. While statistical significance is critically important, some parameters may be correlated, thereby obscuring the ability for individual parameters to articulate platform behavior and characteristics properly within the model. An important tool to discriminate among parameters is the Variance Inflation Factor (VIF). This diagnostic tool is used in regression analysis to detect multicollinearity within a group of modeling variables. Multicollinearity occurs when two or more exogenous (independent) variables in a regression model are highly correlated with each other. This high correlation can make it impossible to determine the individualized effects of each variable on the endogenous (dependent) variable, which can lead to unreliable and unstable coefficient estimates. VIF $<$ about 5 to 10 is generally considered an acceptable range, but greater than that indicates high multicollinearity. We also look to avoid cases where parameter estimates are not within expected ranges, often referred to as *Heywood* cases.

Chapter 3

The Selection, Assessment, Classification Methodology

Modern heterogeneous architectures such as asymmetric ARM big.LITTLE System-on-Chip fabrics integrate out-of-order (OoO) performance cores and energy-efficient (efficiency) cores that introduce severe microarchitectural variance which destabilizes standard linear performance models. Symmetric, homogeneous platforms are typically comprised of uniform, in-order processors allowing for scalar metrics like instruction counts to serve as reliable proxies for execution time. However, within asymmetric hardware, the OoO processors expose dynamic execution cliffs, speculative stalls, and memory hierarchy bottlenecks that break isotropic linear assumptions. To resolve this instability and bypass the intractable combinatorial explosion of testing hundreds of thousands of hardware scenarios, the *Selection, Assessment, and Classification (SAC)* framework formalizes workload characterization into a structured, substrate-parameterized pipeline for Hierarchical Performance Attribution [1, 90, 115, 118, 131]. The SAC framework deconstructs execution modeling into three distinct functional layers:

> Selection: Maps application workload parameters to target instruction sequences, using in-order processors, to isolate software intent before runtime.
>
> Assessment: Maps those instruction sequences and hardware events to quantitative execution metrics.
>
> Classification: Partitions continuous execution profiles into discrete, qualitative performance categories like "Compute Bound" or "Memory Bound".

Rather than treating these intermediate hardware counters as independent regressors under multi-stage Ordinary Least Squares that which suffer from severe error compounding

on speculative out-of-order cores, this methodology leverages Structural Equation Modeling (SEM) to treat microarchitectural metrics as correlated, reflective indicators of an unobserved latent execution state. This unified covariance-based approach eliminates intermediate prediction noise, and as we will demonstrate for out-of-order processors in Chapter 6, restores predictive stability across heterogeneous hardware substrates, and transitions performance modeling from opaque empirical observations to transparent architectural articulation.

3.1 The Selection, Assessment, Classification Framework

Heterogeneous platforms introduce architectural variance that can destabilize predictive pipelines. Alternatively, the structural uniformity of homogeneous platforms permits standard linear regression to map workloads directly to performance indicators. The progression from WORKLOAD PARAMETERS (W) to INSTRUCTIONS (I) and ultimately to EXECUTION TIME (E) is predictable in such isotropic platforms. To decouple workload characteristics from hardware constraints, the *Selection, Assessment, Classification* (SAC) framework formalizes profiling as a multi-stage composite mapping parameterized by the target hardware substrate h:

$$G = (C_h \circ A_h \circ S_h)(W) = C_h\Big(A_h\big(S_h(W)\big)\Big)$$

The pipeline deconstructs execution modeling into three parameterized functional layers, each described below.

3.1.1 Stage 1: Selection (S_h)

Selection ($W \xrightarrow{S_h} I_h$): Maps high-level Workloads (W) to Instruction Sequences (I_h) under hardware parameter h.

$$W = \{w_0, w_1, \ldots, w_n\} \tag{3.1}$$

$$I_h = \{I_{w_0,h}, I_{w_1,h}, \ldots, I_{w_n,h}\} \tag{3.2}$$

$$W \xrightarrow{S_h} I_h \tag{3.3}$$

This isolates how software inputs exercise microarchitectural tiers prior to metric collection.

The Selection operator ($\mathcal{S}_h$) functions as a substrate-parameterized mapping ($\mathcal{S}_h : W \to I_h$) that transforms high-level application workload parameters (W), such as GEMM matrix dimensions, into an intermediate instruction footprint (I_h) under specific platform and thread concurrency constraints. Rather than generating a multi-gigabyte spatial-temporal trace that would impose intractable $O(T)$ memory and tracking overheads on real-time operating system schedulers, instead I_h represents a compact microarchitectural footprint that decouples software algorithmic intent from runtime execution latency. For instance, on simple, in-order Cortex-A55 Little cores where pipeline behavior is deterministic, I_h collapses to a 1-dimensional scalar instruction count I. Conversely, on speculative, out-of-order (OoO) Cortex-A76 Big cores where dynamic loop unrolling and execution stalls decouple instructions retired from clock cycles, I_h expands into a multidimensional microarchitectural event matrix (M_h) that can be comprised of Branch Misses (BM), L1 Loads (L1L), LLC Misses (LLCM), CPU Utilization (CPU), Context Switches (CS), CPU Migrations (CM), Page Faults (PF), and Instructions Per Cycle (IPC). This feature footprint characterizes how software parameters exercise the processor's compilation, fetch, and decode tiers before the Assessment operator ($\mathcal{A}_h$) evaluates physical runtime latency ($M_h \xrightarrow{\mathcal{A}_h} E$).

3.1.2 Stage 2: Assessment (A_h)

Assessment ($I_h \xrightarrow{A_h} E_h$): Maps instruction sequences (I_h) to quantitative performance metrics (E_h), such as execution time, for hardware substrate h.

$$I_h = \{I_{w_0,h}, I_{w_1,h}, \ldots, I_{w_n,h}\} \tag{3.4}$$

$$E_h = \{E_{w_0,h}, E_{w_1,h}, \ldots, E_{w_n,h}\} \tag{3.5}$$

$$I_h \xrightarrow{A_h} E_h \tag{3.6}$$

In the Assessment stage ($\mathcal{A}_h$), while parallel execution time (E) serves as the primary latency metric, several alternative quantitative performance metrics are equally appropriate

depending on system optimization objectives. Key alternatives include *total energy consumption* and the Energy-Delay Product (EDP), which quantify energy dissipation and the fundamental trade-off between execution latency and power overhead on energy-constrained edge platforms. Furthermore, microarchitectural efficiency metrics such as IPC, L1/LLC cache miss rates, branch misprediction rates, and system software overheads (such as context switches, CPU migrations, and page faults) may serve as direct indicators of pipeline health. In the case of OoO processors with their nuanced functionality, these alternative metrics enable the Assessment operator ($\mathcal{A}_h$) to map the intermediate microarchitectural state vector (M_h) to energy, power, or hardware throughput profiles before the Classification operator ($\mathcal{C}_h$) partitions the profile into discrete, actionable operational classes like "Compute Bound" or "Memory Bound".

3.1.3 Stage 3: Classification (C_h)

Classification ($E_h \xrightarrow{C_h} G_h$): Maps quantitative performance metrics (E_h) into discrete qualitative performance categories (G_h), such as "Memory Bound" or "Compute Bound", with the co-domain (G_h) typically partitioned by the necessary performance requirements.

$$E_h = \{E_{w_0,h}, E_{w_1,h}, \ldots, E_{w_n,h}\} \tag{3.7}$$

$$G_h = \{G_{w_0,h}, G_{w_1,h}, \ldots, G_{w_n,h}\} \tag{3.8}$$

$$E_h \xrightarrow{C_h} G_h \tag{3.9}$$

In the Classification stage ($\mathcal{C}_h$), the primary objective is to map continuous, high-dimensional quantitative metric profiles (E_h) into discrete, actionable qualitative performance regimes (G_h). By partitioning the continuous execution co-domain into bounded operational classes—such as *Compute Bound*, *Memory Bound*, *Latency Critical*, or *Optimal Efficiency*—the Classification operator collapses an otherwise intractable continuous search space into a finite decision matrix. This qualitative discretization is essential for real-time operating system governors, thread placement policies, and asymmetric multiprocessor (AMP) runtime schedulers, which require rapid, categorical decision boundaries to execute dynamic core steering and core-affinity allocation. Furthermore, within the inverse mapping pipeline ($\mathcal{C}_h^{-1}$), this

categorical partitioning establishes performance equivalence classes, enabling systems engineers to solve for target workload preimage sets (W) that satisfy strict operational constraints or service-level agreements (SLAs) across heterogeneous edge substrates.

3.1.4 Reverse Mapping and Preimage Optimization

Because the parameterized forward operators S_h, A_h, and C_h are non-injective (many-to-one), they induce mathematical equivalence classes across platform configurations. The formal parameterized inverse mapping resolves target performance goals back to input workload constraints:

$$(S_h^{-1} \circ A_h^{-1} \circ C_h^{-1})(G) = S_h^{-1}\Big(A_h^{-1}\big(C_h^{-1}(G)\big)\Big)$$

Evaluating this inverse mapping identifies the *Preimage Set* of workloads for hardware substrate h:

1. $C_h^{-1}(G) \to \{E_h\}$: Isolates the quantitative performance boundaries corresponding to target category G.
2. $A_h^{-1}(\{E_h\}) \to \{I_h\}$: Identifies the set of instruction sequences that generate that performance range on hardware h.
3. $S_h^{-1}(\{I_h\}) \to \{W\}$: Yields the boundary conditions of input workloads that trigger those instruction profiles.

We can highlight two insights regarding these reverse mappings:

> Systems Engineering: Evaluating the *preimage set* $(\mathcal{S}_h^{-1} \circ \mathcal{A}_h^{-1} \circ \mathcal{C}_h^{-1})(G)$ transforms performance modeling from a passive, forward-predictive tool into an active, prescriptive optimization framework. In real-world deployment across strict-constraint domains—such as autonomous vehicle control loops, edge robotics, and industrial IoT—operating system governors and Asymmetric Multiprocessor (AMP) schedulers utilize inverse mapping to execute closed-form *Capacity Bounding* and dynamic core

allocation. Rather than executing thousands of empirical benchmarking runs to locate safe operational envelopes, a runtime governor evaluates $\mathcal{C}_h^{-1}(G)$ to map strict Service-Level Agreement (SLA) latency bounds ($E_{\max}$) directly back to the maximum allowable workload dimension ($W_{\max} \in \mathcal{W}_{\text{pre}}$) for a target execution substrate h (e.g., partitioning threads between Cortex-A55 Little cores and Cortex-A76 Big cores).

PERFORMANCE ARTICULATION: Preimage optimization exposes the fundamental mathematical asymmetry of speculative out-of-order architectures. While forward composite prediction ($W \to I \to E$) suffers from catastrophic Stage 1 error propagation ($W \to I$) due to microarchitectural volatility, inverse mapping treats metrics (e.g. observed execution latency (E)) as a global constraint that tightly bounds the intermediate state space, enabling near-deterministic reconstruction of original workload selection parameters. Consequently, preimage optimization may collapse large parameter combinatorial search spaces of heterogeneous SoCs into solvable algebraic fibers, allowing hardware co-designers to quantify mathematically how microarchitectural modifications shift the physical boundaries of allowable input workloads without relying on exhaustive cycle-accurate simulation.

3.1.5 Architectural Consistency and Reduced Latent Variance

In a homogeneous system, every core shares an identical pipeline depth, branch predictor logic, and cache hierarchy. With uniform hardware behavior, the relationship between a workload (W) and its resulting microarchitectural events (Stage 1: $W \to I$) is highly stable. In homogeneous environments, a single scalar like INSTRUCTIONS (I) often serves as a sufficient proxy for execution. In this example, the instruction parameter realizes the microarchitectural footprint (M) as a one-dimensional vector.

3.1.6 Industry Application: Predictability vs. Complexity

In industrial settings, these mappings are utilized for three primary functions, each of which favors the predictability of homogeneous devices:

Dynamic Voltage and Frequency Scaling (DVFS): Industry uses forward mappings ($I \rightarrow E$) to predict the performance impact of frequency shifts. In homogeneous systems, a simple linear regression is often sufficient to maintain a high coefficient of determination. This allows for rapid, low-overhead power management decisions. Performance-monitoring counters can be used to construct low-overhead linear models of processor and memory power. Contreras and Martonosi demonstrate such a model for the Intel PXA255/XScale processor, with power weights calibrated offline for each voltage/frequency configuration and average estimated CPU power within approximately 4% of measured power [22]. Such models provide the runtime power estimates needed by power-aware DVFS mechanisms.

Capacity Planning and Scheduling: Operating system schedulers use these mappings to estimate the "cost" of a thread. On homogeneous hardware, the scheduler can assume that 1 billion instructions will take X time across any core. In heterogeneous systems, we postulate that a "wealth of platform indicators" is necessary to reach Stage 2 isolation. The overhead of tracking multiple parameters such as Branch Misses (BM), L1 dcache Loads (L1L), or LLC Load Misses (LLCM) in real-time is often too high for standard industrial schedulers, making the simpler, more stable homogeneous mapping preferable. Yet, heterogeneous systems complicate scheduling because core selection depends on both workload characteristics and underlying microarchitectural capabilities. HASS addresses this problem using compact architectural signatures of application behavior to guide thread-to-core assignment, avoiding the complexity, load imbalance, and performance overhead associated with continuous online profiling [108].

Resource Accounting (Inverse Mapping): Industry uses inverse models ($E \rightarrow M \rightarrow W$) for "billing" or identifying which workload caused a specific system state. These inverse mappings are highly deterministic for simple cores. This high fidelity allows companies to reconstruct workload parameters accurately from observed execution time and basic hardware counters. Hardware performance-monitoring events can be leveraged for thread-level energy accounting. Bellosa demonstrates an event-driven approach in which activity metrics from hardware units (such as integer/floating-point units and cache/memory interfaces) are correlated with energy consumption to construct thread-specific energy usage profiles [9].

3.2 Homogeneous Performance Insights

To understand each platform's performance accurately, its unique architectural perspective and its variability can be expressed via the following ideals:

Architectural Determinism: The Selection, Assessment, and Classification (SAC) framework defines the structural invariance of the Assessment transfer function ($\mathcal{A}_h : I_h \rightarrow E_h$), representing the degree to which a hardware substrate operates as a state-invariant, isotropic execution engine. On in-order Little cores, execution lacks speculative windowing, dynamic instruction reordering, and out-of-order dispatch; consequently, the Cycles Per Instruction (CPI) remains stable across scaling workload dimensions (W). This structural uniformity maintains high Architectural Determinism, enabling a single-parameter scalar instruction count (I) to map deterministically to execution time (E). Conversely, on speculative, out-of-order Big cores, Architectural Determinism collapses because dynamic loop unrolling, branch entropy, out-of-order pipeline dispatch, and memory hierarchy stalls decouple instructions retired from elapsed execution cycles, causing scalar predictive models to fail completely.

Structural Heteroscedasticity: The SAC context denotes the systematic, non-constant scaling of prediction error variance ($\text{Var}\ (\epsilon \mid W) \neq \sigma^2$) across the sequential operator pipeline ($\mathcal{S}_h \rightarrow \mathcal{A}_h \rightarrow \mathcal{C}_h$) as a function of workload dimension (W) or microarchitectural complexity. On in-order Little cores, low structural complexity keeps error variance bounded, preserving near-homoscedastic behavior. However, as workloads scale past L1/L2 cache boundaries into Last-Level Cache (LLC) and DRAM regimes on out-of-order Big cores, the error variance of intermediate performance counters in Stage 1 ($\mathcal{S}_h : W \rightarrow M_h$) explodes non-linearly, producing extreme error volatility across indicators. In multi-stage Ordinary Least Squares (OLS) regressions, this heteroscedastic noise propagates directly into the Stage 2 Assessment model ($\mathcal{A}_h : M_h \rightarrow E$), inducing classical Errors-in-Variables (EIV) attenuation bias that flattens structural coefficients. The SAC methodology mitigates Structural Heteroscedasticity by transitioning to Structural Equation Modeling (SEM), which isolates scale-dependent measurement error variances onto the diagonal matrix ($\mathbf{\Theta}_\epsilon$) and evaluates structural paths over the full empirical joint covariance matrix ($\mathbf{S}$).

Therefore, we posit that successful microarchitectural mapping in the mixed processor platforms will be a function of *Architectural Determinism* and the reduction of *Structural Heteroscedasticity*. In symmetric systems, where the execution units, pipeline depths, and branch predictor topologies are identical, the mapping of workload to microarchitectural footprint and thereafter to performance indicators ($W \rightarrow I \rightarrow E$) operates within a stable covariance structure that asymmetric architectures inherently violate. However, we also anticipate that even in AMP architectures, that while the mapping of workload to microarchitectural footprint and thereafter to performance indicators ($W \rightarrow M \rightarrow E$) may be poor, if the intermediary mapping ($M \rightarrow E$) has a higher predictive value (R^2) this may hint that the microarcitectural footprint (M) may functions as a latent mediator which supports the adoption of SEM techniques.

3.2.1 Stability of the Parameter Vector (β)

In homogeneous systems, typically comprised of symmetric in-order processors, the relationship between a workload domain (W) and its resulting microarchitectural footprint (M) is nearly deterministic because the "architectural tax" (pipeline stalls, branch entropy, and cache hierarchy latency) remains constant across all execution units. Depending on core microarchitecture and model specification, the legal constituents of the microarchitectural footprint vector M range from a scalar instruction count ($M = [I]$) on in-order Little cores to high-dimensional feature vectors ($M = [I, BM, L1M, LLCM, ...]^T$) on speculative Big cores.

Mathematically, this deterministic behavior implies that the coefficient vector β in the Stage 1 Selection mapping ($M = W\beta + \epsilon$) remains invariant regardless of core assignment, where W represents the design matrix of workload scaling parameters and platform covariates.

Reduced Variance (ϵ): Because the physical execution channels are identical across cores, the residual vector ϵ captures only transient system noise rather than the structural variance introduced by ISA execution asymmetry or divergent IPC ceilings.

Well-Conditioned Information Matrix: The information matrix W^TW is significantly better conditioned in symmetric systems. This stability prevents the "Compound

Blowup" wherein minor prediction errors in the intermediate counter matrix M propagate non-linearly into divergent execution time predictions ($\hat{E}$) in further stage assessments.

3.2.2 Elimination of the Mediation Gap

In a homogeneous world, given the uniformity of the platforms and the isotropic performance characteristics of the execution units, the mediation is trivialized:

Linear Scaling: In symmetric hardware, the mapping from instructions (I) to execution time (E) is often linear, as the Cycles Per Instruction (CPI) remains relatively stable for a given workload across the entire core island.

Isomorphic State Transitions: The transition from a workload's algorithmic intensity to its microarchitectural pressure (Stage 1) follows the same physical laws on every core. Consequently, a scalar proxy like instruction count often captures the vast majority of the variance, eliminating the need for a high-dimensional "wealth of indicators" to achieve predictive stability.

Yet, we cross into unpredictable territory when investigating these mappings on heterogeneous platforms. We argue that the adoption of SEM for modeling the asymmetric systems is necessary given that we assume that M functions as a latent mediator which differs between various core types with their associated data buses and memory hierarchies, requiring additional metrics to strengthen the mapping.

3.2.3 Industrial Application: Deterministic Scheduling and DVFS

Industry relies on these homogeneous mappings because they allow for *Look-ahead Determinism* in ways heterogeneous systems cannot:

Deterministic Scheduling: OS schedulers in homogeneous environments use simple mappings to assume that a thread's cost is core-invariant. This allows for $\mathcal{O}(1)$ scheduling decisions because the mapping $W \to E$ is a reliable predictor across the entire resource pool in the case of in-order processors, but requires additional parameters for OoO processors.

HIGH-FIDELITY DVFS: In homogeneous systems (in-order processors), power management algorithms (Dynamic Voltage and Frequency Scaling) can use simple linear models to predict performance degradation. Because the microarchitectural response to frequency scaling is uniform across the fabric, the mapping $I \to E$ does not suffer from the "Stage 1 Error Explosion" that characterizes out-of-order, asymmetric Performance cores.

3.2.4 Efficient Inverse Mapping

The inverse mapping ($E \to M \to W$) is particularly successful in homogeneous worlds because it avoids the *Aliasing Problem*. In asymmetric systems, the same execution time (E) could be produced by a large workload on a high-performance ("Big") core or a small workload on a highly-efficient ("Little") core, creating a bimodal distribution that complicates reconstruction. In a homogeneous system, each execution signature (obtained from $E\&M$) maps to a unique workload state (W), making the reconstruction of workload parameters near-perfect without the need for complex structural regularization.

In summary, homogeneous mappings succeed because they operate on a single, stable manifold. We argue that the complexity introduced by heterogeneous platforms requires high-dimensional matrices and SEM which is the direct result of architectural divergence, which symmetric systems mathematically circumvent by design.

3.2.5 Technical Insight

In computer engineering, this pipeline is frequently referred to as *Hierarchical Performance Attribution* [1, 90, 115, 118, 131] or *Multi-Stage Workload Characterization* [35, 59, 93, 109]. Because C, A, and S are typically non-injective (many-to-one), the reverse mapping ($S^{-1} \circ A^{-1} \circ C^{-1}$) results in a *Preimage Set* rather than a unique point, representing the *Equivalence Class* of workloads for a specific performance outcome. The Selection, Assessment, Classification (SAC) methodology, as formalized in this research, provides a mathematical pipeline for Hierarchical Performance Attribution. By deconstructing the design space into three discrete mappings—S, A, and C—we can transition from brute-force empirical observation to a structured, articulative modeling of heterogeneous performance.

3.3 Heterogeneous Performance Insights

3.3.1 Performance Model Portability

To better understand the behavior of the SAC methodology in the heterogeneous co-design space, we evaluate the two distinct ARM platforms to ensure that the results are not specific to a single SoC generation. Our use of the term "platform" necessarily includes both the processor IC as well as additional components of the computer that the IC is a part of. We will use these two platforms to draw tentative conclusions about the applicability of SAC to these AMP platforms.

3.3.2 Configuration Scenarios

SCENARIOS	
PROCESSING UNITS	8 PROCESSORS
THREADS	16 THREADS
ISLANDS	3 REGIONS
WORKLOAD PARAMETERS	128 OPTIONS

Figure 3.1: Potential Scenarios

For ARM platforms, an example of possible configuration scenarios is shown in Figure 3.1. Each configuration comprises some number of PROCESSING UNITS, THREADS, ISLAND SELECTION, and WORKLOAD PARAMETERS which in this case is the MATRIX DIMENSION. Each of these configurations leads to a particular performance profile which is representative of EXECUTION TIME or other performance indicators. Given a host of performance profiles, we partition these profiles into qualitative groups (e.g. Good, Bad, Better, etc) for given performance criteria (e.g. shortest execution time, lowest energy delay product, etc). The potential scenarios and performance profiles are shown by their connective dependencies graphically in Figure 3.2.

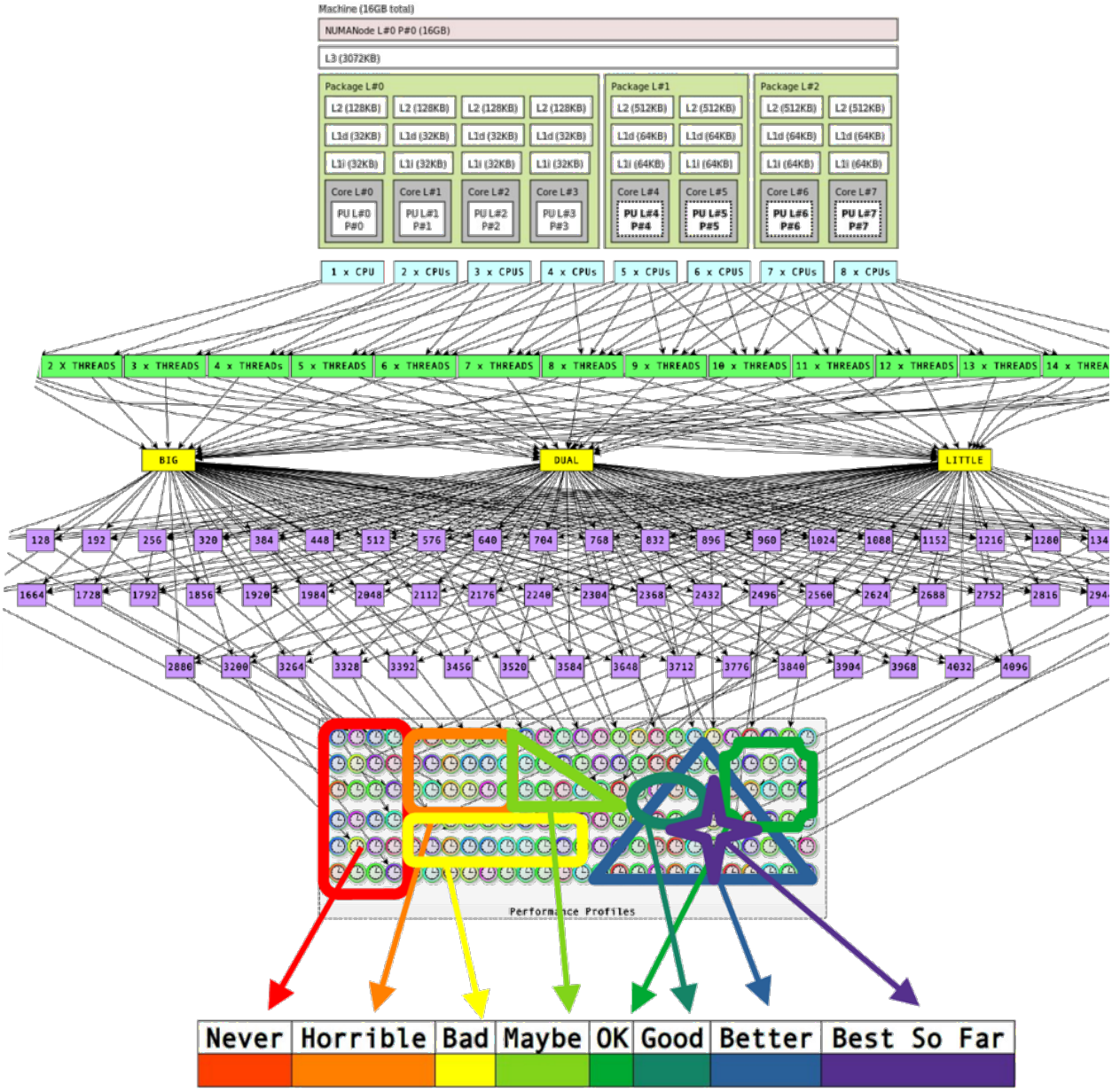


Figure 3.2: Scenario Connections

3.3.3 Problem Formulation: The Brute-Force Trap

Traditional performance modeling in heterogeneous environments scales exponentially. For the potential scenarios referenced in Figure 3.1 we can calculate the number of scenarios directly. The comprehensive list of configuration scenarios is shown in Table 3.1 with each row in the table containing:

Core Counts: A number of cores (k) in the range $k \in \{1, ..., 8\}$

Core Selections: Combinations of selecting k cores out of 8, calculated as $\binom{8}{k}$

Thread Counts: Choices in this range is $(2k - k) + 1 = k + 1$

Thread Selections: Total number of thread assignment options.

Workload Parameters: Constant factor of 128 (matrix dimension)

Scenario Count: Total number of unique configurations for k cores.

Consequently, the product formula for each row k (where $k \in \{1, 2, \ldots, 8\}$) is:

$$\text{Scenario Count}_k = \binom{8}{k} \times (k + 1) \times 128$$

To test every configuration exhaustively will require 163,712 distinct configuration scenarios, per platform, derived from the combinatorial expansion of:

Core Allocations & Affinities: DynamIQ Big.LITTLE topology mapping

Thread Concurrency: Parallel scaling and execution behaviors

Workload Parameters: Multidimensional parameter variations

We theorize that across heterogeneous execution environments, directly mapping empirical performance from host h_1 to h_2 in many cases will fail due to hardware coupling. Because raw execution metrics are bound to the underlying substrate (h_i) for for $i \in \{1, \ldots, N\}$, cross-target projection breaks down. The one exception may be where two platforms have

Table 3.1: Scenarios by Configuration

Core Count	Core Selections	Thread Count	Thread Selections	Workload Parameters	Scenario Count
1 CPU	8	1-2	2	128	2,048
2 CPUs	28	2-4	3	128	10,752
3 CPUs	56	3-6	4	128	28,672
4 CPUs	70	4-8	5	128	44,800
5 CPUs	56	5-10	6	128	43,008
6 CPUs	28	6-12	7	128	25,088
7 CPUs	8	7-14	8	128	8,192
8 CPUs	1	8-16	9	128	1,152
Total Scenarios					**163,712**

similar silicon components, such as the in-order little cores common to both ARM platforms. Furthermore, the loss of performance portability destabilizes the search space by concealing non-linear performance cliffs hidden behind architectural artifacts and rendering systematic space exploration intractable.

3.4 Experimental Setup

The experiments which are to follow as designed to evaluate the predictive fidelity of multi-stage performance models within a heterogeneous system-on-chip (SoC) environment, specifically distinguishing between high-performance "Big" core islands and energy-efficient "Little" core islands. We performed several thousand of the scenarios for various matrix dimensions (ranging from 64×64 to 2048×2048). The time requirements to probe the full range of matrix dimensions was a limiting factor so we restricted that to a meaningful range capturing the full occupancy of the L1, L2, and L3 caches across the compute units. The underlying dataset, built from from `perf` data, was obtained by executing the workloads on both the Orange Pi 5 and the Cubie A7A. We include the performance metrics of both platforms into one dataset which comprises the range of parameters that we will map to execution times, which defines the co-domain. To ensure the mathematical validity of the subsequent log-log transformations, the data was filtered to include only finite, positive values for workload dimension (W), execution time (E), and microarchitectural counters (M). The experimental matrix was partitioned using an 80/20 train-test split to evaluate model generalization. The methodology employs a two-stage ordinary least squares (OLS) regression framework, utilizing log-scale transformations (e.g., $\ln(W) \to \ln(I)$) to capture the power-law relationships inherent in computational scaling.

3.4.1 Feature Vector Construction and Dimensionality

The experiments evaluate three distinct microarchitectural feature sets:

Scalar Baseline (Little Cores): Maps workload (W) to a single (M) proxy variable: instruction count (I).

Initial Matrix (Big Cores): Expands the feature vector (M) to include four critical counters: Instructions (I), Branch Misses (BM), L1 dcache load misses ($L1M$), and LLC load misses ($LLCM$).

High-Dimensional Matrix (Expanded Big Cores): Utilizes a 9-parameter (M) vector incorporating deep platform indicators: Instructions (I), Branch Misses (BM), L1 Loads ($L1L$), LLC Misses ($LLCM$), CPU Utilization (CPU), Context Switches (CS), CPU Migrations (CM), Page Faults (PF), and IPC.

3.4.2 Forward Predictive Mapping Pipeline

We make adjustments to the forward model to account not only for the platform (h_i), but also the exact compute unit which we define as the *cpuid* $j \in \{1, \ldots, k\}$. The forward model, denoted as $W_{h_{i,j}} \to M \to E$, is decomposed into two structural stages:

Stage 1 (Systemic Response): Fits the model $W_{h_{i,j}} \to M$, where h represents the *platform* and *cpuid* j accounts for core-specific variance. This stage predicts the microarchitectural event matrix M based on the input workload W and the specific hardware context.

Stage 2 (Architectural Mapping): Fits $M \to E$, articulating the relationship between the hardware event matrix and the final parallel execution time.

To eliminate model divergence during composite execution, the pipeline implements Physical Boundary Clipping. Intermediate predictions for the counter matrix M are bounded within the absolute minimum and maximum values observed in the training spectrum ($M_{\min}, M_{\max}$). This prevents "Compound Blowup," where Stage 1 estimation errors propagate non-linearly into the Stage 2 execution time prediction.

3.4.3 Inverse Diagnostic Mapping Pipeline

The experiment further validates the structural relationship through an inverse pipeline, $E \rightarrow M \rightarrow W$, designed to reconstruct workload parameters from observed execution signatures:

INVERSE STAGE 1: Maps execution time (E) to the microarchitectural matrix M.

INVERSE STAGE 2: Maps the predicted (and bounded) matrix M back to the original workload dimension W.

3.4.4 Validation Metrics and Diagnostic Protocol

The performance of these mappings is assessed through a dual-validation protocol:

ISOLATED METRICS: Evaluates each stage using ground-truth inputs to determine the theoretical upper bound of the model's fit (R^2) and error (RMSE, MAPE).

COMPOSITE METRICS: Evaluates the end-to-end pipeline using predicted intermediate values, providing a realistic assessment of predictive power in production environments where microarchitectural events are unknown variables.

The interpretation of the isolated metrics is as follows:

R^2 (COEFFICIENT OF DETERMINATION): Measures the proportion of variance in the dependent variable that is predictable from the independent variables. It evaluates how well a model captures the underlying data pattern compared to a baseline model that simply predicts the mean.

RANGE: $(-\infty, 1]$ (frequently expressed as $[0, 1]$ or $[0\%, 100\%]$ for linear models fit with an intercept)

INTERPRETATION: 1 (or 100%) indicates that the model perfectly predicts the variance in the target variable. A value of 0 means the model performs no better than predicting the mean value. Negative values occur when the model performs worse than a horizontal mean line, indicating an improperly fit non-linear or unconstrained model.

RMSE (Root Mean Square Error): Measures the average magnitude of prediction errors by taking the square root of the mean squared differences between predicted and actual values. Because errors are squared before averaging, RMSE penalizes larger errors more heavily, expressing results in the same units as the target variable.

Range: $[0, \infty)$

Interpretation: 0 indicates a perfect fit. Higher values signify larger overall prediction errors, with extreme outliers pulling the score significantly upward.

MAPE (Mean Absolute Percentage Error): Measures prediction accuracy by expressing errors as a percentage of the actual observed values. It calculates the average absolute difference between predicted and actual values divided by the actual value.

Range: $[0, \infty)$ (or $[0\%, \infty)$)

Interpretation: 0% indicates perfect accuracy. Lower percentage values represent better relative predictive precision. The metric becomes undefined if an actual value is 0 and produces skewed results when actual values are close to 0.

This setup allows for the direct comparison of R^2 values—such as the 0.7281 composite forward R^2 versus the 0.9468 composite inverse R^2—to identify the structural bottlenecks in asymmetric performance modeling.

3.5 Experimental Results

This section serves to analyse the empirical performance data, motivating the use of Structural Equation Modeling (SEM) in the characterization of asymmetric computational performance (which is presented in the following chapters). The following tables will summarize the core empirical findings, specifically contrasting the behavior of the scalar baseline (Little cores) against the high-dimensional microarchitectural matrix models (Big cores). These results establish the "Composite Predictive Ceiling" and the "Inverse Accuracy Gap" that inform the move toward SEM.

3.5.1 Analysis of Scalar Mapping (The Baseline)

The result of the one-parameter mappings for both heterogenous platforms is given below with Table 3.2 and Table 3.3 showing the mappings for big and little cores, respectively. The contrast between these two tables provides a mathematically elegant empirical demonstration of the "Asymmetry Tax" imposed by complex microarchitectures when using scalar proxies.

Table 3.2: 1-Parameter Little Core Mapping

Metric	**Stage 1** ($W \to I$)	**Stage 2** ($I \to E$)	**Composite** ($W \to E$)
Fit (R^2)	0.9773	0.9776	0.7281
Test MAPE	48.24%	37.09%	32.14%

Little Core Mapping: For the simple, in-order Cortex-A55 Little cores, instruction count (I) serves as a highly deterministic proxy for execution time (E) because the pipeline lacks speculative out-of-order scheduling. This structural uniformity is reflected in the near-isomorphic isolated fits, where the Stage 1 ($W \to I$) mapping yields $R^2 = 0.9773$ and the isolated Stage 2 ($I \to E$) mapping achieves $R^2 = 0.9776$. Even with a modest Stage 1 Test MAPE of 48.24%, the end-to-end Composite mapping ($W \to E$) remains highly stable, achieving a fit of $R^2 = 0.7281$ and a Test MAPE of 32.14%. Because the Cycles Per Instruction (CPI) of an in-order pipeline remains relatively constant across scaling matrix dimensions, the underlying covariance of the system is easily captured by simple, single-parameter ordinary least squares (OLS) regression.

Table 3.3: 1-Parameter Big Core Mapping

Metric	**Stage 1** ($W \to I$)	**Stage 2** ($I \to E$)	**Composite** ($W \to E$)
Fit (R^2)	0.5355	0.3763	0.0365
Test MAPE	5860%	1083%	727%

Big Core Mapping: Conversely, on the speculative, out-of-order Cortex-A76 Big cores, this scalar paradigm collapses catastrophically. The Stage 1 ($W \to I$) mapping is severely degraded by dynamic compiler optimizations and variable loop scheduling, causing the Test MAPE to explode to 5860%. Simultaneously, the isolated Stage 2 ($I \to E$) fit plummets to $R^2 = 0.3763$ and a MAPE of 1083% because a single scalar indicator cannot capture the non-linear pipeline stalls, branch mispredictions, and cache capacity misses that dominate speculative execution. When these volatile Stage 1 intermediate predictions are propagated

into Stage 2, the Errors-in-Variables (EIV) propagation flatlines the end-to-end predictive power, yielding a virtually useless Composite R^2 of 0.0365 and an end-to-end Test MAPE of 727%.

This failure in the mapping of the Big cores provides empirical support showing that scalar performance modeling is fundamentally invalid on speculative hardware, necessitating a multidimensional covariance-based framework like Structural Equation Modeling (SEM) to isolate the latent "Architectural Intensity" from this high-amplitude pipeline noise.

We now move on to additional paramters to map the Big cores onto the associated metrics to see if our models improve with additional performance metric information.

3.5.2 Dimensional Feature Expansion

To analyze multi-stage performance mappings on complex, out-of-order architectures, we must first establish a mathematically consistent method for evaluating intermediate predictive stages. In the 1-parameter Little core model, Stage 1 is a straightforward scalar-to-scalar mapping ($\ln(W) \to \ln(I)$), meaning its fit and error metrics are uniquely defined. However, when moving to the multidimensional Big core configurations, Stage 1 is an over-determined system where a single scalar predictor—workload size (W)—is mapped to a vector of independent, highly volatile microarchitectural counters.

Presenting individual Stage 1 diagnostics for each of the estimated counters would expand the dimensional complexity of our diagnostic tables, rendering direct symmetric comparison with our baseline models impossible. To resolve this, we formulate the *vector-mean representation* (M_{avg}) by calculating the arithmetic mean of the individual coefficient of determination (R^2) and the Mean Absolute Percentage Error (MAPE) across all performance estimators in the stage:

$$\bar{R}^2_{\text{S1}} = \frac{1}{p}\sum_{j=1}^{p} R^2_{\text{S1},j}, \quad \bar{\text{MAPE}}_{\text{S1}} = \frac{1}{p}\sum_{j=1}^{p} \text{MAPE}_{\text{S1},j}$$

where p represents the count of microarchitectural features in the mapping matrix. This average acts as a composite index of the microarchitectural transfer function's overall modeling

complexity, compressing the high-dimensional indicator space into a single metric. This allows us to maintain a consistent 3-column structural layout while directly exposing the average prediction noise propagating into the final execution stage.

Table 3.4: 4-Parameter Big Core Mapping (Vector-Mean Stage 1 Representation)

Metric	**Stage 1** ($W \to M_{avg}$)	**Stage 2** ($M \to E$)	**Composite** ($W \to E$)
Fit (R^2)	0.7001	0.9524	0.7299
Test MAPE	3297%	73%	45%

4-Parameter Big Core Mapping: The empirical results for the *4-parameter microarchitectural matrix* reveal a severe decoupling between isolated and composite predictive capabilities on the high-performance Big core island. Under isolated conditions where ground-truth microarchitectural states are known, the Stage 2 model ($M \to E$) achieves a high training fit of $R^2 = 0.9524$. This confirms that incorporating essential pipeline and memory stalls ($I, BM, L1M, LLCM$) successfully explains the physical variance in execution time.

However, when assembling the end-to-end forward predictive pipeline ($W \to M \to E$), the Composite Forward fit is capped at $R^2 = 0.7299$ with a Test MAPE of 45%. This performance ceiling is the direct mathematical consequence of the Stage 1 Error Propagation represented by the vector-mean Stage 1 Test MAPE of 3297%. This massive average error is heavily weighted by the catastrophic prediction spikes of individual out-of-order execution counters, such as a 5860% Test MAPE for instructions and a 6952% Test MAPE for L1 dcache load misses.

When these highly distorted predicted inputs are fed into the Stage 2 ordinary least squares (OLS) model, the classical *Errors-in-Variables (EIV)* effect induces severe attenuation bias in the structural coefficients. This limits the composite R^2 to the 0.73 barrier, regardless of the high theoretical accuracy of the isolated stage.

Table 3.5: 9-Parameter Big Core Mapping (Vector-Mean Stage 1 Representation)

Metric	**Stage 1** ($W \to M_{avg}$)	**Stage 2** ($M \to E$)	**Composite** ($W \to E$)
Fit (R^2)	0.5090	0.9293	0.7319
Test MAPE	1546%	45%	37%

9-Parameter Big Core Mapping: The results for this configuration further confirm this structural ceiling and highlight the limits of high-dimensional OLS under high input volatility. By expanding the microarchitectural feature set to nine parameters, we incorporate deeper platform indicators—such as page faults, context switches, CPU migrations, and IPC—to capture the complex pipeline dynamics of the Cortex-A76 cores. This high-dimensional matrix successfully stabilizes the Stage 1 vector-mean space, reducing the average predictive error by more than half, from 3297% down to 1546%.

Yet, despite this substantial improvement in Stage 1 input fidelity and a highly precise isolated Stage 2 fit of $R^2 = 0.9293$, the Composite R^2 remains completely flat at 0.7319, with a persistent end-to-end Test MAPE of 37%. This stagnation reveals the fundamental limitation of multi-stage OLS modeling on heterogeneous hardware. Even a dramatic reduction in Stage 1 prediction error cannot break the composite performance barrier because OLS cannot model the joint covariance structure of the indicators.

In out-of-order execution, counters such as IPC, instruction count, and L1 loads are physically coupled by pipeline back-pressure and resource contention. By forcing these variables into separate, independent Stage 1 regressions, OLS discards this covariance, treating them as orthogonal.

This persistent composite bottleneck of ~ 0.73 R^2 across both 4-parameter and 9-parameter configurations is the needed justification for transitioning to **Structural Equation Modeling (SEM)**. SEM handles these interconnected variables as reflective indicators of a single latent construct, optimizing the entire structural path simultaneously to bypass the localized Stage 1 "Compound Blowup".

Table 3.6: 9-Parameter Inverse Big Core Mapping (Vector-Mean Stage 1 Representation)

Metric	**Stage 1** ($E \to M_{avg}$)	**Stage 2** ($M \to W$)	**Composite** ($E \to W$)
Fit (R^2)	0.5156	0.9345	0.9468
Test MAPE	1752%	15%	11%

9-Parameter Inverse Big Core Mapping: Inverting the causal path reveals a profound structural asymmetry, detailed in Table 3.6. Reversing the mapping trajectory drives a substantial increase in composite explanatory power, raising the overall fit to $R^2 = 0.9468$ and reducing end-to-end Test MAPE to 11%. This demonstrates that aggregate Execution Time (E) acts as a tightly constrained global proxy for macro-level workload behavior (W).

However, this high composite fidelity obscures massive intermediate instability: Stage 1 ($E \rightarrow \mathbf{M}_{avg}$) suffers an extreme Test MAPE of 1752%, confirming that singular scalar outputs cannot uniquely resolve high-dimensional microarchitectural states when modeled via isolated OLS regressions. This stark dichotomy between forward and reverse configurations provides the definitive empirical justification for Structural Equation Modeling (SEM):

FORWARD DIRECTION ($W \rightarrow E$): Constrained by a composite bottleneck ($R^2 \approx 0.73$) due to OLS discarding joint microarchitectural covariance.

REVERSE DIRECTION ($E \rightarrow W$): Achieves high macro-level trajectory fit ($R^2 = 0.9468$) despite catastrophic intermediate point-wise error propagation (1752% MAPE).

Together, these results prove that hardware performance counters do not function as independent intermediate variables, but rather as correlated reflective indicators of an underlying, unobserved latent execution state. Multi-stage OLS fails because it optimizes each stage in isolation, forcing uncoordinated error propagation across stages. We argue that SEM is required to simultaneously model the measurement and structural equations, preserving joint covariance across $\mathbf{M}$ and eliminating stage-wise error compounding.

3.5.3 Key Observations

The empirical observations detailed below summarize the structural dynamics and predictive trade-offs observed when modeling multi-stage execution paths on speculative Big cores. These findings document the persistent performance ceiling of multi-stage OLS regressions under intermediate feature volatility, demonstrating why independent scalar models fail to capture joint microarchitectural covariance and establishing the mathematical necessity of an integrated structural framework:

HIGH-DIMENSIONAL FEATURE EXPANSION: To capture the complexity of "Big" cores, the feature vector was expanded to include deep microarchitectural counters. Using additional parameters, the matrix M for Stage 2 mapping ($M \rightarrow E$), achieved near-deterministic isolated accuracy. This proves the first assumption: Adding additional parameters articulates execution time more succinctly.

The "Compound Blowup" and Error Propagation: The limitation of standard OLS in this context is the Stage 1 Error Explosion. When we attempt to predict the 9-parameter matrix M from workload W, the volatility of individual counters becomes unmanageable. In a disjointed regression, these errors are fed directly into Stage 2. Even with physical boundary clipping to prevent divergence, the "noisy" inputs from Stage 1 bias the execution time prediction. Standard OLS cannot account for the fact that these intermediate variables are estimates rather than observations, leading to the observed R^2 ceiling of 0.73, despite the increased dimensionality.

The Inverse Mapping Proof (The "Smoking Gun"): The multi-parameter Composite Inverse ($R^2 \geq 0.92$) confirms that execution time and the microarchitectural states effectively "encode" the workload, providing the mathematical justification for using SEM to reconcile these asymmetric results. The inverse mapping models achieve far superior performance compared to the forward paths, thereby revealing that these features (M) serve as a mediating latent construct. Though the system behaves as a deterministic physical process ($W \to M \to E$), the "wealth of platform indicators" in those features are highly sensitive to platform-specific noise (h).

3.5.4 Motivation for Structural Equation Modeling (SEM)

The empirical limitations of the multi-stage Ordinary Least Squares (OLS) paradigm demonstrate that independent scalar regressions cannot adequately model the complex, interdependent physical dynamics of speculative out-of-order microarchitectures. Although high isolated stage fits ($R^2 > 0.95$) confirm that intermediate state vectors contain the physical signal governing execution time, sequential OLS treats predicted intermediate counters as error-free observations. This disjointed formulation discards cross-counter covariance, amplifies Stage 1 prediction noise through Errors-in-Variables propagation, and imposes an artificial ceiling ($R^2 \approx 0.73$) on composite predictive accuracy. Resolving these methodological pathologies requires transitioning to Structural Equation Modeling (SEM), a full-information covariance framework specifically engineered to optimize simultaneous path networks, model latent constructs, and isolate system signal from high-amplitude measurement error through:

Mediated Path Modeling: The results show that W does not determine E directly, but through the state M. SEM allows the simultaneous estimation of these paths, treating M as a mediator rather than a simple independent variable.

Covariance Structures: Counters like IPC, Instructions, and Cache Misses are physically coupled. Disjointed OLS ignores their inter-correlation. SEM models the covariance matrix of M, extracting the shared signal that drives E while filtering out the counter-specific noise.

Measurement Error Mitigation: SEM specifically addresses the "Errors-in-Variables" problem. Since the inputs to Stage 2 are predicted (and thus noisy), SEM's full-information maximum likelihood (FIML) estimation can yield unbiased coefficients that OLS cannot achieve in the presence of 6000% Stage 1 MAPEs.

The experimental data confirms that a high-dimensional mapping exists and is necessary to characterize Big core performance. However, the current multi-stage OLS approach is bottlenecked by error propagation. The inverse mapping $R^2 \geq 0.94$) validates the notion that structural information is present in the microarchitectural matrix. Consequently, to accurately "predict" and "articulate" this behavior, the research must move to an SEM framework that can reconcile the high training-spectrum variance with the deterministic architectural underlying structure.

3.6 Cross-Disciplinary Promise and Future Domains

In an effort to find generalizable methodologies to normalize heterogeneous platforms, the utilization of SAC and SEM holds significant promise for many domains characterized by pluralistic parallelism. The successive chapters, comprised of a selection of peer-reviewed published works, illuminate this hidden performance modeling and show both its efficacy and application to a wide range of domains including:

Autonomous Systems: Real-time object detection and localization (as found in Autonomous Vehicles, Robots, Mobile Sensing), must dynamically offload matrix operations to energy-efficient cores without crossing "scheduling cliffs".

Advanced Instrumentation: Astrophysical Telescopes for computation-assisted localization of phenomena in space and Catoptric Actuation Devices for visual feedback control, where hardware-agnostic parameters must be mapped to specific performance envelopes in high-entropy environments.

Biomedical Engineering: Continuous health monitoring via wearable devices, where signal processing workloads must be balanced against stringent battery-life constraints.

Industrial IoT: Smart Factories requiring matrix computations for predictive maintenance or quality control on edge devices.

To adopt SAC rapidly, a researcher must simply ensure their domain meets two criteria: the availability of diverse heterogeneous compute resources and partitionable (e.g., matrix-based, data-parallel tasks, embarassingly parallel) workloads. We will demonstrate that the tripartite mapping of Selection, Assessment, and Classification provides the high-fidelity telemetry required to navigate modern silicon topography.

3.7 Chapter Conclusion

The SAC methodology distinguishes this work from state-of-the-art (SOTA) predictive black-box models by providing performance articulation. While SOTA models ask questions such as, "What is the execution time?" using the SAC models we can ask, "Why does the workload have that execution time?" By using SEM to bridge the SAC methodology, researchers can turn the "dragons" of non-deterministic latency into a structured, linearly improving knowledge base that maintains performance portability across selections of heterogeneous edge computing platforms. Finally, we show graphically in Figure 3.3 and Figure 3.4 the partitioning of the SAC methodology into digestible chunks that can be used to facilitate rapid research and development of new models, techniques, and insights for heterogeneous platforms.

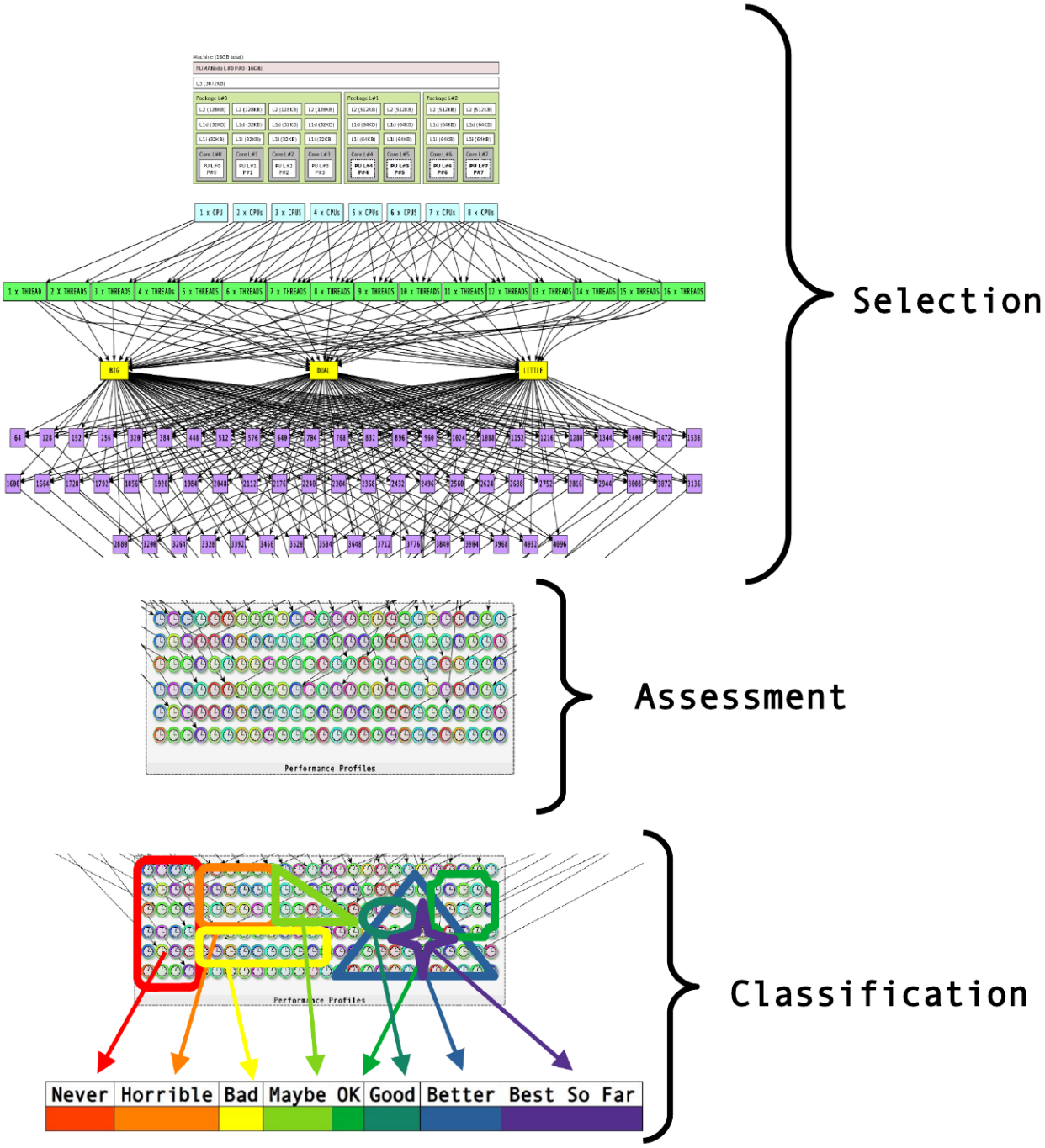


Figure 3.3: Selection, Assessment, Classification Stages

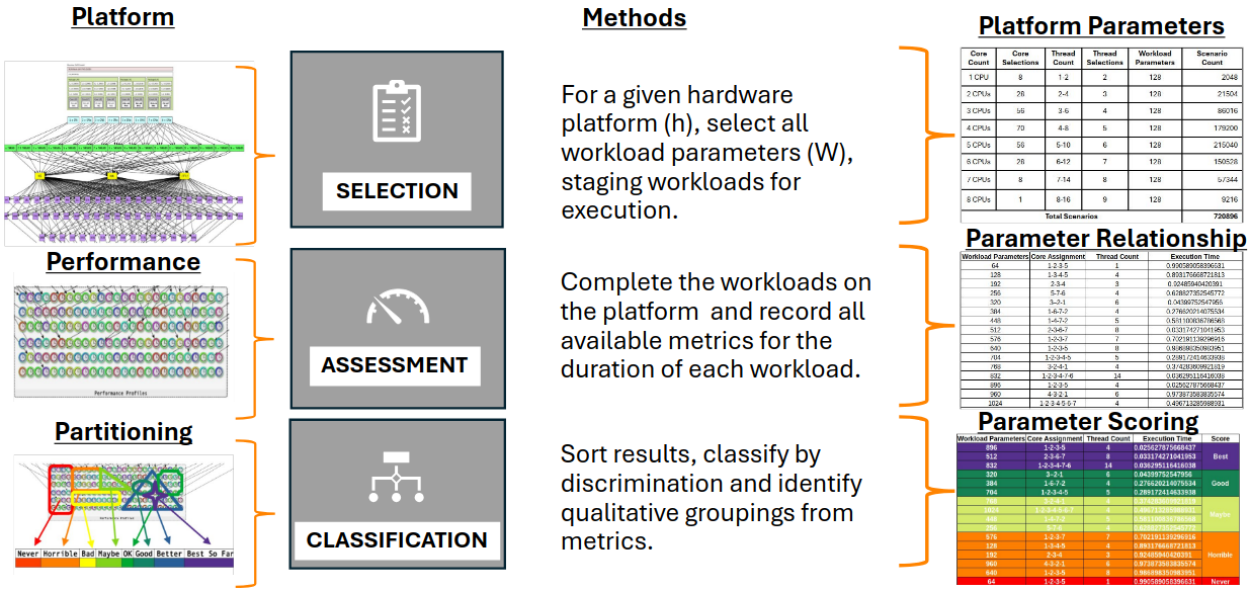


Figure 3.4: Applied Selection, Assessment, Classification in Action

Chapter 4

Performance Models for Non-Uniform Heterogeneous Platforms and their Validation Using Structural Equation Modeling

4.1 Introduction

Non-uniform heterogeneous computing architectures have become increasingly pervasive across modern mission-critical edge-cloud and embedded computing domains. In addition to conventional workloads such as network processing and data analysis, emerging cyber-physical tasks—including real-time sensing, high-throughput perception, and localized control—increasingly demand inference-capable edge deployment. To support these multi-faceted execution requirements, platform vendors have widely adopted heterogeneous chip designs replete with feature-rich component collections. Representative industrial examples include ARM (big.LITTLE), Intel (Performance/Efficient), Qualcomm (Kryo), and MediaTek (Dimensity).

The primary advantage of asymmetric multiprocessing platforms (AMPs) lies in their execution flexibility wherein the multiplicity and diversity of processor and memory characteristics offer unprecedented opportunities to optimize trade-offs between execution throughput, energy consumption, system utilization, and real-time latency deadlines. However, this flexibility introduces significant orchestration and management challenges. Because a task's placement and execution target may not be known *a priori*, realizing the full performance

potential of these platforms requires principled characterization of the microarchitectural characteristics that dictate execution behavior.

Compared to traditional homogeneous multiprocessor platforms, experience with AMPs remains limited, and conventional analytical frameworks often prove insufficient. On uniform processors, identifying a feasible execution profile typically scales with raw core count at a given clock frequency. Heterogeneity breaks these uniform relationships wherein core heterogeneity augments traditional compute semantics, rendering simple volume metrics (e.g. the number of active cores) inadequate. Instead, scheduling and task-mapping frameworks must explicitly account for the distinct *constitution* and structural parameters of each compute unit as a specific *type*.

Because the polymorphic design space of AMPs grows exponentially with platform and workload parameter choices (e.g., active core counts, dataset working set sizes, etc), exhaustive empirical profiling quickly becomes time-prohibitive even for modest systems. Nevertheless, we posit that the workload-to-platform relationship in AMPs possesses an underlying, discoverable structure that can be leveraged for scalable heterogeneous performance modeling across entire platform classes.

In this chapter, we present principled performance analysis techniques and evaluate their effectiveness empirically on the Rockchip RK3588S-equipped Orange Pi 5 platform (an ARM big.LITTLE system). Working toward new abstractions and coordination models to navigate emerging edge architectures, we make the following contributions:

- We formulate and evaluate parameterized first-principles performance models that exploit the discoverable structural relationships between workload dimensions and heterogeneous core configurations.
- We introduce multivariate Structural Equation Modeling (SEM) to validate statistically our modeling assumptions and analyze low-level microarchitectural bottlenecks, cache cliffs, and memory starvation.
- We demonstrate performance equivalence relationships across distinct physical core islands, providing a foundation for dynamic task mapping and resource allocation on edge AMPs.

Finally, the work presented in this chapter is adapted from the peer-reviewed journal publication:

> Harris, S. D., Chamberlain, R. D., and Gill, C. D. (2026). "Performance Models for Non-Uniform Heterogeneous Platforms and their Validation Using Structural Equation Modeling." *International Journal of Networking and Computing*, 16(1), pp. 20–51.

A subset of the journal publication was presented at the 27th Workshop on Advances in Parallel and Distributed Computational Models, associated with the 39th IEEE International Parallel and Distributed Processing Symposium, held in Milan, Italy, in June 2025.

4.1.1 Chapter Overview

This chapter establishes foundational mathematical modeling and multivariate statistical validation frameworks for asymmetric multiprocessing platforms (AMPs) operating at the edge. In the broader scope of this dissertation, this work serves as an analytical anchor. Understanding and modeling non-uniform hardware configurations is a prerequisite to navigating more complex multi-parameter interactions (Chapter 5), resolving physical-electrical "architectural entanglement" via measure-theoretic thermodynamics (Chapter 7), and enforcing real-time clinical constraints in edge biomedical devices (Chapter 8). Here, we develop first-principles execution models and pioneer the application of Structural Equation Modeling (SEM) to validate the architectural and workload assumptions that underpin edge scheduling.

The remainder of this chapter is structured as follows. Section 4.2 details our formal research hypotheses. Section 4.3 outlines hardware core topology configurations, workload characteristics, and hardware performance counter setups. Section 4.4 follows the experiments conducted on the platforms to investigate platform performance and architecture-specific dependencies. Section 4.5 details the structural equation modeling validation workflow, covering measurement models, specified structural models, bootstrapping, and memory-bound workload simulations. Section 4.6 summarizes the takeaways of this chapter. Finally, Section 4.7 draws conclusions and projects forward by explicitly bridging the findings to the multi-parameter interactions investigated in Chapter 5.

4.1.2 Problem Statement & Objectives

Asymmetric multiprocessing platforms (AMPs) demand radically different orchestration paradigms to manage the compute diversity inherent to this polymorphic design space. This chapter utilizes the Orange Pi 5 (an ARM big.LITTLE system) to deconstruct these novel architectural topologies. The resulting performance analysis illustrates the necessity of new techniques. We will show that these empirical results provide initial foundations for several new performance modeling methodologies designed both to navigate and to articulate the feature-rich landscape of edge-computing architectures.

Task performance is highly sensitive to execution profile parameters, including *core counts*, *clock frequencies*, *thread scheduling*, and *workload dimensions*. Unlike uniform processors, establishing a feasible baseline profile on an AMP requires discrete evaluation of each compute unit *type*, even prior to optimization. This hardware asymmetry severs the traditional linear relationship between workloads and compute units. Core heterogeneity fundamentally alters compute semantics, causing performance expectations based solely on the raw volume of active cores at a given clock speed to under-specify performance articulation. Therefore, scheduling models must account explicitly for the microarchitectural constitution of each processing unit, as the task of optimization on AMPs is no longer merely a function of compute *volume*, but of compute *constitution*.

Because empirical profiling is a time-intensive multistage process, exhaustive design-space exploration on AMPs quickly becomes computationally prohibitive, even for "small" parameter spaces. However, the structured workload-to-platform relationships documented in homogeneous systems can be generalized to heterogeneous environments. We posit that execution behavior on AMPs is systematically knowable and structurally applicable across broader classes of asymmetric platforms. Using the Orange Pi 5, this chapter proposes and evaluates mathematical performance models that explicitly exploit this underlying architectural structure.

4.1.3 Chapter Research Questions

To deconstruct the performance dynamics of AMPs systematically, this chapter investigates the following four fundamental research questions:

1. How are the performance parameters characterized?
2. Are there equivalence relationships between components?
3. Can we classify the performance behavior?
4. Can we find a model that captures platform behavior for accurate performance prediction?

4.2 Research Hypotheses

To address the challenges of resource planning and scheduling on AMPs, we propose and evaluate four central hypotheses:

Hypothesis 1: Discoverable Structure in Workload-to-Platform Relationships. We posit that workload behavior naturally clusters into distinct groupings partitioned by the numbers and types of active compute units. This structure includes regions of overlapping performance, where separate configurations yield statistically similar execution profiles for a given workload.

Hypothesis 2: Component Equivalence Relationships. Based on the expected overlapping performance, we hypothesize that there exist distinct configurations across different core types that can satisfy the same performance requirements, establishing component equivalence relationships.

Hypothesis 3: Existence of a Generalizable Model. We assume that the relationships between platform parameters and workload execution metrics can be modeled as a function:

$$\vec{P} = F(\vec{A}, \vec{H}, \vec{S})$$

where $\vec{P}$ is the performance vector (e.g., execution time, throughput, energy), $\vec{A}$ is the application vector (e.g., matrix dimension, cache locality), $\vec{H}$ is the hardware vector (e.g., instruction set architecture, core counts, memory hierarchy), and $\vec{S}$ is the system software configuration (e.g., scheduling policies, thread counts).

Hypothesis 4: Existence of the Inverse Model. Given the forward model F, we assume the existence of an inverse mapping F^{-1} such that:

$$(\vec{A}, \vec{H}, \vec{S}) = F^{-1}(\vec{P})$$

This relationship acts as a powerful discriminator, allowing an operating system scheduler to input desired performance targets and output precise configuration subsets (core types and speeds) required to satisfy those parameters.

4.3 Methodology

Single board computers are promising foundational devices through which to explore and understand the complexities of emerging architectures. These devices highlight a realistic range and capacity of platforms that may be found in practice for many industry applications. In this chapter we consider the Orange Pi 5 architecture that is representative of important capabilities of platforms operating at the edge. We use the Orange Pi 5 as a baseline frame of reference for investigation into the larger AMP design space. We will use the GEMM workload as outlined in Chapter 2

4.3.1 Performance Model

Our goal is to develop a performance model from first principles, examine how well that model fits the data, refine the model as needed, and validate the assumptions that are present in the model via structural equation modeling techniques. To frame our approach in the context of Section 4.2, our performance vector $\overrightarrow{P}$ may be a simple metric such as execution time, T_e; our hardware vector $\overrightarrow{H}$ is (N_B, N_L), the number of Big cores and the number of Little cores; our application vector $\overrightarrow{A}$ is the matrix dimension, $M = K = N$ (we will use M in what follows); and the system software $\overrightarrow{S}$ entails one thread for each core.

First-Principles Mathematical Derivations

To construct an analytical representation of workload execution, we begin with the computational complexity of our GEMM workload. Since matrix multiplication is algorithmically bounded by $\mathcal{O}(M^3)$ operations, we define the application work, W, as a cubic function of the matrix dimension M:

$$W = a_3 M^3 + a_2 M^2 + a_1 M \tag{4.1}$$

where the coefficients a_3, a_2, and a_1 act as platform-dependent scaling constants that represent different physical stages of the execution:

- The cubic coefficient a_3 represents arithmetic floating-point operations.
- The quadratic coefficient a_2 corresponds to memory bandwidth requirements for storing, retrieving, and aligning matrix elements.
- The linear coefficient a_1 accounts for pre- and post-processing steps, loop initialization, and OpenMP thread synchronization overhead.

On uniform cores, the expected execution time T_e scales inversely with the number of processing units:

$$T_e = \begin{cases} \frac{W}{N_L} & \text{if executing exclusively on Little cores} \\ \frac{W}{N_B} & \text{if executing exclusively on Big cores} \end{cases} \tag{4.2}$$

To unify these equations into a single continuous performance model across heterogeneous configurations, we express the total available processing capacity as an equivalent composite core count, N_C. We define a dimensionless Big-to-Little execution performance ratio, R_{BL}, which normalizes the computational capacity of a Big core to the finest available compute granularity—a single-threaded Little core:

$$N_C = N_L + N_B R_{BL} \tag{4.3}$$

Using Equation 4.3 to collapse our execution model from Equation 4.2 yields the first-principles performance equation:

$$T_e = \frac{W}{N_C} = \frac{a_3 M^3 + a_2 M^2 + a_1 M}{N_L + N_B R_{BL}} \tag{4.4}$$

While Equation 4.4 provides a solid global approximation, empirical trials reveal that the performance ratio R_{BL} is not constant but varies based on the working set size (S_{WS}) of the workload. Specifically, as the matrix footprint transitions across cache boundaries, R_{BL} undergoes a discrete step change. We incorporate this non-linear behavior by defining a conditional performance ratio:

$$R_{BL} = \begin{cases} R_{BL.\text{Low}} & \text{if } M < S_{WS} \\ R_{BL.\text{High}} & \text{if } M \geq S_{WS} \end{cases} \tag{4.5}$$

where the spatial transition threshold S_{WS} is localized empirically around $M = 1000$.

To achieve high-fidelity prediction, we further refine this framework into a fine-grained, continuous re-parameterization where the composite core count and execution time are evaluated as a continuous function of the exact matrix dimension M:

$$N_{C.M} = N_L + N_B R_{BL.M} \tag{4.6}$$

$$T_e = \frac{W}{N_{C.M}} \tag{4.7}$$

Table 4.1 summarizes the notation used throughout the paper.

Performance Metric Telemetry

Finally, to evaluate the microarchitectural phenomena affecting execution, we acquire low-level hardware telemetry during runtime. To quantify the microarchitectural mechanisms

Table 4.1: Notation and Description

Symbol	Description
$a_i, \quad 1 \leq i \leq 3$	Cubic coefficients
$\overrightarrow{A}$	Application vector; (M)
$\overrightarrow{H}$	Hardware vector; (N_B, N_L)
M	Matrix dimension
N_B	Number of Big cores
N_C	Number of composite cores
N_L	Number of Little cores
$\overrightarrow{P}$	Performance vector; (T_e)
R_{BL}	Big-to-Little ratio
S_{WS}	Working set size
T_e	Execution time
W	Application work

driving these coarse-grain timing trends, we complement macro-level execution measurements with fine-grained performance counter telemetry using the Linux 'perf' tool to capture high-resolution samples of hardware performance counters with the sampled metrics including:

- `cycles` and `instructions` to compute Instructions Per Cycle (IPC).
- `l1_dcache_loads` and `l1_dcache_load_misses` to assess primary cache locality and bandwidth bottlenecks.
- `llc_loads` and `llc_load_misses` to track data movement through the unified cache and trace the boundaries of memory starvation.
- Operating system overhead metrics, including `context_switches` and `cpu_migrations`.

While execution time establishes baseline performance across hardware configurations, hardware metrics including `Instructions Per Cycle` (IPC), `L1 data cache load misses`, `Last Level Cache` (LLC) starvation, `context switches`, and `CPU migrations` can provide

visibility into the underlying CPU pipeline and memory hierarchy bottlenecks. Bridging runtime behavior with low-level hardware phenomena sets the stage for our empirical evaluation. We begin with a coarse-grain workload-to-platform assessment across island configurations and matrix dimensions before leveraging the hardware telemetry to isolate the exact microarchitectural constraints governing platform efficiency.

4.4 Experiments

These experiments present an empirical evaluation of the heterogeneous platform under varying workload parameters and hardware configurations. To characterize the performance trade-offs, we systematically evaluate matrix multiplication workloads across core types (Big, Little, and Dual islands), core counts, and problem scales. The experimental methodology progresses from a coarse-grained workload-to-platform assessment to fine-grained microarchitectural analysis, correlating macro-level execution timings with hardware performance counters. Through this structured evaluation, we map the execution behavior across the configuration search space and validate the hypotheses established in Section 4.2.

4.4.1 Parameters: Island + Matrix Dimension

The first set of experiments can be viewed as a coarse-grain workload-to-platform assessment. Starting with island selection, we iterate through all available configuration options to identify the performance implications for the workload based on parameter (island, matrix dimension) selection.

Figure 4.1 illustrates workload execution time by matrix dimension. Colored by island selection (Big, Dual, Little), the results indicate that Little cores have the longest execution time, with Big and Little combinations (i.e., Dual island) taking up the middle ground, followed by Big cores having the shortest execution time. Notice that many of the data points overlap for different island selections.

Coarse-grained analysis is a wonderful tool for orienting the performance of a platform. However, once oriented, further exploration is required in the direction of the performance

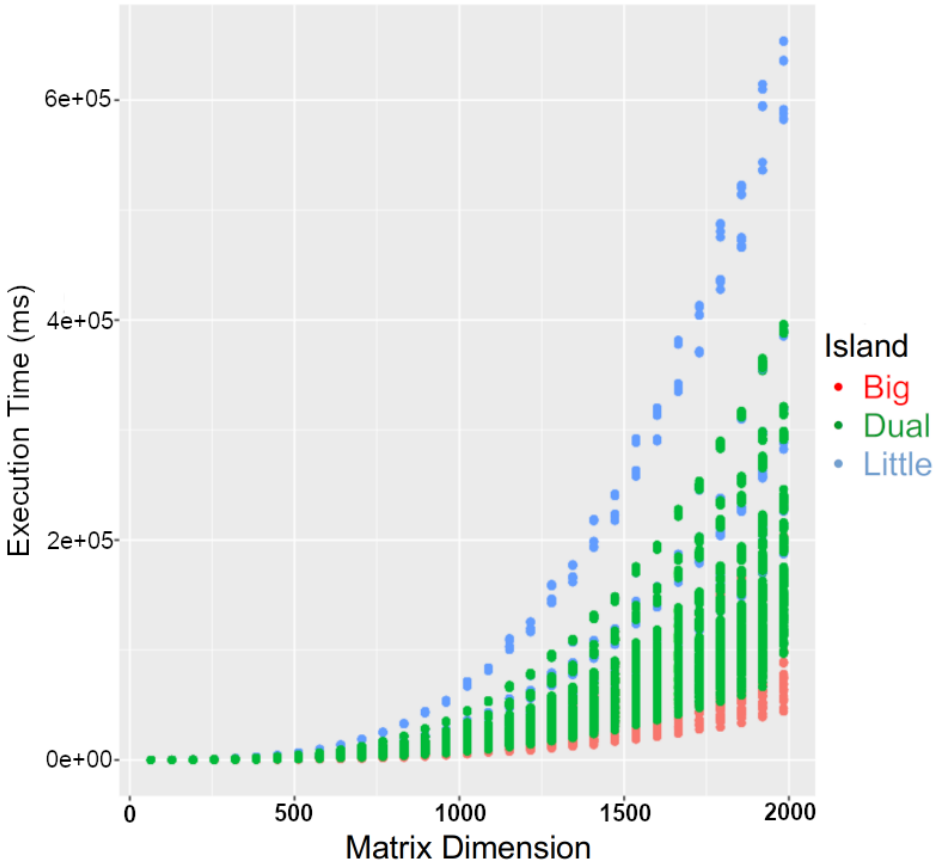


Figure 4.1: Orange Pi 5 Matrix Multiplication by Island

goals. These introductory experiments confirm that there is a structure to the workload-to-platform performance. If the overlap is valid, it also suggests equivalence relationships between configuration components.

4.4.2 Parameters: Island + Matrix Dimension + Cores

In contrast to the coarse-grained analysis of the two parameters as shown in Figure 4.1, is an example of a finer-grained analysis shown in Figure 4.2, utilizing additional parameters to include the number of cores (and threads, though to limit the data density we only present cases where the core counts match thread counts). The additional parameters provide clarifying insights on the prior performance assessment parameters.

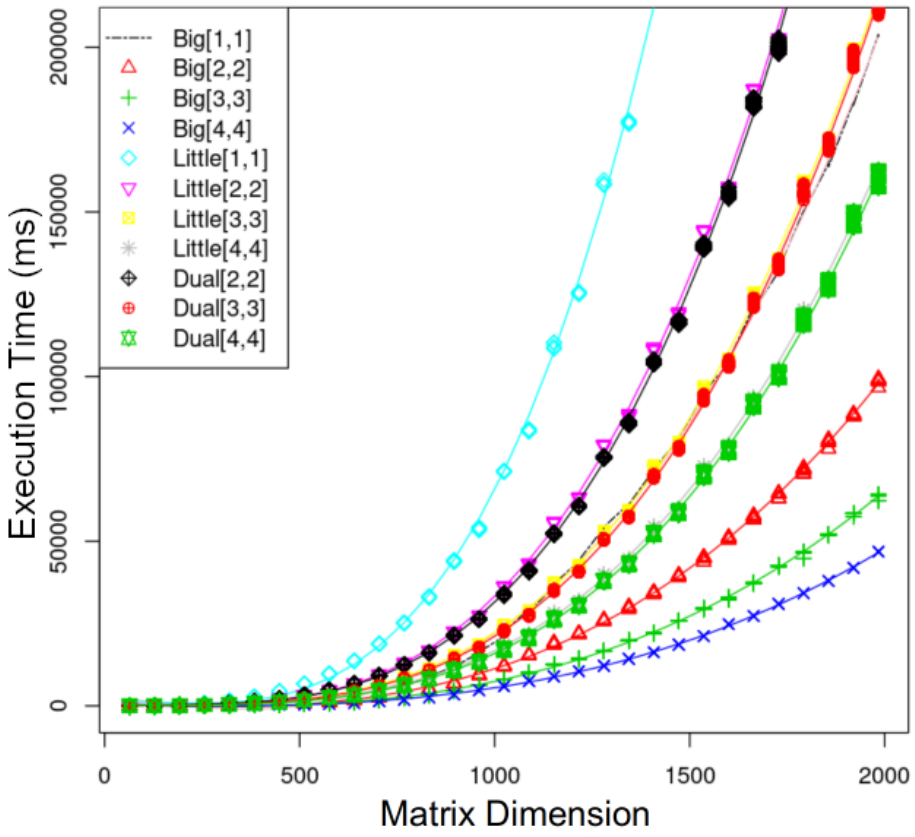


Figure 4.2: Orange Pi 5 Matrix Multiplication by Configuration

To ease configuration differentiation, each set of experiments is labeled using the format: **Island[Cores,Threads]**. For example, the lowest performance configuration occurs for workloads utilizing one core and one thread executing on any Little island core (i.e., Little[1,1]).

With execution time on the y-axis and the matrix dimension on the x-axis for Figure 4.1 and Figure 4.2, each point represents the empirical workload execution time. In Figure 4.2, a cubic trend-line is fit to each configuration, indicated by solid lines (i.e., the coefficients a_i in Equation (4.1) are separately determined for each configuration). In all cases, the measurements fit the cubic line regression function with an R-squared value of 0.9989, comfortably close to a perfect fit. This function is consistent with the run-time for a standard (3-loop) matrix multiplication computation. We see, clearly and as expected, that matrix multiplication has a cubic execution time relationship with the workload.

More interesting is that Figure 4.2 also exhibits substantial performance overlap between distinct configurations, pointing to an underlying fundamental structure to the platform performance: this is in direct support of the first and second hypotheses articulated in Section 4.2. We then want to quantify, classify, and order the discovered performance. To do so, we will normalize performance to the smallest compute granularity (i.e., minimum workload performance) on the platform to use as the basis for measurement in the discussion below.

4.4.3 Performance Normalization

Given the observed performance, we denote a single-threaded Little core as the finest compute granularity. The Little core hardware capabilities referenced in Figure 2.3, along with the demonstrated performance as shown in Figures 4.1 and 4.2 support this choice for smallest compute granularity. Next we scale the performance of each configuration to that of one single-thread Little core, reflected in Table 4.2. With respect to the smallest compute granularity, each configuration has a speedup, an average ratio (across all matrix dimensions) of target configuration execution time to the reference execution time (Little[1,1]), and the standard deviation of this ratio. The speedup is linear as core count increases, and is relatively fixed as we transition from a Little island to a Big island. Little cores have less variability across matrix dimensions as opposed to their Big core counterparts, as reflected in the standard deviation (SD). Supporting the notion of normalization to Little core configurations, the SD is at-least one order of magnitude smaller than any Big core configuration.

Table 4.2: Big-to-Little Normalized Ratios

Configuration	Speedup	Average Ratio	Standard Deviation
L[1,1]	100%	1	0
L[2,2]	200%	0.50	0.010
L[3,3]	300%	0.33	0.005
L[4,4]	400%	0.25	0.004
B[1,1]	300%	0.32	0.089
B[2,2]	600%	0.15	0.022
B[3,3]	1000%	0.10	0.029
B[4,4]	1400%	0.07	0.016

At this stage, in lieu of a comprehensive series of statistically significant measurements and analysis confirming the relationships, let us assume that the performance trends are valid (we will re-examine this assumption below). This is a simple assumption often made at run-time for orchestration decisions. Given the configuration relationships, we should be able to model the execution requirements in terms of the matrix dimension (fit to a Little[1,1] configuration, i.e., using the coefficients a_i from that individual configuration), the number of cores, and the Big-to-Little ratio, utilizing Equations (4.1), (4.3), and (4.4). The predictive modeling is shown in Figure 4.3 with the measured execution times presented as data points and each line showing the modeled (predicted) time for the configuration. The presentation is reversed for Big[1,1] with actual values shown as lines and predictions denoted with data points to differentiate the overlapping performance trends (with Little[3,3]).

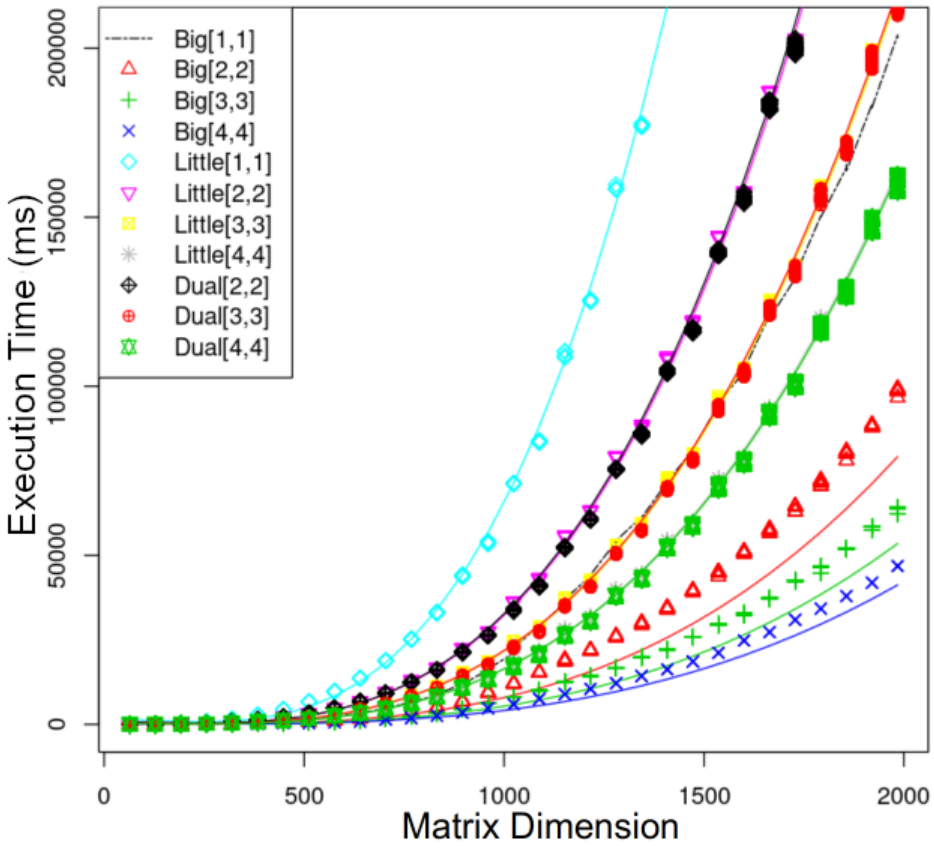


Figure 4.3: Static Big-to-Little Ratio

Our first observation is that, for many purposes, this model does a good job of predicting the performance across all configurations tested. While there is some separation between

modeled and measured results for large problem sizes on the speediest configurations, overall it does a good job.

Looking a bit deeper into the set of configurations that don't match the model so well, the obvious prediction deviations are readily observable for the Big island multi-core configurations. Notice in Figure 4.3 that for Big[2,2] through Big[4,4] the predictions fall below the actual execution time. In contrast, notice that the single core Big island configuration (i.e., Big[1,1]) oscillates between over- and under-prediction, suggesting a parameter dependent performance change across the matrix dimensions. We investigate these discrepancies next.

4.4.4 Parameter Characterization

The prediction results in Figure 4.3 and the normalized ratios calculated in Table 4.2 indicate a measure of data variability. It can be challenging to see from this vantage point how the variation is impacting our predictive ability. Thus, we need to expand our perspective and reorient the data. We will change perspectives and look at the normalized Big-to-Little ratios graphically.

Figure 4.4 shows the Big-to-Little ratio (actually, the ratio of execution time for all Big and Little configurations relative to Little[1:1]) as a function of the matrix dimension. This plot reveals a hidden performance nuance that otherwise would have escaped our attention. The normalized ratio for Big[1,1], has distinct ratio intervals depending on matrix dimension. There appear to be stable performance ratios for intervals of matrix dimensions between approximately (300, 800) and (1200, 1700). For the sake of discussion, we could base an assessment around the inflection point occurring at matrix dimension 1000. We see a performance dip in other Big core configurations as well, though the effect seems to diminish with increasing core numbers. These results suggest that proper performance prediction will require normalization over subset intervals of workload parameters to maintain performance portability.

To ground this idea, we again normalize all of the experiments to the lowest execution unit performance (i.e., Little[1,1]). If our assumptions are correct, we should be able to predict the performance of other configurations by simply using the Big-to-Little performance ratio, denoted R_{BL}. However, the ratio is no longer assumed to be constant across the entire

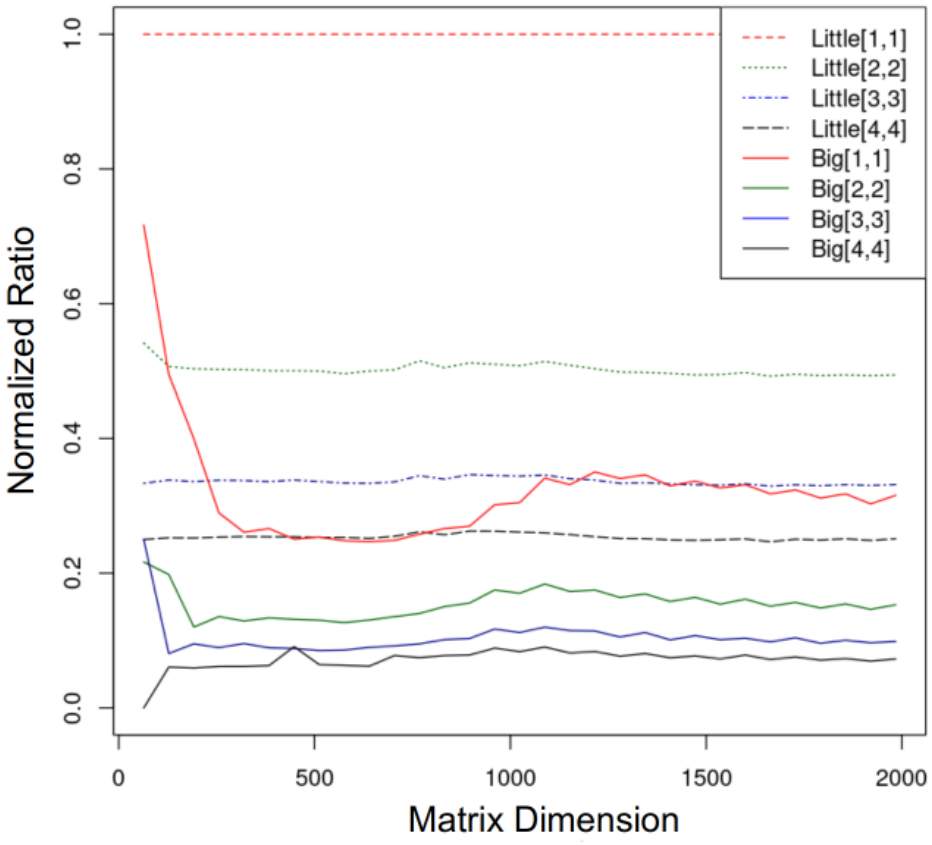


Figure 4.4: Normalized Execution Time Ratio

range of matrix dimensions. As illustrated in Figure 4.4, the speedup for all Big cores varies across matrix dimensions. This is particularly pronounced for the Big core configuration using one-core and one-thread (i.e., Big[1,1]), where the behavior only begins to stabilize around matrix dimension 300, but shifts again around matrix dimension 1000. Initially, the performance for the Big[1,1] configuration is concave. Around matrix dimension 1000, there is an inflection point shifting performance to a convex profile. Our model accommodates the different behavior before and after this point, with the ratio being conditional on the matrix dimension, e.g.,

$$R_{BL} = \begin{cases} R_{BL.Low} & \text{if } M < 1000 \\ R_{BL.High} & \text{if } M \geq 1000. \end{cases} \tag{4.8}$$

Note that this is simply a re-expression of Equation (4.5) with an empirically determined threshold of 1000. This analysis depends on profiling the performance behavior across a wide range of configurations and workloads. We hypothesize that while additional workload parameters may improve the accuracy of our model, a smaller set of key parameters may be

sufficient to characterize and predict system performance. In Section 4.4.5 we explore a few of the system parameters to understand which components give clarifying insight into the dynamic platform behavior.

4.4.5 System Parameter Correlation

We continue our investigation by probing lower-level interactions to identify parameters that are critical to modeling platform performance effectively. Using a wide range of metrics garnered during execution from the Linux performance measurement tool `perf`, we investigated the pairwise-correlations between a number of metrics. The results presented above suggest that the matrix dimension plays an important role in the overall performance, but are there other predictors within the platform that support this notion?

For each configuration, we took samples of `perf` metrics to understand the overall behavior during workload execution. Looking at the pairwise correlations illustrated in Figures 4.5 and 4.6, we can see clearly that the correlations change dramatically across workload parameters. Notice in Figure 4.5 that the L1 data cache miss-percentage is negatively correlated with cycles, instructions, ipc, and L1 data cache metrics (e.g., loads and misses). This combination of negative correlations suggests data starvation occurring at the core level. All instruction related metrics rates would decrease as they wait on data to load into the L1 data cache.

If we are correct in our starvation assumption, the correlation should change as we increase the amount of available data. Increasing the workload size (i.e., matrix dimension) supports data availability to the cores. As we look to the larger matrix size in Figure 4.6, the negative L1 data cache miss-percentage correlation with cycle rate, cycles, instructions, ipc, and L1 load and miss metrics improves. Notice that negative correlation pivots from L1 data cache miss percentage metrics to context switch and Last-Level-Cache (LLC) Load metrics. While negative correlation with L1 caches persists, notice that for the larger size matrix dimension the L1 data cache miss percentage is actually improving. This suggests that our starvation bottleneck has moved above the core level (beyond L1/L2 caches) to the LLC. While far from definitive, the above supports our suggestion that the conditional in Equation (4.5) referencing an empirically determined working set size, denoted S_{WS}, is a reasonable notion.

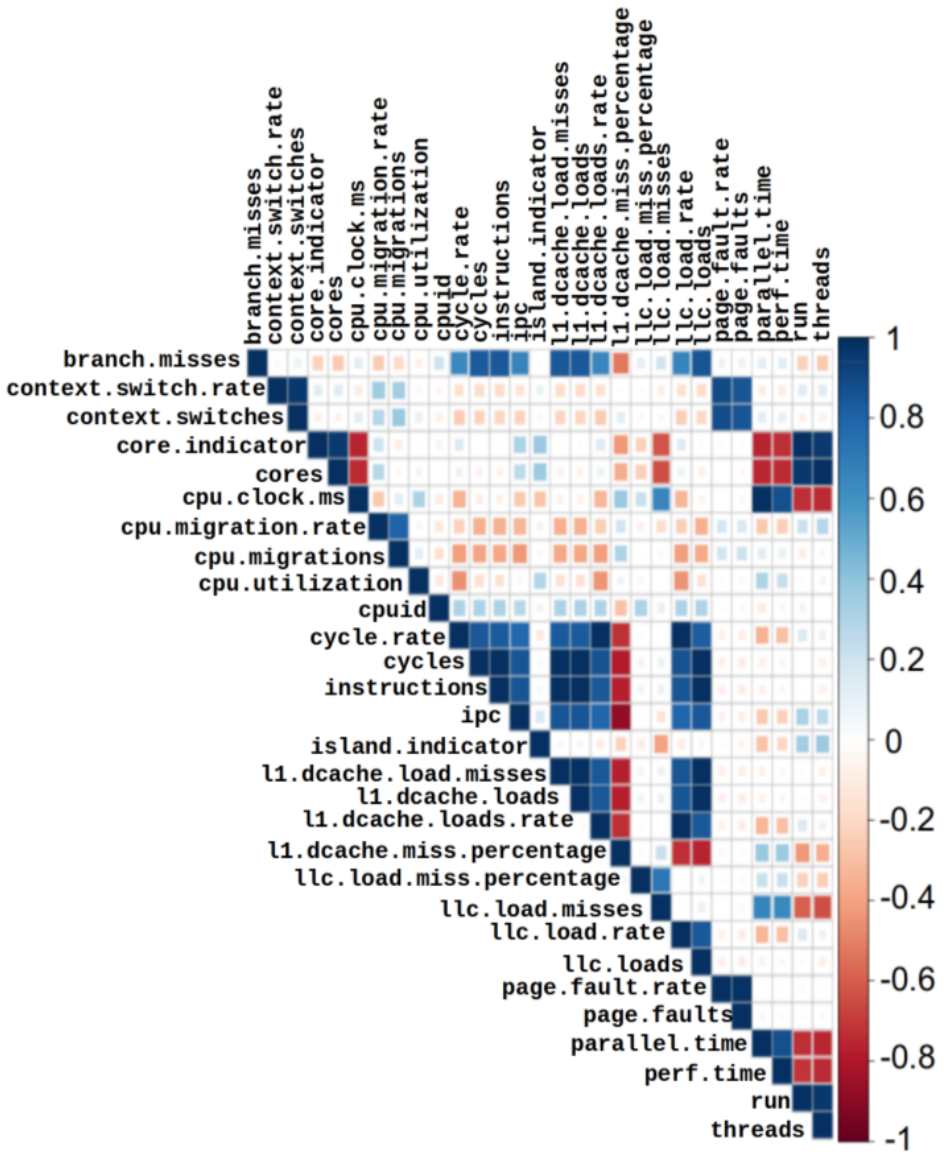


Figure 4.5: Correlations – 768 × 768 Matrices

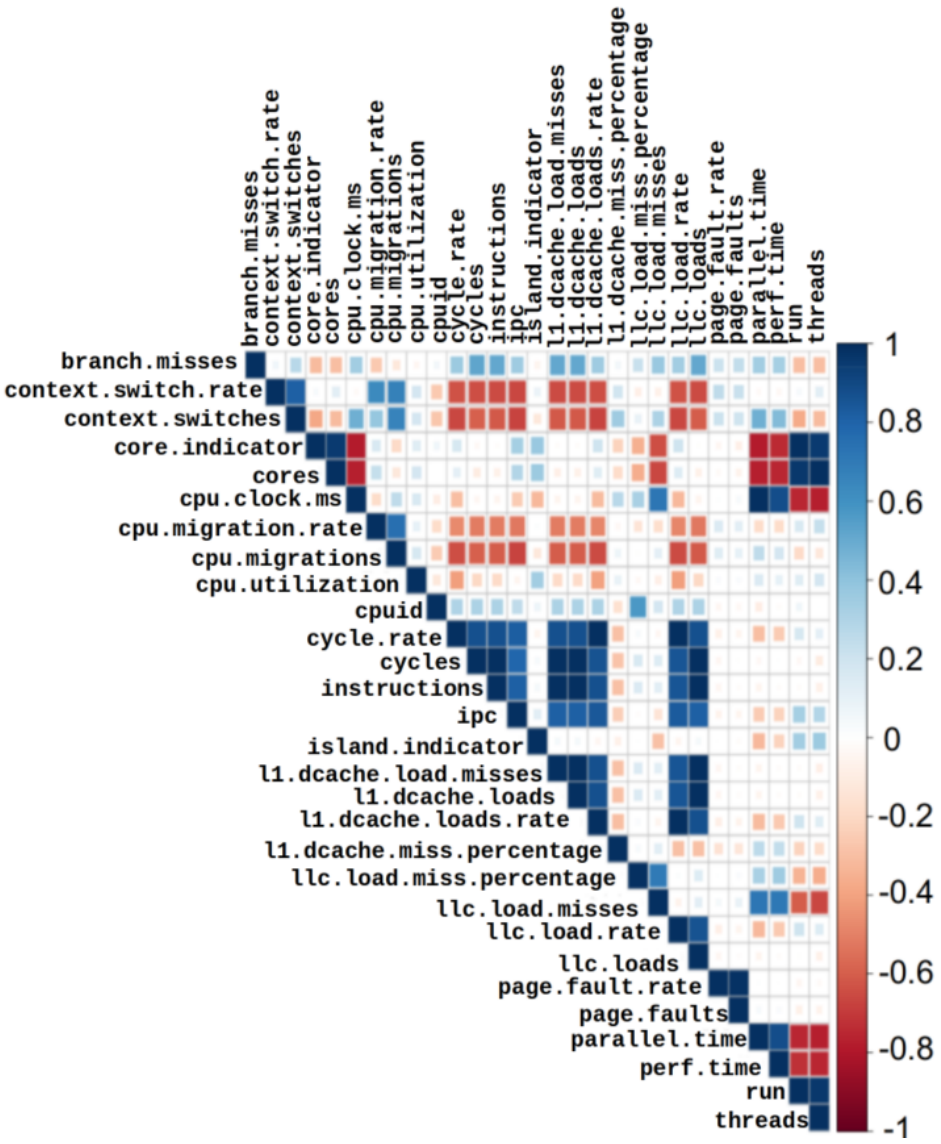


Figure 4.6: Correlations – 1216 × 1216 Matrices

4.4.6 Re-parametrization

The data clearly suggest that the performance varies with the workload size. Therefore, by setting a threshold, we can condition on the matrix dimension in a coarse-grained fashion as depicted in Equation (4.8), where we partition the performance into stable regions. Instead, suppose that we went a step further and created a Big-to-Little ratio as a function of the specific matrix dimension M, denoted $R_{BL.M}$, so that Equation (4.3) becomes:

$$N_{C.M} = N_L + \frac{N_B}{R_{BL.M}} \tag{4.9}$$

and Equation (4.4) becomes:

$$T_e = \frac{W}{N_{C.M}}. \tag{4.10}$$

Thus, using Equation (4.10), we are able to calculate and predict performance accurately using the fined-grained Big-to-Little ratio for all independent configurations. As there may be some confounding behavior using the joint (dual) cores, we will not perform that analysis in this work. As depicted in Figure 4.7 each data point reflects actual experimental values with lines being used to predict the performance using the re-parameterized Big-to-Little Ratio. The exception is in the case of the Big[1,1], where the actual data is a presented as a line, and the prediction is a data point. The line travels exactly through the center of each data point. We reversed plotting for Big[1,1] to distinguish between the overlapping performance with Little[3,3] and to illustrate how closely the predicted mappings follow the actual data.

Finally, taking the mean ratio across all matrix dimensions gives rise to the predictions shown in Figure 4.8. The distinction between these two graphs being that in Figure 4.7 the predictions follow the empirical performance precisely, given a fine-grained Big-to-Little ratio, while Figure 4.8, using the mean ratio, is an example of the course-grained approximation.

4.4.7 Can we validate the behavior?

The results are indicative of a clear relationship between the workloads and architecture. In the case of the Orange Pi 5, we found that there is performance overlap between combinations

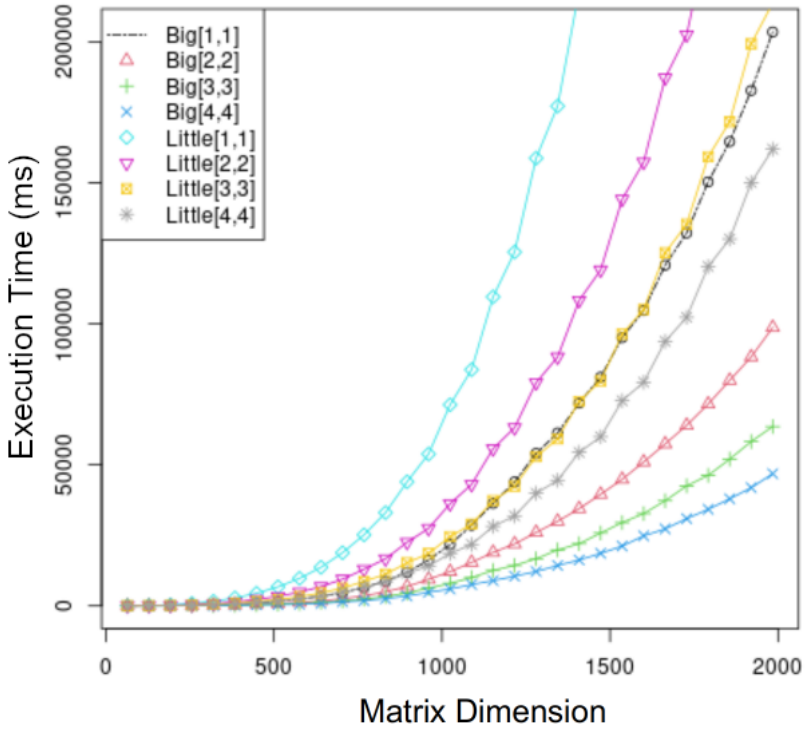


Figure 4.7: Re-parameterized Big-to-Little Ratio

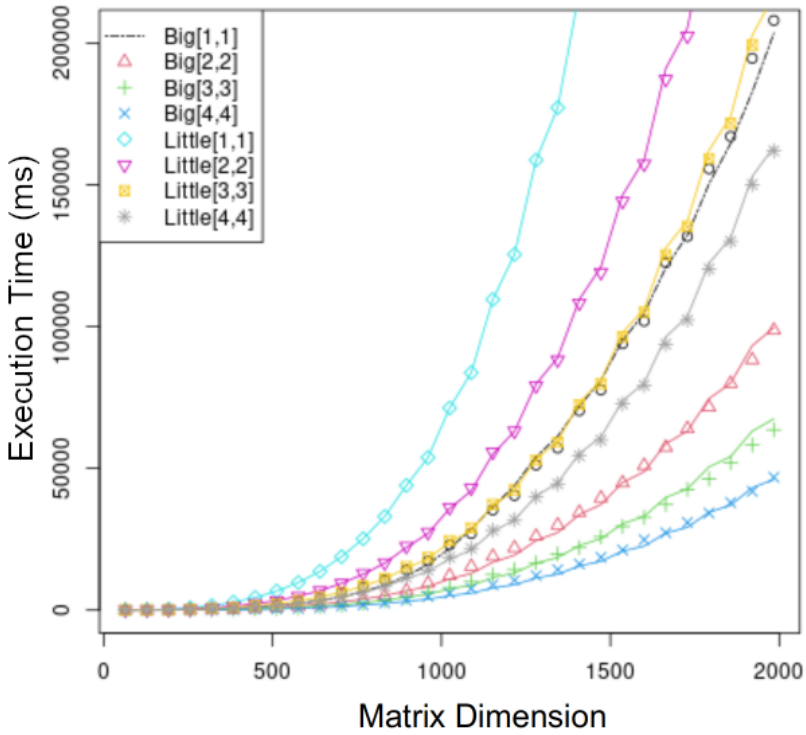


Figure 4.8: Complete Model

of Big and Little cores. Particularly, we have found that we can model the mean execution time for our workloads on one Big core as equivalent to that of three Little cores. Using the Big-to-Little ratio calculation, we identified a pair of configurations suggesting equivalent performance, particularly, the L[3,3] and the B[1,1] as shown in Table 4.3.

Table 4.3: Big-to-Little Normalized Ratios

Configuration	Speedup	Average Ratio
L[3,3]	300.00%	0.3361
B[1,1]	300.00%	0.3225

Next, we wanted to determine if the mean execution time for one Big core and three Little cores are statistically different. Consequently, we perform a Welch two-sample t-test on the two means with the alternative hypothesis that the true difference in means is not equal to zero. Those results are shown in Table 4.4, showing the (P)aired and (U)npaired results, respectively. Both the paired and unpaired tests show significant results supporting the

Table 4.4: Big-to-Little Normalized Ratio Tests

Configuration	t	df	p-value
(P) Big[1,1] $\approx$ Little[3,3]	5.4069	31	6.687e-06
(U) Big[1,1] $\approx$ Little[3,3]	5.4069	31	6.687e-06

conclusion that the execution times are not statistically different from one another.

The above validation, however, is quite limited. While the agreement between the model and measured results is quite good, there are many assumptions that have been made in formulating the model for which we have, at best, only intuition to guide us. To help us assess the validity of these assumptions, we turn to a family of statistical analysis techniques called structural equation modeling.

4.5 Structural Equation Modeling

Structural equation modeling (SEM) techniques are widely used in social sciences, psychology, education, and business [44, 60]; however they are less frequently encountered in traditional engineering models, which are dominated by first principles models (e.g., of the type described above in Sections 4.3.1 and 4.4.6). We will exploit some of the capabilities of

SEM to assess the validity of the assumptions made in the development of the first principles model presented above.

4.5.1 SEM Measurement Model

Our initial use of SEM techniques will be to assess the choice of parameters used to develop the model of Equation (4.4) presented in Section 4.3.1. The input parameters are M, N_L, and N_B, the matrix dimension, number of Little cores, and number of Big cores, respectively. The output parameter is T_e, execution time. Note that in SEM analysis, we are using the entire raw dataset as is, whereas in the earlier model we would utilize the mean execution time for a set of experiments with the same input parameter values. Also, in this use of SEM, we are not trying to replicate the predictions of the earlier first principles model. Rather, we are trying to assess the relative importance of the input parameters and their impact on the output parameter. Using traditional statistical methods to center and scale the dataset, we therefore can compare variables irrespective of scale, identify relative importance, and interpret the effect size, even when the original units of measurement differ.

The set of variables used in our SEM measurement model is as follows:

Main Effects: Matrix Dimension, Little Island, Cores

`matrix_dimension_c` (Centered Matrix Dimension (`md_c`))

`island_L` (The Little Island)

`cores_i` (Cores ranging from i = 2 to 4)

2-Way Pairwise Interactions: Matrix Dimension `x` Little Island

`md_c_x_island_L` (Matrix Dimension Centered `x` Little Island)

2-Way Pairwise Interactions: Matrix Dimension `x` Cores

`md_c_x_cores_2` (Matrix Dimension Centered `x` Two Cores)

`md_c_x_cores_3` (Matrix Dimension Centered `x` Three Cores)

`md_c_x_cores_4` (Matrix Dimension Centered `x` Four Cores)

2-Way Pairwise Interactions: Little Island `x` Cores

`island_L_x_cores_2` (Little Island `x` Two Cores)

`island_L_x_cores_3` (Little Island `x` Three Cores)

`island_L_x_cores_4` (Little Island `x` Four Cores)

3-Way Pairwise Interactions: Matrix Dimension `x` Island `x` Cores

`md_c_x_island_L_x_cores_2` (Matrix Dimension Centered `x` Little Island `x` Two Cores)

`md_c_x_island_L_x_cores_3` (Matrix Dimension Centered `x` Little Island `x` Three Cores)

`md_c_x_island_L_x_cores_4` (Matrix Dimension Centered `x` Little Island `x` Four Cores)

We next examine the following measurement model postulating how all the parameters relate to execution time:

```
Execution Time = f(Main Effects + 2-Way Interactions + 3-Way Interactions)
```

The outcome of this analysis yields what is called the *standard estimate*. Standard estimates are indicative of the change in `Execution_time` (i.e., in standard deviations) for one-standard-deviation change in the predictor (the variables listed above). The standard error on the standard estimate gives a measure of the precision of that standardized estimate with a smaller standard error meaning a more precise estimate.

Starting with a pair of individual effects, the analysis shows:

- The matrix dimension has a strong impact. A one-standard-deviation increase in M is associated with a 0.705 standard deviation increase in T_e (standard error = 0.023).
- There are diminishing returns to increases in core count. E.g., going from 3 cores to 4 cores (irrespective of whether the cores are Big vs. Little) is associated with the smallest improvement in execution time (only 0.180 decrease in standard deviation of execution time for a one-standard-deviation change in core count, standard error = 0.010).

Neither of the above results is in any way surprising. In fact, they are precisely what one expects. However, they are now validated by statistical reasoning rather than simply justified by intuition.

We next examine some pairwise effects:

- We first assess the combined effect of matrix dimension and island. This is the strongest effect demonstrated, with a one-standard-deviation change in the interaction term between matrix dimension and island (Big vs. Little cores) associated with a 1.097 standard deviation change in execution time (standard error = 0.023).
- Pairing matrix dimension with core counts, we continue to see diminishing returns for additional cores.

All these effects (both individual and pairwise) are *statistically significant* ($\texttt{pvalue} \leq 10^{-3}$), meaning they are not likely due to random chance, given the large sample size.

Again, the results from statistical reasoning match our intuition. The combined effect of matrix dimension and Big vs. Little cores has a substantial impact on overall execution time. The above analysis gives credence to the model of Equation (4.4), plotted in Figure 4.3, which does a quite reasonable job of predicting performance over a wide range of configurations.

4.5.2 SEM Structural Model

We next turn to the use of SEM structural models to investigate the hypothesis that the relationship between working set size and cache is the proximate cause of the refinements to the model that were developed in Section 4.4.6.

Model 1: Intuitive Latent Variable Construction

Computer performance over the years has relied on the fact that computations are bound primarily in two dimensions: *time* and *space*. We now consider the effect of those two dimensions and some model implications:

Time (Processor Core Performance): The time complexity typically refers to the number of operations that a computational process can complete. This relates directly to processor performance (e.g., speed), as a more performant processor core can execute more instructions per second, thereby reducing time on some real-world task.

Space (Memory Cache Efficiency): The space complexity typically refers to the amount of storage space (or memory) that a computational process requires to run. This relates to memory capacity, but even with infinite capacity, a limiting factor on memory is data availability or the speed with which the data can be retrieved. Consequently, cache performance has a direct impact on the speed at which the computation can make use of the storage space.

Given these fundamental considerations, we need to derive intuitive structures that reflect these considerations within an appropriate model. We therefore start by hypothesizing a pair of latent variables, `Core_Performance` and `Cache_Efficiency`, in an attempt to separate out the effects of the individual cores and their caches (which we suspect is the issue to be addressed in this stage of the development of the first principles model). We then select a subset of the values measured (some via `perf`) in Section 4.4.5: `ipc`, `l1_dcache_load_rate`, and `matrix_dimension` associated with `Core_Performance` plus `threads`, `cores`, and `llc_loads` associated with `Cache_Effects`. These relationships are shown graphically in Figure 4.9. We see that there is a negative correlation (represented by the double-headed arrow) between `Core_Performance` and `Cache_Efficiency`, which implies that as the score on one latent variable tends to increase, the score on the other latent variable tends to decrease. In other words, they are inversely related.

Notice the single arrows (representing regression paths or direct effects) with the standardized parameter estimates shown on each line. Reading directly from Figure 4.9 we can see the impact of each parameter not only on the latent variables but the impact of each latent variable on the (latent) variable of interest, `Performance`. Notice that the R^2 value for `Performance` is only 0.619, which is not a good score.

This model explains a bit over half of the variability in the model but we postulate that the model is underspecified and there are additional constructs and/or factors that must be incorporated into the model to improve its accuracy. Essentially, our latent variable `Performance`, though influenced by `Core_Performance` and `Cache_Efficiency` is only regressed on one factor (`parallel_time`). We have shown above that we can have overlapping

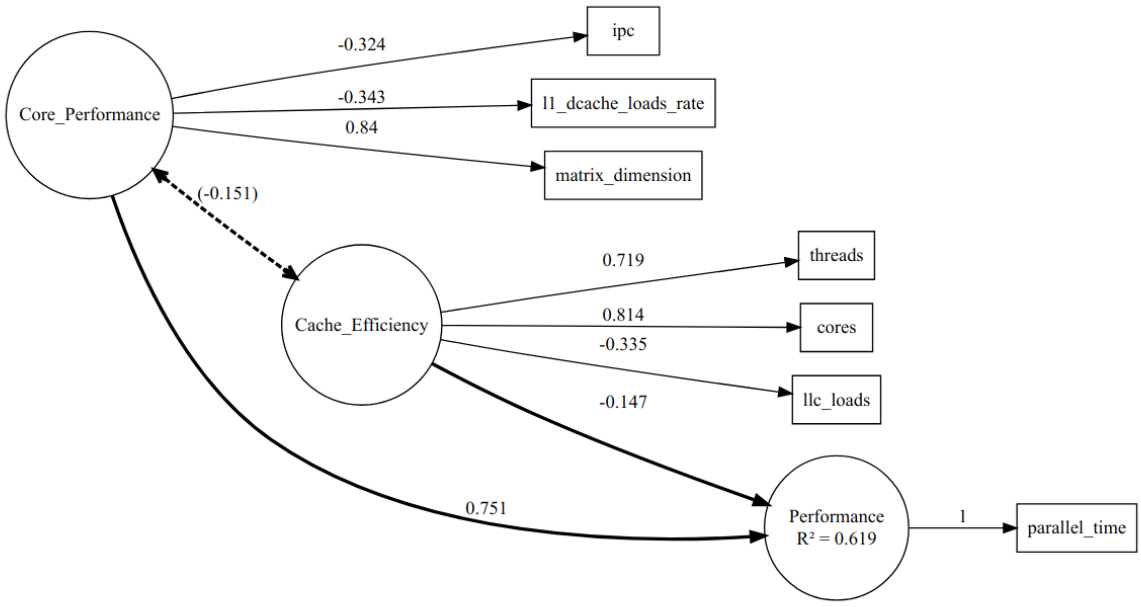


Figure 4.9: Initial Full SEM Model

performance for one or more configurations (e.g., one Big core is equivalent to three Little cores) which indicates that a single factor is not enough to differentiate the performance variation. In the subsequent model, we will add an island parameter that will enable differentiation of `parallel_time` with respect to island thereby increasing our ability to express the platform behavior for the given factors.

Model 2: Evidence Directed Parameter Addition

Intuitively, our principled analysis methods would support the notion that given a `parallel_time` and including an island indicator may assist in better explaining the performance. In the model shown in Figure 4.10 we test to see if it is the case, by regressing the latent construct `Performance` on an additional variable `island_numeric`, which is an indicator variable used to identify the island type (Big or Little). This is clearly beneficial to

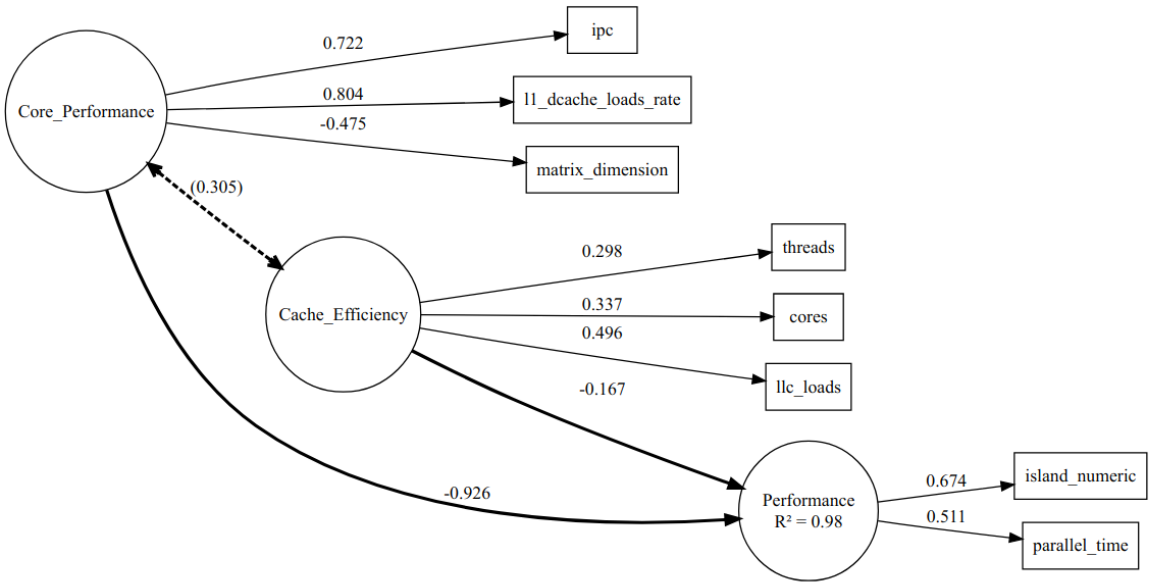


Figure 4.10: Specified SEM Structural Model

our model as the R^2 jumps up to an excellent 0.98 where we can infer that the parameters shown explain 98% of the performance. These methods are based off of a single analysis run. Next we will bootstrap the model to ensure we have the proper estimates.

Model 3: Bootstrapping for Parameter Significance

Given that our initial model had some challenges we added an additional parameter which adjusted the R^2 value but this is one of many models and we would like to understand the parameters in our model better before we seek to find the best model. To help us understand the parameters in our model, we will bootstrap the SEM model.

Bootstrapping, particularly for SEM models, is a power resampling technique used to estimate the precision of parameter estimates (like factor loadings, regression coefficients, or

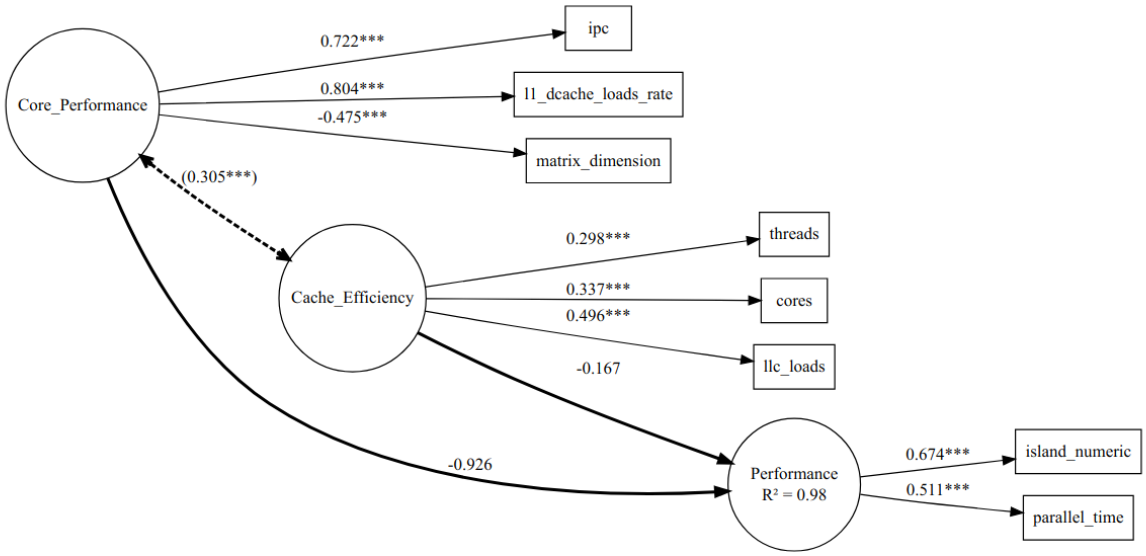


Figure 4.11: Bootstrapped Specified Model

indirect effects) and to assess their statistical significance without relying on traditional parametric assumptions. Essentially, bootstrapping performs:

Resampling with Replacement: From the original dataset of N observations, a new sample of size N is drawn with replacement. This resampling process is repeated many times, in this instance 1,000 times, creating a large number of *bootstrap samples*.

Estimation for Each Sample: The full SEM model (both measurement and structural components) is estimated for each of these bootstrap samples. This results in a distribution of estimated parameters (e.g., a distribution of beta β_i values, a distribution of factor loadings, and a distribution of indirect effects) for each parameter in the model.

Inference: As opposed to relying on theoretical sampling distributions (and assuming normality), bootstrapping uses the empirical distributions to derive standard errors,

confidence intervals, and p-values. The standard deviation of a parameter's bootstrap distribution becomes its bootstrap standard error. Bootstrap confidence intervals—e.g., 95% bias-corrected and accelerated (BCa) confidence intervals—are constructed from the percentiles of the bootstrap distribution. If a confidence interval for a parameter does not include zero, that parameter is considered statistically significant.

As indicated in Figure 4.11, we find all the factors for each latent construct to be statistically significant (as shown by the triple stars $***$ next to each of the parameters' standardized estimates). Of particular importance for our purposes is that the relationship between the latent structures `Core_Performance` and `Cache_Efficiency` are also statistically significant. This provides a level of confirmation that the interaction of working set size and caches was a reasonable motivation for the choices made in the extended version of the first principles model of Section 4.4.6.

4.5.3 Performance Simulation for SEM Structural Model Confirmation

The SEM Structural Model is a powerful tool for understanding the relationship between the parameters. We were able to select an intuitive model and confirm the relationships through our empirical measurements. Yet, there is a subtle importance that needs to be promoted as it is often overlooked. The SEM models that we have outlined specify the relationships between the parameters, which may fit not only the current dataset but a large range of datasets for the platform, given that the model is based on the underlying relationships between the hardware-agnostic components. Moreover, though matrix multiplication is typically compute bound, to round out the analysis we would like to understand how well this methodology would generalize for memory-bound workloads or the many irregular workloads which are commonly found in edge/embedded computing. Since our analysis is based on empirical measurements made from hardware-agnostic parameters, to understand memory-bound workloads, we only need to consider how the parameters change when the workload shifts from compute bound to memory bound. Suppose that we consider matrix multiplication workloads which are memory bound, what can we say about these parameters?

Relationship and Proportionality Changes

For the sake of discussion, let us consider only the model parameters used with the exception of *threads* and *cores* which will no doubt change the scaling behavior but will lose is ability to translate proportional speedup as in the memory-bound scenario, the memory subsystem lags behind causing these parameters to lose their strong impact on performance. We anticipate that as workloads shift from compute-bound workloads to memory-bound workloads we will experience a shift in metrics as outlined in Table 4.5.

Table 4.5: Anticipated Metric Shift

Model Parameters	**Processor to Memory Metric Shift**
Instruction Per Cycle (ipc)	Lower: CPU stalls
L1 Data Cache Load Rate (l1_dcache_loads_rate)	Higher: Data exceeding L1/L2 capacity
Last Level Cache Loads (llc_loads)	Higher: Data retrieval from L3/LLC
Matrix Dimension (matrix_dimension)	Higher: Matrices exhausting L1/L2/L3/LLC
Parallel Execution Time (parallel_time)	Higher: Latency and limiting bandwidth

CPU-bound to Memory-bound Metric Distributions

To support this notion, we have synthetically perturbed the existing dataset to shift parameters into memory-bound constraints, with the original dataset being shifted according to the memory-bound values shown in Table 4.6. Given the heterogeneity of the platform, we cannot assume the distribution of the data and therefore cannot simulate new data as the distributions may differ dramatically. Instead, we choose to permute the existing dataset to maintain the original distributions but changed the values so that they reflect the memory-bound constraints that we wish to investigate. To support our decision to rescale the data, for each parameter shift consideration, we will illustrate the simulated distribution overlaid with the original distribution, along with the mean values used for modeling a memory-bound workload against the original compute-bound dataset for comparison.

It is highly unlikely that a single constant of proportionality exists between compute-bound and memory-bound workloads. Yet, a simple model that either simulates a metric shift or perturbs the data in this way will prove both effective and useful in determining the efficacy of our SEM models in light of such considerations. For a naive 3-loop matrix multiplication

workload shifted from compute-bound to memory-bound characteristics, we anticipate that the distributions for sampled metrics on the same platform would change according to prior performance analysis and intuitive mathematical reasoning as outlined in the compute-bound to memory-bound prediction given in Section 4.5.3 below.

Table 4.6: Model Parameters and Investigated Metric Shift

Shifted Model Parameters	**Memory-Bound Values**
Instruction Per Cycle (ipc)	Decrease by 75%
L1 Data Cache Load Rate (l1_dcache_loads_rate)	Increase by 800%
Last Level Cache Loads (llc_loads)	Increase by 250%
Matrix Dimension (matrix_dimension)	Increase by $10,000\%$
Parallel Execution Time (parallel_time)	Increase by $2,000\%$

Metric Predictions

Here, we make several predictions of impacts that we anticipate will result from a shift to a memory-bound workload. These predictions will then be validated in Section 4.5.3.

Prediction: Instruction Per Cycle (ipc)

Compute-Bound: The IPC would likely be high and tighly clustered as this reflects the processor execution many instructions per clock cycle because of the general availability of fast caches, registers, and memory subsystem enabling high utiliation of execution units.

Memory-Bound: The IPC distribution may be significantly lower and broader. Particularly, the peak of the distribution would shift to the left as the processor stalls frequently waiting for data to be fetched from the slower main memory (DRAM) into cache. The broadening may also reflect variability in the memory access latency.

Prediction: L1 Data Cache Load Rate (l1_dcache_loads_rate)

Compute-Bound: This rate would most likely be high and tightly clustered. The CPU which running at high speed would rapidly request data from the L1 cache to furnish the execution units.

Memory-Bound: This rate distribution may shift lower. Though the CPU continues to request data, the effective rate would decrease due to the frequent stalling of the processor. The distribution shape may become wider due to the variable stalling behavior caused by memory access times.

Prediction: Last Level Cache Loads (llc_loads)

Compute-Bound: The LLC loads distribution would most likely be lower or moderate. With data moving through the caches, a CPU-bound workload will make use of data more effectively within the L1/L2 caches resulting in fewer compulsory and/or capacity misses reaching the slower LLC.

Memory-Bound: The LLC loads distribution most likely would be significantly higher. The shape may be similar but translated to a higher mean. The memory-bound workload would be characterized by poor data locality and frequent lower level (L1/L2) cache misses ultimately resulting in a high request volume that must be addressed by the LLC.

Prediction: Parallel Execution Time (parallel_time)

Compute-Bound: The distribution would be expected to be lower (given faster execution times) and potentially more skewed to the right. This would be due in part to potential uniformity of matricies and a lack of any significant memory contention with the time being dominated by computations (FLOPS).

Memory-Bound: The distribution would be higher (given longer execution times). In this instance, the total execution time would be dominated by memory access latency due to time spent waiting for data resulting in a broader and potentially heavier right tail or multi-modal appearance if there are significant non-uniform memory access (NUMA) effects or variability in memory contention and/or bandwidth saturation across runs with the time being dominated by memory stalls.

Metric Shift Results

As shown in Table 4.7, using the dataset with the parameters that simulate a memory-bound workload is identical to the previous bootstrapped model shown in Figure 4.11. This

is because the SEM model expresses the underlying relationship between the parameters which intuitively should not change because of the workload as the data will flow through the same components and the relationships between those components should not change. This analysis further confirms the relationships between components and the structure of the model. Plots that confirm the predictions above are shown in Figures 4.12 to 4.15.

Table 4.7: Bootstrapped Memory-Bound Model Regressions and Variances

	Estimate	Std.Err	P(>z)
Regressions:			
Performance ~			
Cache_Efficiency	-0.1667	0.0219	0.000
Cache_Efficiency ~ cores	0.3368	0.0385	0.000
Cache_Efficiency ~ llc_loads	0.4960	0.0167	0.000
Cache_Efficiency ~ threads	0.2975	0.0377	0.000
Core_Performance	-0.9262	0.0137	0.000
Core_Performance ~ ipc	0.7222	0.0110	0.000
Core_Performance ~ l1_dcache_loads_rate	0.8045	0.0112	0.000
Core_Performance ~ matrix_dimension	-0.4751	0.0150	0.000
island_numeric	0.6740	0.0087	0.000
parallel_time	0.5109	0.0119	0.000
Variances:			
Performance	0.9796	0.0204	0.000

4.6 Discussion

We organize the discussion of the models into two parts, first discussing the first principles model and following that with a discussion of the structural equation modeling.

4.6.1 First Principles Model

The coefficients on the workload expression, a_3, a_2, a_1, are likely proportional to factors that determine how efficiently the platform can perform arithmetic operations, handle memory access, and manage initialization and completion requirements. As the number of cores increases, these coefficients decrease as additional processing units allow for faster computation

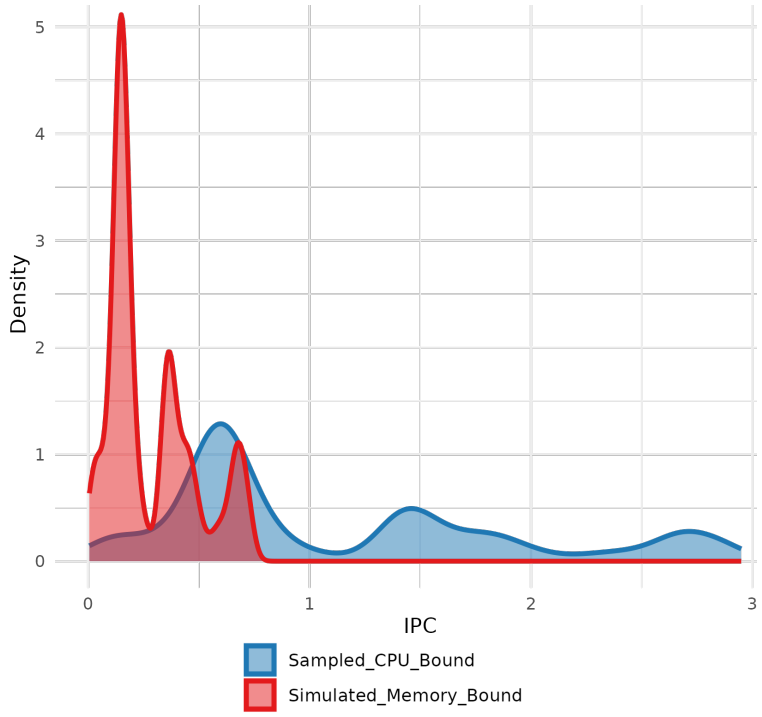


Figure 4.12: IPC Distribtution, Mean: Sampled (1.13) vs. Simulated (0.28)

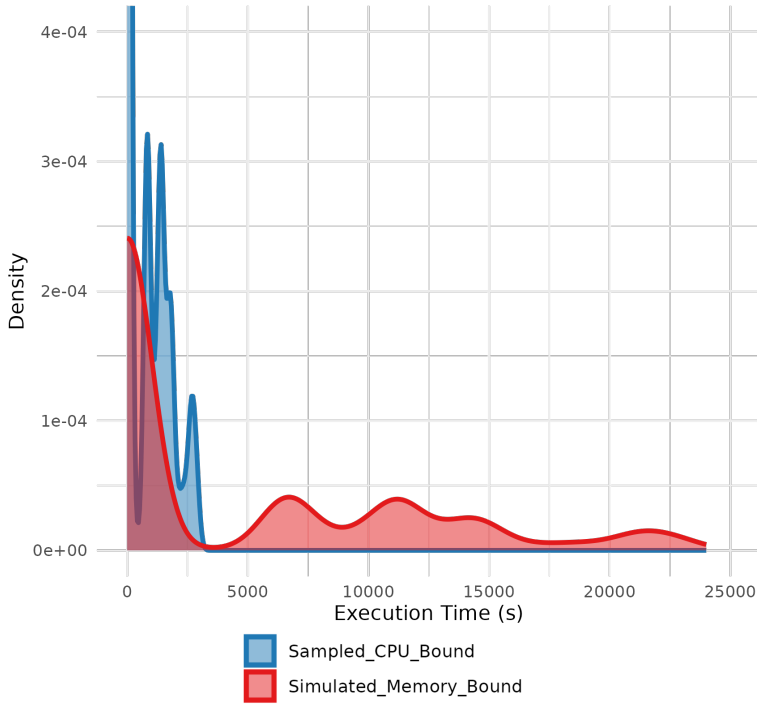


Figure 4.13: L1 DCache Loads Rate Dist., Mean (s): Sampled (593) vs. Simulated (4,744)

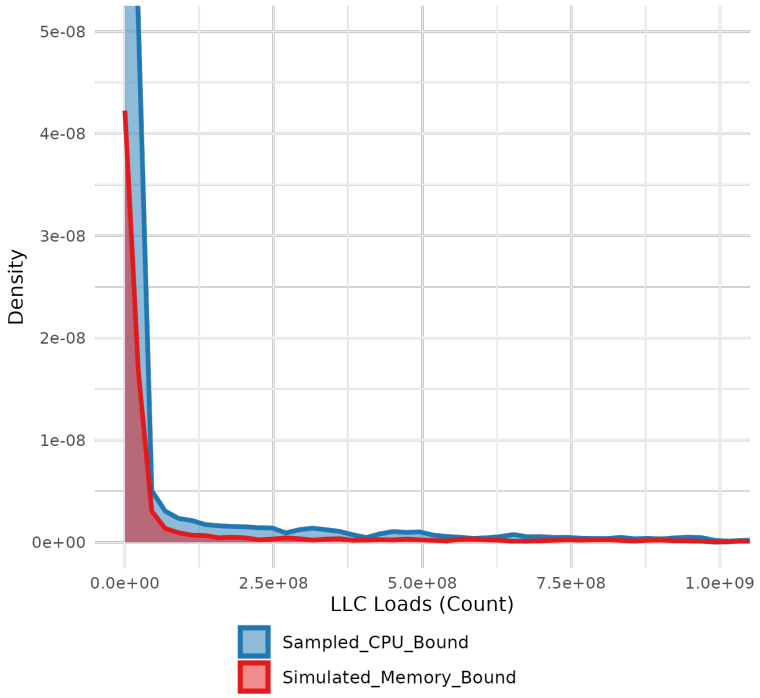


Figure 4.14: LLC Loads Distribution, Mean (Count): Sampled (114,658,410) vs. Simulated (286,646,026)

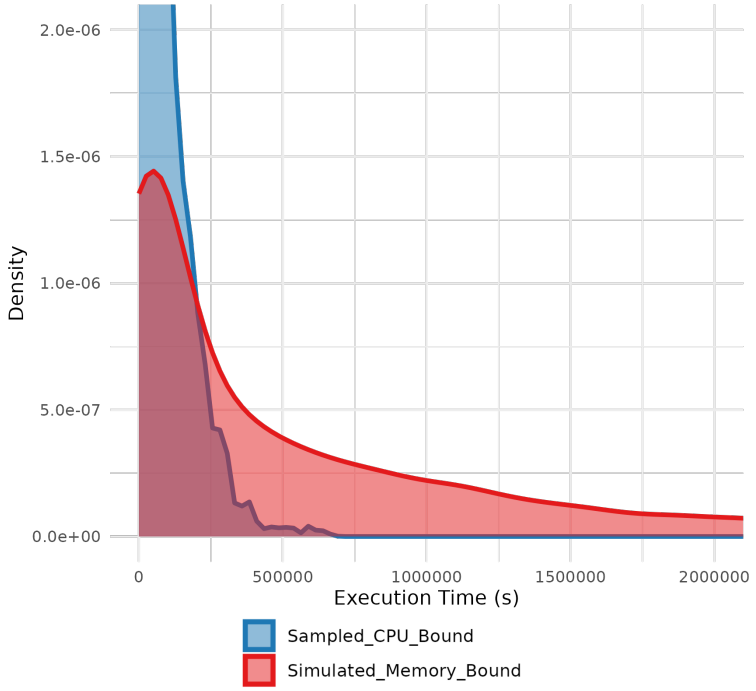


Figure 4.15: Parallel Time Distribution, Mean (s): Sampled (14,287) vs. Simulated (285,740)

and improved data handling, reducing memory and lower-level overheads through additional parallelism.

Each coefficient corresponds to different stages of the computation, specifically:

1. The cubic term ($a_3 M^3$) represents arithmetic operations.
2. The quadratic term ($a_2 M^2$) corresponds to memory bandwidth required for storing and retrieving matrix elements.
3. The linear term ($a_1 M$) accounts for both pre- and post-processing steps.

As the number of cores increases, these coefficients are influenced by the parallelism in computation, memory access, and initialization:

- a_3 decreases as more cores allow faster arithmetic operations.
- a_2 decreases as memory bandwidth improves with more cores.
- a_1 decreases due to improved task granularity and scheduling.

Therefore, increasing cores reduces each coefficient proportionally.

Overall, the model presented in Sections 4.3.1 and 4.4.6 does a good job both in matching empirical measurements (indicating correctness) and also providing insight into how and why certain behaviors exhibit themselves. Both of the above points are important in judging the quality and usefulness of the model.

The above supports the latter two hypotheses presented in Section 4.2. The forward model (actually, several forward models) has been shown and empirically validated, and all of the expressions are invertible.

4.6.2 Structural Equation Modeling

While SEM techniques can be used to develop models that directly predict outcomes given particular inputs, that is not how we are using them. Instead, we are using SEM techniques

(both SEM measurement models and SEM structural models) to help us assess whether or not the assumptions made in the first principles model were appropriate.

As such, the initial use (measurement models) is quite conclusive, demonstrating the importance of the input parameters chosen in the first principles model, in addition to confirming several performance effects that we already strongly suspected.

The second use (structural models) is supportive of our assumptions. The bootstrapped model of Figure 4.11 shows a statistically significant relationship between the latent variables representing both core performance and cache efficiency, with the pathways connecting them to the performance latent variable being statistically significant as well.

The above implies that there is likely additional hidden structure in the data (i.e., additional latent variables) that we have not yet included in the SEM structural model. For example, the performance latent structure is associated with the vector $\overrightarrow{P}$ and the core and cache latent structures are associated with the vector $\overrightarrow{H}$. Potentially we could include in the model a latent structure associated with $\overrightarrow{A}$, and maybe even $\overrightarrow{S}$. Investigating alternative models that could expose this structure is left for future work.

4.6.3 Insights

The heterogeneous performance space is structured, knowable, and exploitable. Analyzing the Orange Pi 5 platform demonstrates that parallel workloads on asymmetric multiprocessing architectures yield predictable, statistically validated relationships. The experiments in Section 4.4 reveal this underlying structure, allowing precise navigation of the multi-objective performance space via analytical modeling.

Primary takeaways and hypothesis resolutions:

> **Workload-to-Platform Structure (H1 & H3):** Performance clusters naturally based on physical core configurations (Section 4.4.2). Modeling Big execution as a scaled version of Little execution tightly aligns empirical measurements with theoretical predictions.

Component Equivalence (H2): Section 4.4.3 verifies execution overlap between heterogeneous configurations. Welch t-testing confirms that a single Big core equals three Little cores (Big[1,1] vs. Little[3,3]), yielding a statistically stable 3:1 performance ratio over target workloads.

Predictive Modeling (H4): Normalizing execution to the smallest compute unit enables precise time prediction (Section 4.4.3). We derived a continuous, straightforwardly invertible first-principles forward model ($T_e = \frac{W}{N_{C.M}}$). Section 4.4.6 confirms the model's precision upon re-parameterization for workload size dependencies.

SEM Validation: Structural equation modeling statistically validates first-principles parameter choices. SEM structural models verify cache-cliff bottlenecks and prove robust framework generalization for simulated memory-bound workloads (Section 4.4.5).

This chapter establishes a robust first-principles model on a balanced platform assuming isolated resource consumption. Exploiting asymmetric systems requires resolving how dynamic, multi-parameter interactions operate across varying topologies. Chapter 5 expands this framework to the highly asymmetric Radxa Cubie A7A (2+6 configuration). It systematically analyzes multi-parameter interactions to isolate interconnect bottlenecks and the "funnel effect." Chapter 5 also introduces the Critical Complexity Index (χ) to mathematically define the exact thresholds where resource contention and inter-island friction neutralize parallel scheduling advantages.

4.7 Chapter Conclusions

While our initial work postulated abstract scaling vectors (**A**, **S**, **H**, **P**) operating under isolated resource consumption, expanding these concepts into non-uniform, asymmetric topologies required replacing decoupled regression with a unified, causal pipeline. The conceptual vector space has now materialized into the *Selection, Assessment, and Classification* (SAC) framework. By formalizing this progression as a composable operator chain ($\mathcal{G} = \mathcal{C}_h \circ \mathcal{A}_h \circ \mathcal{S}_h$), the SAC methodology transforms high-level speculative parameter projections into a physically grounded attribution pipeline. This advancement directly addresses the combinatorial explosion of the heterogeneous design space—mitigating the intractability of evaluating over 163,712 unique runtime scenarios per SoC—by establishing a structured foundation

for characterizing microarchitectural bottlenecks, parameter interactions, and performance portability across modern edge AMPs.

4.7.1 Evolution: From Foundational Vectors into the Tripartite SAC Methodology

The conceptual scaling vectors proposed in our future projections—applications (**A**), system software (**S**), hardware (**H**), and performance (**P**)—undergo a critical mathematical evolution to form the concrete *Selection, Assessment, and Classification* (SAC) performance characterization framework. Rather than treating these domains as independent, abstract coordinate axes, the SAC methodology formalizes them as a sequential chain of composable, substrate-dependent operators parameterized by the target hardware platform h:

$$\mathcal{G} = (\mathcal{C}_h \circ \mathcal{A}_h \circ \mathcal{S}_h)(W) = \mathcal{C}_h(\mathcal{A}_h(\mathcal{S}_h(W))) \tag{4.11}$$

This tripartite operator chain represents a logical transition from high-level, speculative scaling functions to a physically grounded, hierarchical performance attribution pipeline. Under this mathematical formulation, the multidimensional application vector (**A**) is projected onto the input workload space $W = \{w_0, w_1, \ldots, w_n\}$, representing the scaling parameters of the program (such as GEMM matrix dimensions). The system software vector (**S**), which encapsulates thread concurrency scaling and core-affinity controls, acts as the operational parameter of the *Selection* operator ($\mathcal{S}_h$). This first-stage operator maps high-level workload parameters to target instruction sequences ($W \xrightarrow{\mathcal{S}_h} I_h$), isolating the microarchitectural footprint before runtime metric collection occurs.

The abstract hardware vector (**H**) is materialized as the physical microarchitectural substrate h, parameterizing the second-stage *Assessment* operator ($\mathcal{A}_h$). The Assessment operator maps the instruction stream to quantitative performance profiles, such as parallel execution time ($I_h \xrightarrow{\mathcal{A}_h} E_h$). On simple, in-order Little cores (Cortex-A55), the behavior of $\mathcal{A}_h$ is highly deterministic ($R^2 = 0.9776$), as execution cycles scale proportionally with instruction count. However, on speculative, out-of-order Big cores (Cortex-A76), $\mathcal{A}_h$ must model a high-dimensional event matrix containing branch mispredictions, cache misses, and pipeline

stalls to capture the non-linear execution cliffs that arise when workloads scale past cache boundaries.

Finally, the performance vector ($\mathbf{P}$) is projected into discrete, qualitative performance categories via the *Classification* operator ($\mathcal{C}_h$): $E_h \xrightarrow{\mathcal{C}_h} G_h$. By partitioning the continuous execution co-domain into discrete performance classes (such as "Memory Bound" or "Compute Bound"), the Classification operator eliminates the need to navigate an infinite execution time space. This tripartite decomposition replaces empirical brute-force methods—which suffer from a combinatorial explosion of over $163,712$ unique configuration scenarios per SoC—with a structured, articulative modeling framework capable of characterizing heterogeneous performance portability.

Chapter 5

Understanding Asymmetric Multicores Using Structural Equation Modeling

5.1 Introduction

Heterogeneous architectures, more specifically Asymmetric Multicore Platforms (AMPs), have fundamentally disrupted traditional performance modeling paradigms. Legacy analytical models rely on architectural homogeneity to project performance both incrementally and linearly via simple constants of proportionality. Modern AMP edge infrastructure, such as the ARM big.LITTLE system-on-chip (SoC), rejects these linear assumptions, presenting a complex landscape of non-linear interactions across execution units, memory subsystems, and platform topologies. Critical metrics like execution time and power consumption do not exist in isolation, rather they are emergent properties of these non-linear parameter interactions.

This work examines Structural Equation Modeling (SEM) [61, 78] as a foundational analytical methodology for edge systems engineering to address performance modeling for these non-linear properties. SEM enables the formal behavioral articulation, performance navigation, and structural explanation of asymmetric design spaces. By evaluating modern AI/ML primitive workloads which we simulate via General Matrix Multiply (GEMM) operations across heterogeneous platforms including the Orange Pi 5 and the Radxa Cubie A7A, this research demonstrates that performance capabilities are driven not merely by execution unit count, but by the precise architectural constitution of those units. The interaction of workload and platform topology generates unique empirical signatures. To formalize these

multi-parameter dynamics, this chapter introduces a comprehensive suite of SEM-derived performance indicators that quantitatively decode asymmetric design spaces.

Central to this framework are *Complexity Sensitivity* ($\sum \beta_i \mathbf{M}_c^j$), which quantifies execution volatility under increasing task complexity, and *Inter-Island Scaling Friction* ($\mathbf{M}_c \times I_L$), which measures latency penalties incurred during cross-cluster data movement. Complementing these, *Multi-Core Saturation* ($\Delta\beta_i C_3 \rightarrow C_4$) captures the marginal utility of expanding active core counts to detect parallel overhead tipping points, while *Stochastic Coupling Significance* evaluates system susceptibility to architectural noise and runtime fluctuations. Synthesizing these dimensions, we formulate the *Critical Complexity Index* (χ), a metric derived from the ratio of interaction to direct SEM path coefficients that isolates the structural threshold where inter-island friction and cache-coherency traffic negate core efficiency, thus signaling when task scheduler isolation must supersede parallelism to prevent memory bus poisoning. Accompanied by empirical validation across heterogeneous platforms, this multi-indicator methodology provides a generalized framework for identifying resource utilization patterns, mapping platform-level trade-offs, and advancing heterogeneous performance modeling.

Finally, the findings presented in this chapter are adapted from the peer-reviewed publication:

> Harris, S. D., Chamberlain, R. D., and Gill, C. D. (2026). "Understanding Asymmetric Multicores Using Structural Equation Modeling." *Proceedings of the International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, IEEE, pp. 57–66.

An extended version of the manuscript has been accepted to the *International Journal of Networking and Computing* and is tentatively planned for publication in the January 2027 issue [51].

5.1.1 Chapter Overview

Within the continuum of this dissertation, this chapter advances from first-principles baseline performance modeling to a multivariate framework capable of dissecting dynamic, non-linear parameter interactions on Asymmetric Multicore Platforms (AMPs). By operationalizing Structural Equation Modeling (SEM), this work establishes a methodology to quantify cross-cluster latency, non-linear core saturation, and memory subsystem bottlenecks across heterogeneous system-on-chip (SoC) topologies. Specifically, we formulate a suite of empirical performance indicators—headlined by the Critical Complexity Index (χ)—that isolate the tipping points where inter-island scaling friction and cache-coherency overhead compromise parallel efficiency. Quantifying these multi-parameter dynamics provides the statistical and structural foundation required to evaluate physical-electrical entanglements via measure-theoretic thermodynamics (Chapter 7), and enforce real-time determinism in edge biomedical platforms (Chapter 8).

The remainder of this chapter is structured as follows. Section 5.2 details our successive path model pruning workflow to eliminate multicollinearity. Section 5.3 uses SEM to model performance as latent constructs. Section 5.4 explores the usage of SEM in tandem with first-principals models to develop richer more precise understand of workload and platform interactions. Section 5.5 reveals several new performance indicators that can be used to assess heterogeneous performance characteristics. Section 5.6 outlines our findings and conclusions while forming a strategic bridge leading into our investigation of processor independence in Chapter 7.

5.1.2 Problem Statement & Objectives

Classical computational abstractions evaluate hardware capability through deterministic, linear scaling laws, assuming performance scales proportionally with core counts or operational intensity. However, asymmetric multi-core architectures feature non-uniform memory access characteristics, shared last-level caches, and unified system bus interconnects that introduce severe non-linear bottlenecks under parallel arithmetic kernels. Exhaustive empirical profiling across the resulting multi-dimensional design space is computationally and temporally prohibitive. The objective of this chapter is to formulate a mathematically sound

statistical validation framework using SEM to quantify dynamic interaction effects. While broadly applicable across heterogeneous computing platforms, the framework is instantiated and evaluated here on Single-Board Computers (SBCs)

5.1.3 Chapter Research Questions

Fully leveraging AMPs for mission-critical edge AI/ML workloads requires answering several foundational questions. However, rather than focusing on a singular metric, this chapter investigates the following research questions:

1. Are there hidden insights within metric interactions?
2. Do those interactions hold the keys to new performance insights?
3. If so, where do these critical parameters reside?
4. What is the method of their identification?
5. Through what analytical frameworks can they be evaluated?
6. How do these interactions reveal non-obvious performance bottlenecks?

5.2 SEM Measured Variable Path Analysis Models

In our initial investigation, we use Measured Variable Path Analysis (MVPA), a subset of SEM analysis, wherein we analyze only the relationships between measured variables [89]. This technique helps us to understand not only the relationships between the exogenous and endogenous variables but also the degree to which each exogenous variable impacts the endogenous variable. We iterate through several models with the first regression model being

defined as follows.

$$\begin{aligned} \texttt{parallel_time} \sim\ & \texttt{threads} + \texttt{cores} + \texttt{matrix_dimension} + \\ & \texttt{cpuid} + \texttt{cpu_utilization} + \texttt{context_switches} + \\ & \texttt{cpu_migrations} + \texttt{page_faults} + \texttt{cycles} + \\ & \texttt{instructions} + \texttt{ipc} + \texttt{branch_misses} + \\ & \texttt{l1_dcache_loads} + \texttt{l1_dcache_load_misses} + \\ & \texttt{llc_loads} + \texttt{llc_load_misses} + \texttt{island_numeric} \end{aligned}$$

The path analysis model is extremely powerful as it enables the rapid interpretation of parameter effects and impacts. The first SEM path model, shown in Figure 5.1, depicts the relationship between independent variables shown on the left and and the dependent variable, parallel execution time (`parallel_time`) shown on the right. The single-headed arrows flow from the exogenous variables to the endogenous variable. In Figure 5.1 the coefficient for `threads` (i.e., -0.06) indicates that a one-standard-deviation increase in this exogenous variable corresponds to a 0.06 standard-deviation *decrease* of the endogenous variable `parallel_time`. The `cores` parameter follows the same pattern but with a smaller decrease of -0.02. Notice for the `matrix_dimension` parameter, a one-standard-deviation increase in matrix dimension is associated with a 0.13 unit standard-deviation increase in `parallel_time`. The signs of the estimates show whether the given parameter will increase or decrease parallel execution time. Subsequent iterations systematically pruned variables exhibiting secondary variance inflation and statistical insignificance to construct the terminal proxy. The effects of pruning on the parameter values are shown graphically in Figure 5.2.

This presentation modality allows the parameter effects to be identified easily from the model to understand the exact parameter impacts on the endogenous variables. Other interesting information gained from the parameter estimates in Figure 5.1 can be seen with the values for both `instructions` (-165.25) and `l1_dcache_loads` (164.18). The two parameters have similar magnitude, several orders of magnitude larger than the other estimates, but differ by sign. This indicates that these variables may be exhibiting multicolinearity wherein these parameters are expressing the same underlying phenomena and simultaneous inclusion in the model exacerbates the estimates.

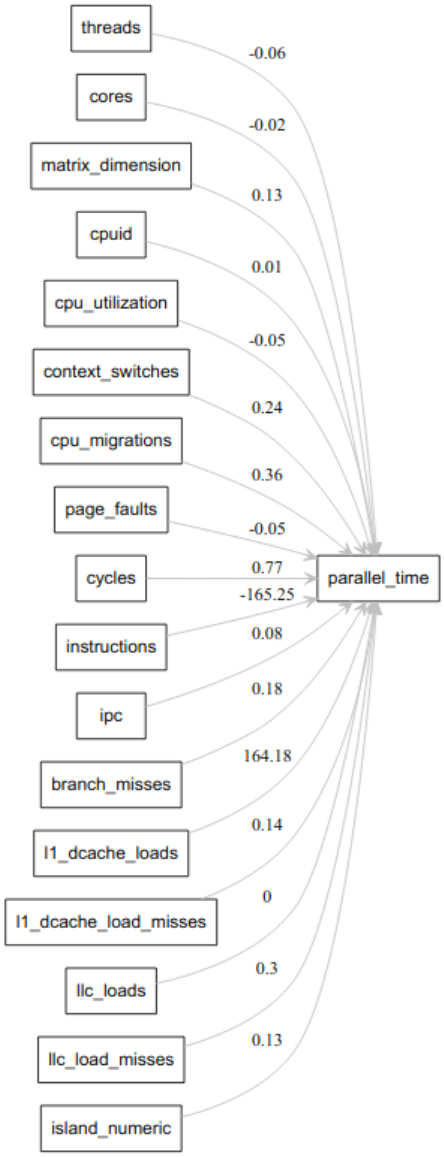


Figure 5.1: SEM Path Analysis Model 1

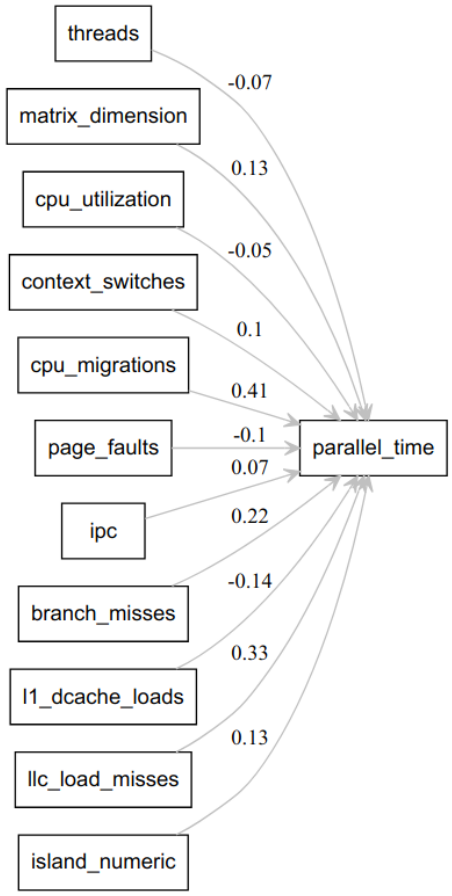


Figure 5.2: SEM Path Analysis Model 4

The results shown in Table 5.1 for each model delineate the regression parameters and the variance with the regression parameters listed in the 1st column and the standardized coefficient *estimates* given in the subsequent columns for each model. The model variance and VIF are shown on the last two rows for each model. The regression parameters, shown in bold, were not significant: `cpuid`, `l1_dcache_load_misses`, and `llc_loads`. For Model 1, three parameters had orders of magnitude higher VIF than other parameters: `instructions` (2.48×10^7), `l1_dcache_loads` (2.49×10^7), and `l1_dcache_load_misses` (3.06×10^3).

The iterative sequence was driven by the necessity to eliminate severe parameter multicollinearity and isolate the statistically significant drivers of execution time. The baseline configuration, Model 1, incorporated all 17 platform attributes, establishing a high explanatory bound ($R^2 = 0.917$) but manifesting catastrophic multicollinearity (VIF $> 2.4 \times 10^7$). This redundancy was explicitly localized between `instructions` (2.48×10^7) and `l1_dcache_loads` (2.49×10^7) parameters. Addressing this instability, Model 2 addressed this primary instability by excising the instructions parameter, which provide fewer microarchitectural insights than cache-based metrics. This single modification collapsed the primary variance inflation and stabilized the standard parameter estimates with negligible loss in explanatory power ($R^2 = 0.916$).

Model 3 removed the `cycles` parameter to eliminate the remaining high-VIF distortions, alongside the highly insignificant `llc_loads` parameter, stabilizing the model variance at $R^2 = 0.913$. This transition revealed a critical hardware-level insight: standard architectural counts, specifically `cores` and `cpuid`, failed to achieve statistical significance ($\alpha = 0.001$). Model 4 achieved structural resolution by eliminating these remaining static hardware parameters, yielding a definitive framework where all retained variables possess absolute statistical significance and strict independence (VIF < 10).

The sequence from Model 1 to Model 4 shows analytically that runtime performance in heterogeneous edge environments is governed by stochastic execution dynamics—such as CPU migrations and L1/LLC load misses—rather than static physical hardware enumerations.

Table 5.1: Orange Pi 5 SEM Path Analysis Across Successive Models

Exogenous Parameter	**Model 1**	**Model 2**	**Model 3**	**Model 4**
threads	-0.062	-0.065	-0.070	-0.073
cores	-0.016	**-0.015**	**-0.004**	–
matrix_dimension	0.127	0.139	0.127	0.126
cpuid	**0.013**	**0.014**	**0.012**	–
cpu_utilization	-0.046	-0.047	-0.053	-0.053
context_switches	0.240	0.223	0.101	0.101
cpu_migrations	0.364	0.374	0.410	0.410
page_faults	-0.052	-0.096	-0.096	-0.096
cycles	0.766	0.692	–	–
instructions	-165.246	–	–	–
ipc	0.082	0.085	0.068	0.069
branch_misses	0.180	0.161	0.225	0.225
l1_dcache_loads	164.175	-1.173	-0.136	-0.135
l1_dcache_load_misses	**0.143**	**0.328**	–	–
llc_loads	**-0.004**	**0.012**	–	–
llc_load_misses	0.299	0.294	0.326	0.326
island_numeric	0.131	0.131	0.139	0.130
Model Variance (R^2)	0.917	0.916	0.913	0.913
Maximum VIF Constraint	$\sim 2.49 \times 10^7$	$\sim 3.03 \times 10^3$	< 270	< 10

Note: Bold values indicate statistically insignificant parameters ($p > 0.001$). Dashes (–) indicate parameter removal.

5.2.1 Parameter Impacts

Given that the standardized coefficients are unitless and all variables have been converted to a standard deviation scale (mean = 0, standard deviation = 1), we can determine the strength and magnitude of the change in the dependent variable by the value of the coefficient, measured in its own standard deviations, assuming all other predictor variables are held constant. This can be used primarily for comparing the relative strength of different paths within the same model. Because the coefficients are unitless, you can compare directly which predictor has the strongest influence on an outcome variable.

Figure 5.3 illustrates the parameter effects more intuitively, with positive impacts improving performance (decreasing execution time) and negative impacts inhibiting performance (increasing execution time). This representation is powerful given that the assessments intrinsically indicate both the magnitude and direction of parameter impacts, which can direct platform optimizations and create informative models articulating platform behavior, thus supporting hypotheses discovery and exploratory analysis. While increasing the number of threads and CPU utilization clearly supports increased performance, notice that page faults in the context of matrix-matrix computations has a larger effect given that page faults serve as a mechanism, leveraging the cooperative nature of shared memory and enabling prefetching to increase data locality effectively for multiple processes. Similarly, `cpu_migrations` having such an outsized negative impact on performance can motivate an investigation into how alternative scheduling mechanisms might be of benefit. SEM is thus an informative and potent tool for optimization discovery and performance articulation.

5.2.2 Extension to Radxa Cubie A7A

Given the value and utility of the SEM path models for informing platform performance and behavior, we wish to investigate the benefit of these specific components, not simply on the Orange Pi 5, but on other devices as well. We are also interested in understanding whether the discovered parameters are important and informative for other platforms. Consequently, we performed a subset of experiments on the Radxa Cubie A7A using the parameters selected from the Orange Pi 5 for the final regression (i.e., Model 4). The regression statistics for both platforms are shown in Table 5.2 with the parameter impacts for the Cubie A7A given

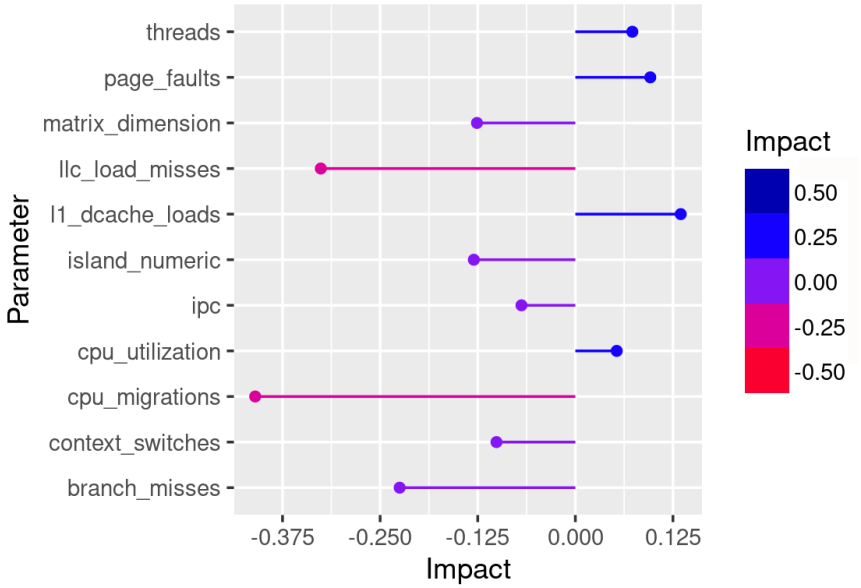


Figure 5.3: Parameter Impacts for Orange Pi 5

in Figure 5.4. Notice that all parameters remain significant when using the same parameters on the Cubie A7A. The model based on these parameters actually improved the R^2, clearly demonstrating the value of these kinds of analysis on heterogeneous performance platforms.

Increasing `threads` reduces execution time, suggesting successful parallelization of the workload. The higher `cpu_utilization` correlates with lower execution time, indicating the system is effectively saturated with productive computational tasks rather than idling. The `l1_dcache_loads` parameter is a strong predictor of increased execution time. High L1 data cache load volume suggests high memory intensity or frequent data access patterns that dominate the execution cycle. Predictably, increasing the problem size (`matrix_dimension`) significantly increases execution time due to increased computational complexity, which is $O(M^3)$. Frequent switching between threads or processes introduces kernel overhead and cache pollution, increasing execution time. Both factors, `cpu_migrations` and `branch_misses` represent pipeline and locality inefficiencies. Moving threads between cores (migrations) destroys cache warmth, while branch mispredictions cause pipeline stalls.

Table 5.2: Comparative Model Regressions and Variances ($R^2_{Orange} = 0.913, R^2_{Cubie} = 0.927$)

	Orange Pi 5		Cubie A7A		
Variable	Estimate	Std.Err	Estimate	Std.Err	P(>—z—)
Regressions:					
`threads`	-0.073	0.004	-0.058	0.002	0.000
`matrix_dimension`	0.126	0.007	0.450	0.004	0.000
`cpu_utilization`	-0.053	0.004	-0.030	0.001	0.000
`context_switches`	0.101	0.007	0.214	0.004	0.000
`cpu_migrations`	0.410	0.009	0.122	0.003	0.000
`page_faults`	-0.096	0.004	-0.013	0.002	0.000
`ipc`	0.069	0.005	0.316	0.005	0.000
`branch_misses`	0.225	0.010	0.121	0.003	0.000
`l1_dcache_loads`	-0.135	0.007	0.484	0.005	0.000
`llc_load_misses`	0.326	0.005	-0.018	0.003	0.000
`island_numeric`	0.130	0.005	0.309	0.003	0.000
Variances:					
`parallel_time`	0.087	0.001	0.073	0.001	0.000

5.3 Performance Modeling as SEM Latent Variables

With the identification of a list of statistically significant parameters which explain platform performance through the lens of parallel execution time, we move on to full SEM modeling using those selected parameters. While there are numerous latent variables that could be used to model platform behavior, notice that all the parameters have an impact on performance. Furthermore, we propose that their impacts logically fall into several categories that make up the latent variable `Total Impacts` along with the associated latent variables `Processor Impact`, `Memory Impact`, and `Platform Impact` measured by the subsequent variables as shown in Table 5.3. It follows that within these given impact categories, the parameters may be either a CB-SEM based *reflection* or a PLS-SEM based *construct* of the latent impacts, which in turn make up the total impact on performance. While this begins as speculation, the SEM models will help to confirm or deny the relationships we posit.

We hypothesize that the workload performance can be modeled in terms of the total impacts which consist of the aggregate of processor, memory, and platform contributions to platform performance. Although run-time scores are often the most sought after metric for performance, we decided to look at additional metrics including `cycles` and `instructions` (as

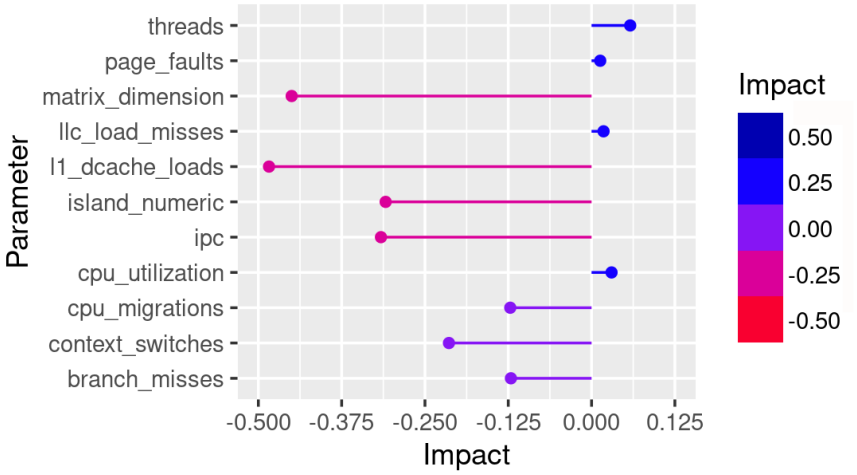


Figure 5.4: Parameter Impacts for Cubie A7A

Table 5.3: Total Impacts

Processor Impact	Memory Impact	Platform Impact
ipc	page_faults	threads
cpu_migrations	branch_misses	island_numeric
cpu_utilization	matrix_dimension	
	context_switches	
	llc_load_misses	
	l1_dcache_loads	

dependent variables) which are intuitive measures of performance behavior, as increases in these variables often positively correlate with longer run-times thereby providing a grounded supplemental performance assessment for our models. When possible, we attempt to bootstrap the models by resampling 1000 times. Where the parameters are significant in the model, that will be indicated by one to three stars ($*$) next to the estimate, with the significance levels ranging from 5% ($p < 0.05$) = *, to 1% ($p < 0.01$) = **, and 0.1% ($p < 0.001$) = ***, for 1, 2, or 3 stars, respectively.

5.3.1 Methodological Synthesis and Model Selection

The primary contribution of this analysis is the empirical elucidation of hardware performance metrics utilizing SEM. Traditional covariance-based (CB-SEM) reflective specifications for both architectures as shown in Figures 5.5-5.8 impose erroneous causality on fundamentally independent low-level architectural counters, such as CPU utilization and matrix dimensions. Out of all the reflective models, only the Orange Pi 5 reflective model, shown in Figure 5.5 yielded an execution time variance explanation of $R^2 = 0.950$. However, the reflective framework suffered from numerous Heywood cases and implausibly high path coefficients, signaling improper convergence. The integration of additional performance variables, specifically `cycles` and `instructions`, caused the performance estimate to explode to an impossible $R^2 = 6.492$ (see Figure 5.7), confirming profound model misspecification.

Conversely, the partial least squares (PLS-SEM) formative architecture, as shown in Figures 5.9-5.12 maps hardware-level execution behaviors to overarching performance degradation, isolating a theoretically robust template for system bottleneck analysis. In the formative model, composite latent variables are denoted via hexagons, and the parameters are the cause of the latent variables. In this framework, parameters act as causal drivers of localized systemic constraints. Variables such as `threads` and `island_numeric` function as direct contributing factors to platform impact.

By utilizing formative precursor pathways while preserving a multidimensional reflective measurement for the endogenous performance construct, the formative template maintains proper measurement soundness. This configuration successfully isolates systemic bottlenecks, providing substantial and plausible explanatory power across distinct hardware platforms with R^2 values of 0.972 (Orange Pi 5) and 0.934 (Cubie A7A). Table 5.4 give the R^2 values for each proposed SEM model investigated.

Table 5.4: Latent Performance Variable R^2 Values Across SEM Models

	1-Variable		**3-Variable**	
	Execution Time		**Time + Cycles + Instructions**	
Platform	**Reflective**	**Formative**	**Reflective**	**Formative**
Orange Pi 5	0.950	0.827	6.492	0.972
Cubie A7A	1.045	0.831	1.055	0.934

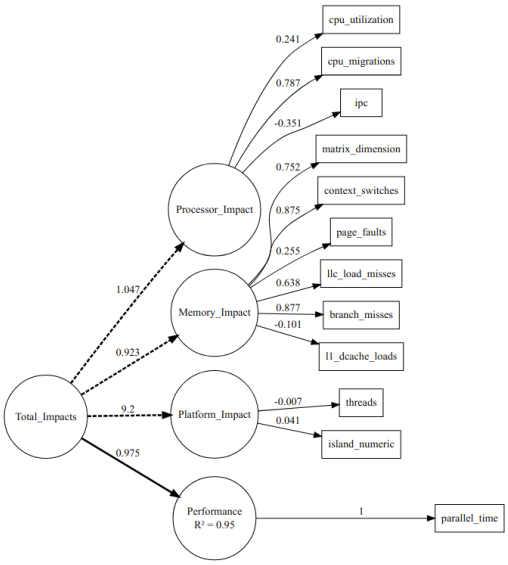


Figure 5.5: Orange Pi SEM Reflective Model: Execution Time

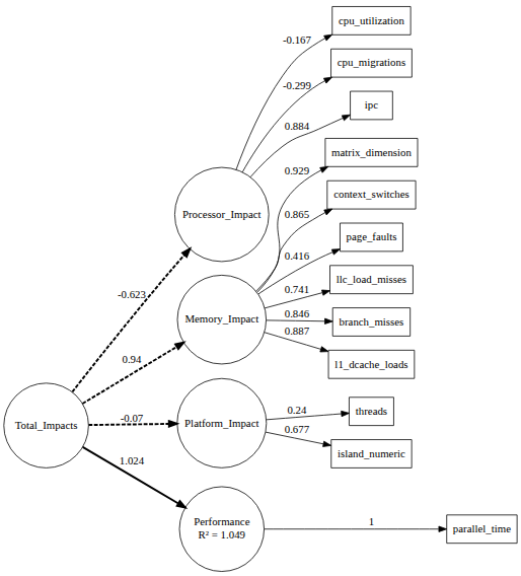


Figure 5.6: Cubie SEM Reflective Model: Execution Time

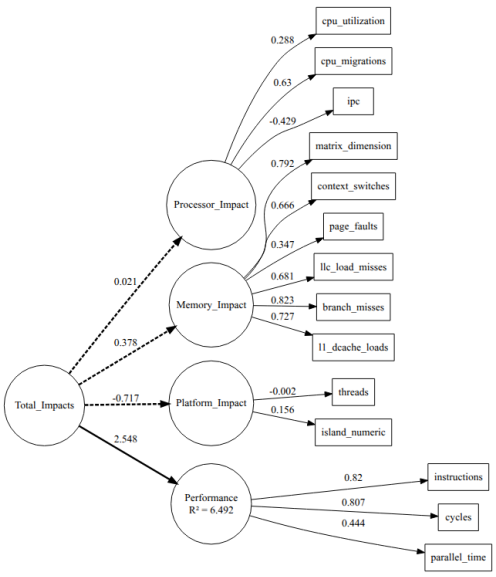


Figure 5.7: Orange Pi SEM Reflective Model: Time + Cycles + Instructions

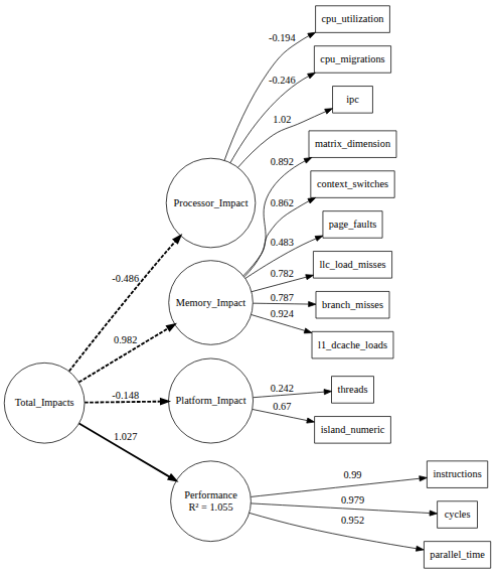


Figure 5.8: Cubie SEM Reflective Model: Time + Cycles + Instructions

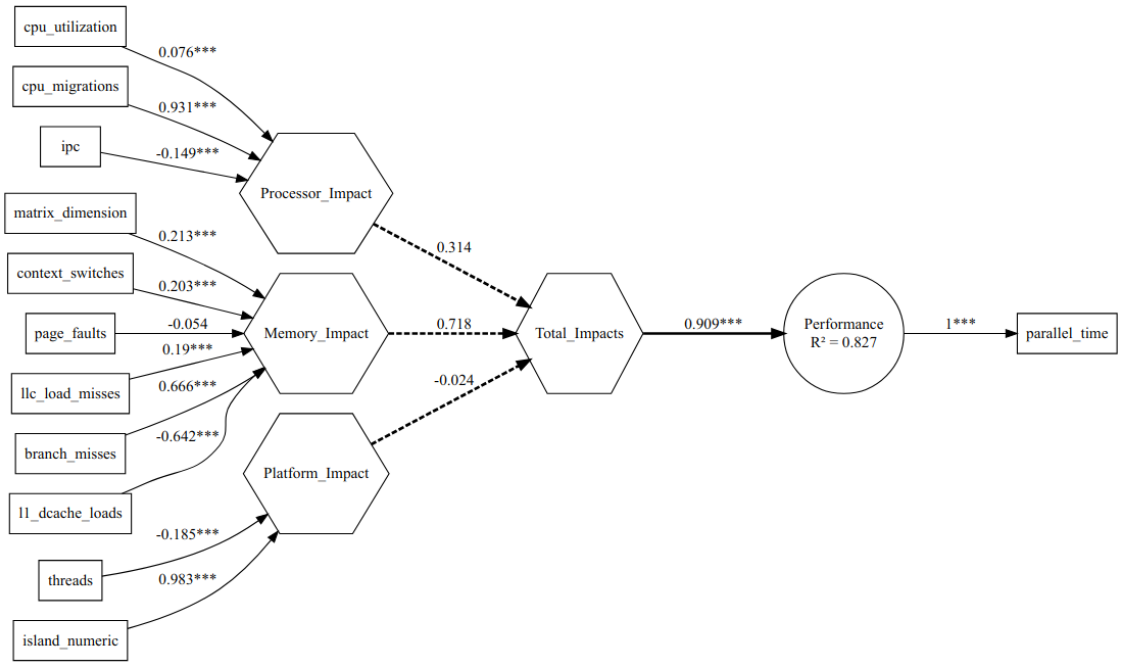


Figure 5.9: Orange Pi SEM Formative Model (Bootstrapped): Execution Time

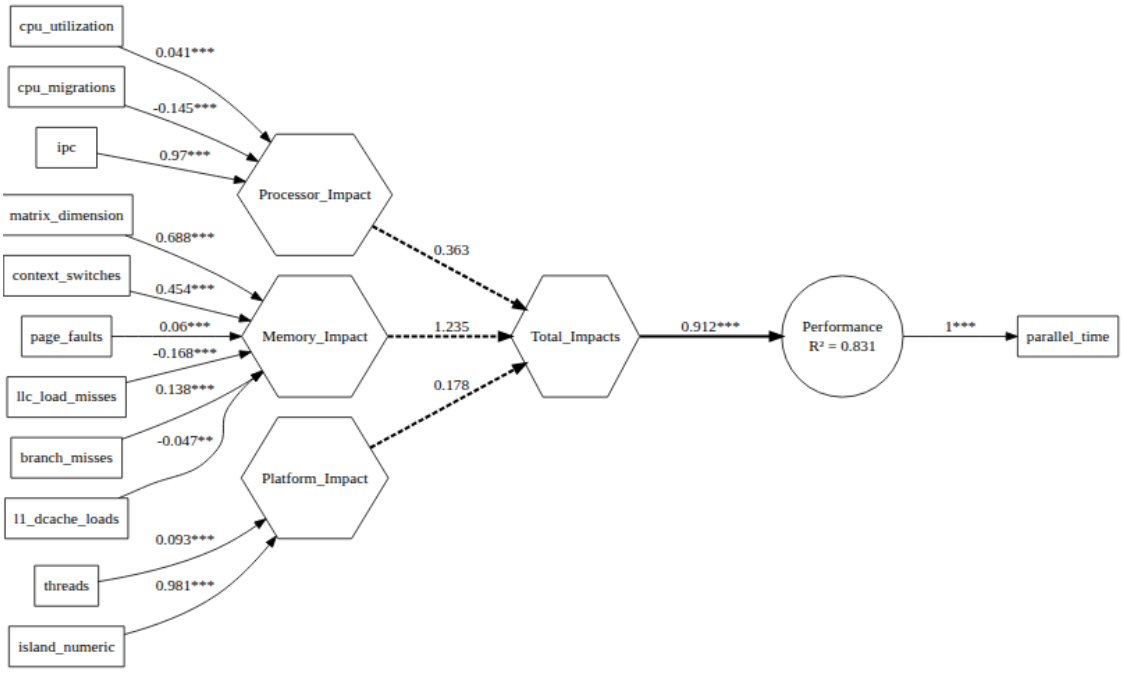


Figure 5.10: Cubie SEM Formative Model (Bootstrapped): Execution Time

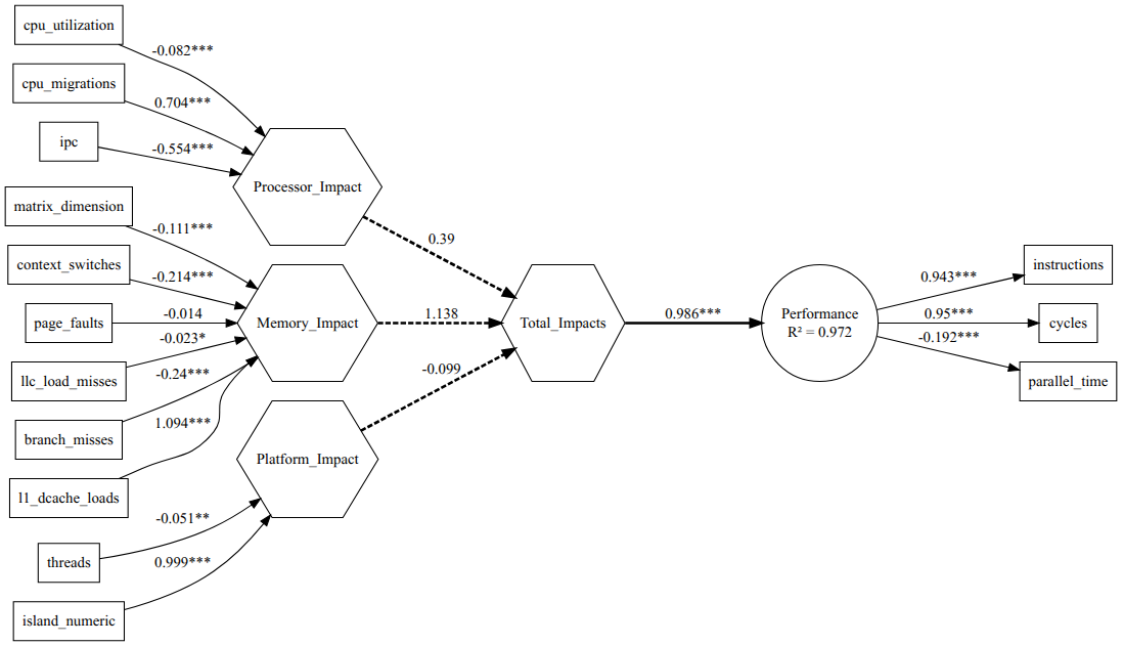


Figure 5.11: Orange Pi SEM Formative Model (Bootstrapped): Execution Time + Cycles + Instructions

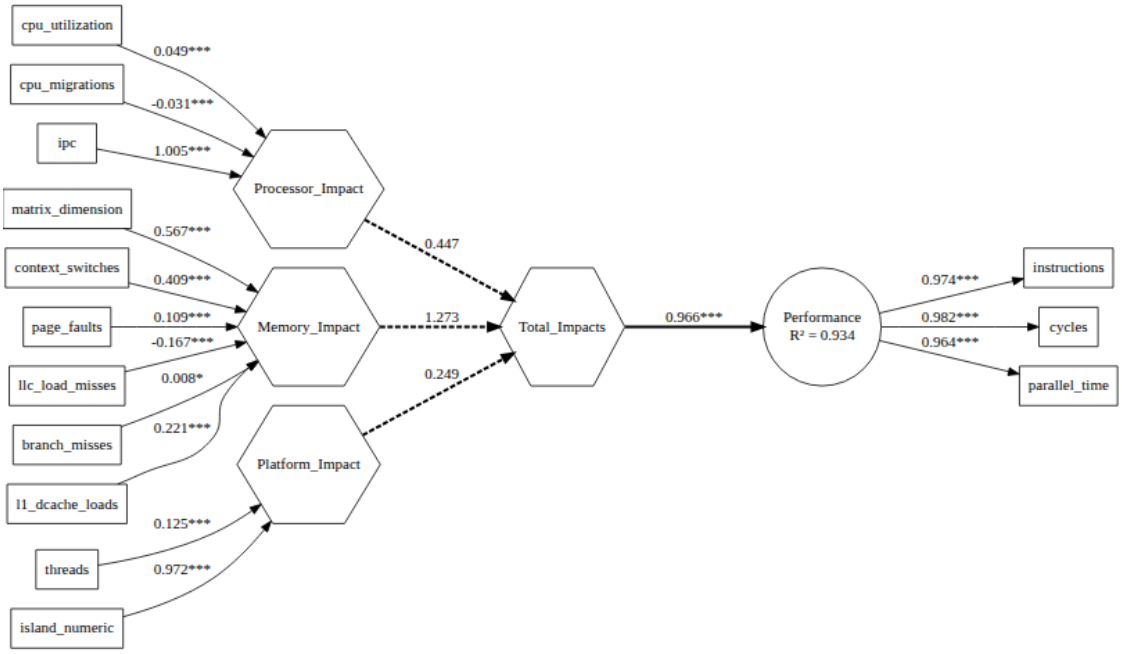


Figure 5.12: Cubie SEM Formative Model (Bootstrapped): Execution Time + Cycles + Instructions

Evaluation of the two formative models demonstrates that a single-indicator performance construct is suboptimal. Although a single-indicator formative model explains parallel execution time with an $R^2 = 0.827$ and $R^2 = 0.831$ (for Orange Pi 5 and Cubie A7A respectively), it discards the multidimensional variance captured by supplementary metrics such as cycles and instructions.

The multi-indicator formative framework is therefore identified as the proper conceptual template. In the case of the Orange Pi 5, subsequent models would remove `page_faults` from the memory impact, given its lack of significance in this model. Consequently on the Cubie A7A, we may have additional layers of impacts given the score (1.235) of memory impacts on total impacts, suggesting either splitting the memory impacts paramters across additional or nested latent variables. Nevertheless, by integrating `parallel_time`, `cycles`, and `instructions`, the model preserves a multidimensional reflective measurement for the endogenous performance construct. This architecture yields strong, statistically significant outer loadings and high explanatory power, achieving $R^2 = 0.972$ for the Orange Pi platform and $R^2 = 0.934$ for the Cubie platform.

5.4 Combining SEM with First Principles Modeling

As we have seen in the previous sections, SEM is a useful tool in helping us understand the relative importance of various parameters in determining overall performance. A limitation of SEM, however, is the fact that it is entirely data driven. The conclusions it draws are based on empirical measurements, effectively ignoring the first principles information of how the system operates as part of model development. In this section, we address this shortcoming by combining fundamental information we know about the application's execution (i.e., its asymptotic execution time) with SEM techniques to explore a unified approach to modeling.

Here, we limit ourselves to input parameters that are explicitly chosen by the user at execution time: `matrix_dimension`, `cores`, and `island`. We first explore the polynomial impact of `matrix_dimension`, and then investigate the impacts of multiple parameter interactions (a particular strength of SEM techniques).

5.4.1 Polynomial Performance Modeling & Analysis

First principles tell us the standard 3-loop GEMM computation runs in $O(M^3)$ time, and we traditionally model performance based solely on this term. If all machines had exactly the same performance for a GEMM workload, this notion of modeling with the single term makes sense as the result would be consistent across platforms. Yet, we suspect this to be untrue, as different devices have different architectural and/or programmatic advantages enabling different terms to vary from platform to platform. Other terms may be required in the consideration of GEMM performance on a platform, so we consider regressions of parallel execution time on matrix dimension ($\texttt{parallel_time} \sim M^j$) to understand their importance.

This section details our experimental process and results for modeling the relationship between the single predictor variable, `matrix_dimension` (M), and the single outcome variable, `parallel_time` (Y), using polynomial regression (e.g., $Y \sim f(M) + \epsilon$). The primary objective is to identify non-linearity and determine if it is significant enough to warrant additional terms for accurate functional model representation.

We start with an initial model which predicts the outcome variable `parallel_time` based on a centered linear term ($\texttt{M}_c$), a squared term ($\texttt{M}_c^2$), and a cubed term ($\texttt{M}_c^3$) of a predictor variable ($\texttt{M}_c$). The model formula is as follows:

$$\text{parallel_time} \sim \beta_0 + \beta_1 \texttt{M}_c + \beta_2 \texttt{M}_c^2 + \beta_3 \texttt{M}_c^3 + \epsilon.$$

Our investigation seeks to determine which terms in the polynomial are significant and should be included for the purposes of modeling.

Experimental Steps and Rationale

The analysis follows a four-step methodology, progressing from model preprocessing to formal statistical comparison in order to identify the significant polynomial terms. Those steps are as follows:

Variable Creation and Centering. To capture the features of the data, polynomial regression mandates **centering** the predictor variable ($\mathtt{M}_c = M - \bar{M}$) prior to creating polynomial terms ($\mathtt{M}_c^2, \mathtt{M}_c^3$). This step is used to mitigate **multicolinearity** between the linear higher-order terms, thereby ensuring more stable and interpretable regression coefficients.

Full Model Estimation. The regression models in our analysis use different combinations of matrix dimension terms to assess the individual significance of the proposed polynomial effects (linear, quadratic, and cubic). Using those findings we are able to identify significant terms critical for performance modeling for both the Orange Pi and the Cubie A7A platforms.

Model Visualization. Diagnostic plots for both platforms, overlayed with a cubic fitted curve, are shown in Figure 5.13 and Figure 5.14. These visualizations provide confirmation of the complex, non-linear curvature accurately captured by the polynomial terms. A nested model comparison using Information Criteria (IC) favored the full cubic specification, indicating it exhibited the lowest AIC and BIC values (see description below).

Formal Model Comparison. Finally, to justify the inclusion of any terms, a nested model comparison was performed using linear model (lm) regression and the analysis of variance (ANOVA) on the data. We used the model fit Information Criteria (IC) to discover the full model. Consequently, the *AIC/BIC* results indicate that the full cubic model exhibited the **lowest AIC and BIC** values, favoring the most complex specification.

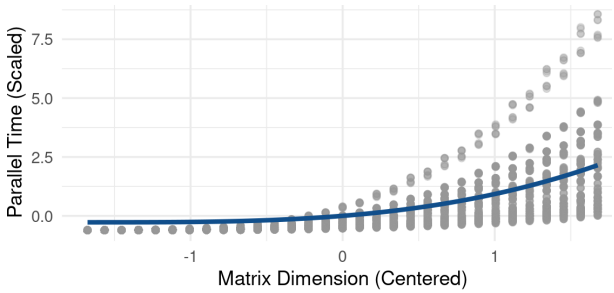


Figure 5.13: Orange Pi Model Visualization

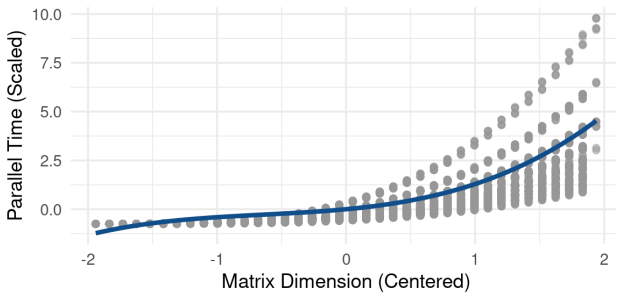


Figure 5.14: Cubie Model Visualization

Akaike and Bayesian Information Criteria: AIC & BIC

Information criteria are statistical tools used for **model selection**. They provide a quantitative means to compare different mathematical models applied to the same dataset, balancing the accuracy of the fit against the complexity of the model [113].

The AIC formula is:

$$AIC = 2k - 2\ln(\hat{L})$$

where k is the number of estimated parameters (complexity) and $\hat{L}$ is the maximum likelihood of the model (goodness of fit). The AIC estimates the relative amount of information lost by a given model. The goal is to select a model that describes the data well without unnecessary complexity.

The **BIC**, derived from Bayesian probability, serves a similar purpose to AIC but introduces a stronger penalty for models with more parameters, especially as the sample size increases. The BIC formula is:

$$BIC = k\ln(n) - 2\ln(\hat{L})$$

where n is the number of observations (sample size) and $k\ln(n)$ is the complexity penalty, which scales with data volume.

The "lower is better" rule is a direct result of the mathematical structure of these equations, which serve as loss functions. Both expressions contain the term $-2\ln(\hat{L})$.

- Likelihood ($\hat{L}$) measures how well the model explains the observed data. A "perfect" fit has a high likelihood.
- The negative sign means that as the model's fit improves (higher $\hat{L}$), the overall score **decreases**.

Adding more variables (increasing k) will almost always improve the fit ($\hat{L}$), but it risks **overfitting** which captures noise rather than the underlying signal.

- The AIC and BIC terms act as a "cost" or "tax" on complexity.

- To achieve a lower total score, a model must improve the fit significantly enough to outweigh the mathematical cost of adding new parameters.

A lower *AIC* or *BIC* indicates a model that achieves the highest possible likelihood with the fewest possible parameters.

Polynomial Model Results

The independent variable, matrix dimension ($\texttt{M}_c$), was centered, and polynomial terms were generated ($\texttt{M}_c^2$, $\texttt{M}_c^3$). The dependent variable is `parallel_time`. The models are given in in Table 5.5.

Table 5.5: Tested Polynomial Regression Models

Model	Description	Equation
1	Linear	$\texttt{parallel_time} \sim \texttt{M}_c$
2	Quadratic	$\texttt{parallel_time} \sim \texttt{M}_c + \texttt{M}_c^2$
3	Cubic	$\texttt{parallel_time} \sim \texttt{M}_c + \texttt{M}_c^2 + \texttt{M}_c^3$

All model comparisons relied on the standard *lm* function for reliable *ANOVA* tests. The information criteria (*AIC* and *BIC*) provide a measure of relative fit, where **lower values indicate a better model**. Those results are given in Table 5.6, with the estimates for the regression parameters given in Table 5.7. The results show clearly that all three predictors are **highly significant**.

Table 5.6: Model Fit Information Criteria Comparison: Orange Pi 5 vs. Cubie A7A

		Orange Pi 5		**Cubie A7A**	
Model	**Description**	*AIC*	*BIC*	*AIC*	*BIC*
1	Linear	17889	17903	80757	80774
2	Linear + Quadratic	16589	16610	55114	55140
3	Linear + Quadratic + Cubic	**16541**	**16569**	**53266**	**53301**

Given that Model 3 has been determined to be the most informative for the prediction of parallel execution time of the workload on both platforms, we decided to look at the model regression to understand the statistical significance and importance of the parameters. The results of the full model regression are shown in Table 5.7.

Table 5.7: Comparative Regression and Variance Results ($R^2_{Orange} = 0.514, R^2_{Cubie} = 0.793$): Estimate, Standard Error, and Significance

	Estimate		**Std. Error**		$P(>\|z\|)$
Variable	**Orange Pi 5**	**Cubie A7A**	**Orange Pi 5**	**Cubie A7A**	**All Models**
Regressions:					
$\mathtt{M}_c$	0.521	0.623	0.020	0.005	0.000
$\mathtt{M}_c^2$	0.334	0.401	0.009	0.002	0.000
$\mathtt{M}_c^3$	0.072	0.094	0.010	0.002	0.000
Variances:					
`parallel_time`	0.486	0.207	0.008	0.001	0.000

The results indicate that all of the matrix terms are significant, which warrants their inclusion in the full model. Given that the standardized estimate coefficients are unitless and all variables have been converted to a standard deviation scale (mean = 0, standard deviation = 1), we can determine the strength and magnitude of the change in the dependent variable by the value of the coefficient, measured in its own standard deviations, assuming all other predictor variables are held constant. This can be used primarily for comparing the relative strength of different paths within the same model. Because the coefficients are unitless, a direct identification of the predictor with the strongest influence on an outcome variable can be made.

The parameters represent both linear and non-linear effects that have a statistically significant impact on parallel execution time. We will start with the Orange Pi 5, followed by the Cubie A7A, in order to outline the parameters and their effects on the outcome variable (parallel execution time) as follows:

Linear Effect:

LINEAR TERM: $\mathtt{M}_c$ is statistically significant
($p < 0.001$, Estimate(Orange Pi, Cubie) $= 0.521, 0.623$). This indicates a strong, positive **linear** association: as the centered matrix dimension increases, parallel time increases, independent of the non-linear curvature.

Non-Linear Effects:

QUADRATIC TERM: $\mathtt{M}_c^2$ is highly significant
($p < 0.001$, Estimate(Orange Pi, Cubie) $= 0.334, 0.401$). This indicates a significant non-linear effect where the rate of increase in parallel time accelerates as the matrix dimension grows.

CUBIC TERM: $\mathtt{M}_c^3$ is also highly significant
($p < 0.001$, Estimate(Orange Pi, Cubie) $= 0.072, 0.094$). This indicates that a more complex, S-shaped curvature or an inflection point is necessary to fully model the relationship.

The relationship between matrix dimension and parallel time is unquestionably non-linear, and all three polynomial terms contribute significantly to predicting the outcome.

5.4.2 Multi-Parameter Interaction Modeling

We next focus on models that incorporate interactions among multiple parameters. To quantify the performance overhead and efficiency gains using statistical modeling of heterogeneous computing architectures (ARM big.LITTLE), we take the standard approach of assigning the Big cores as the *dummy variable* (or reference group) while explicitly modeling the Little core parameter in equation models to support enhanced measurements.

Modeling with the Little island as the active (dependent) variable allows for direct measurement of power-to-performance scaling. The Little cores being designed for deterministic, low-power background tasks, have coefficients that determine the "threshold of utility:" that is, the point at which the task becomes too intensive or too expensive for the Little island and requires preemptive migration to the Big island.

Interaction Effects and Thermal Throttling

Given that we are exploring new models with unknown features, interactions, and parameters, under this scenario the Little cores are often the variable of interest given that their performance is more sensitive to resource contention and shared cache (L2/L3) constraints. Models based on the Little island can be used to track:

Context Switching Overhead: Determines the cost of moving a thread from the reference (Big) to the modeled (Little) core.

DVFS (Dynamic Voltage and Frequency Scaling) Sensitivity: Little cores often have narrower frequency bands. Modeling them explicitly captures the precise saturation rate compared to the Big core baseline.

Mathematical Identification

Using the Big cores as the reference can simplify speedup factor interpretation. When the variable of interest is a metric like parallel execution time (`parallel_time`), then a positive coefficient for Little-based measurements will immediately identify increased latency (performance degradation) as opposed to a negative coefficient indicative of decreased latency (performance improvement).

Table 5.8: Notation and Description

Parameter	Description
$\mathtt{M}_c$	Matrix Dimension Centered
$\mathtt{M}_c^2$	Matrix Dimension Centered Squared
$\mathtt{M}_c^3$	Matrix Dimension Centered Cubic
`island`	Compute Island *(Baseline: Big Island)*
`cores`	Compute Unit *(Baseline: 1 Core)*
$\mathtt{M}_c \times$ `island`	Linear **x** Island *(Interaction)*
$\mathtt{M}_c^2 \times$ `island`	Quadratic **x** Island
$\mathtt{M}_c^3 \times$ `island`	Cubic **x** Island
$\mathtt{M}_c \times$ `cores`	Linear **x** Core
$\mathtt{M}_c^2 \times$ `cores`	Quadratic **x** Core
$\mathtt{M}_c^3 \times$ `cores`	Cubic **x** Core
$\mathtt{M}_c \times$ `island` $\times$ `cores`	Linear **x** Island **x** Core
$\mathtt{M}_c^2 \times$ `island` $\times$ `cores`	Quadratic **x** Island **x** Cores
$\mathtt{M}_c^3 \times$ `island` $\times$ `cores`	Cubic **x** Island **x** Core

With the parameters fully identified for the GEMM workload operation, we next explore the individual impacts of the most prominent and obvious parameters common to both platforms: *matrix dimension*, *cores*, and *island*. Using the additional parameters, we derive our first SEM based model, with the notation given in Table 5.8, and the parameter interactions denoted with an '**x**' between parameters. The model depicts the relationship between the independent variables (platform parameters) and the dependent variable (`parallel_time`).

This SEM path model enables the rapid interpretation of parameter effects. The estimates given in Table 5.9 are the standardized path coefficients, also known as beta coefficients (e.g., β_i). For example, the coefficient for $\mathtt{M}_c$ (i.e., 0.940) indicates that a one-standard-deviation increase in this exogenous (independent) variable corresponds to a 0.940 standard-deviation *increase* of the endogenous (dependent) variable `parallel_time`. The `cores` parameters follows a similar pattern but has a marked *decrease* in parallel execution time. The signs of the estimates are indicative of the direction of change with respect to the parallel execution time.

This presentation modality allows the parameter effects to be read easily from the model estimates, to understand the endogenous variables impacts directly. Other interesting insights gained from the parameter estimates can be seen with the values for both $\mathtt{island}_L$ (0.137) and its interaction with matrix terms (e.g., $\mathtt{M}_c \times \mathtt{island}_L$) which all *increase* parallel execution time. The behavior has been determined numerically and via rigorous statistical analysis as opposed to direct architectural insights about the platform known prior to analysis. This is a potent and powerful tool as it can rightfully identify parameters and their contributions (whether positive or negative) to platform performance. This investigation revealed interesting findings for both platforms as follows:

ORANGE PI 5: The final 23-parameter model achieves $R^2 = 0.937$, increasing the explained variance of the 3-parameter ($\mathtt{M}_c, \mathtt{M}_c^2, \mathtt{M}_c^3$) model ($R^2 = 0.514$) by over 80%. Standardized, unitless coefficients permit direct comparison of predictor magnitude and direction on parallel execution time. Matrix dimension estimates increased relative to initial models; the linear term ($\mathtt{M}_c = 0.940$) demonstrates the strongest impact, supported by significant secondary ($\mathtt{M}_c^2 = 0.511$) and tertiary ($\mathtt{M}_c^3 = 0.208$) terms. The complete Orange Pi 5 model is detailed in Table 5.9.

CUBIE A7A: Given the value of the Orange Pi 5 model, we determined to run it with the same parameters for the Cubie A7A and found a similiar high-significance

Table 5.9: Comparative Regression Models: Orange Pi ($R^2 = 0.937$) vs. Cubie A7A ($R^2 = 0.950$)

	Orange Pi		**Cubie A7A**	
Regression Parameter (parallel_time $\sim$)	**Est.**	$P(>\|z\|)$	**Est.**	$P(>\|z\|)$
$\texttt{M}_c$	0.940	0.000	0.674	0.000
$\texttt{M}_c^2$	0.511	0.000	0.442	0.000
$\texttt{M}_c^3$	0.208	0.000	0.206	0.000
$\texttt{island}_L$	0.137	0.000	0.070	0.000
$\texttt{cores}_2$	−0.150	0.000	−0.065	0.000
$\texttt{cores}_3$	−0.190	0.000	−0.311	0.000
$\texttt{cores}_4$	−0.167	0.000	−0.366	0.000
Two-Way Interactions				
$\texttt{M}_c \times \texttt{island}_L$	0.350	0.000	1.111	0.000
$\texttt{M}_c^2 \times \texttt{island}_L$	0.275	0.000	0.747	0.000
$\texttt{M}_c^3 \times \texttt{island}_L$	0.110	0.000	0.353	0.000
$\texttt{M}_c \times \texttt{cores}_2$	−0.358	0.000	−0.117	0.000
$\texttt{M}_c \times \texttt{cores}_3$	−0.491	0.000	−0.657	0.000
$\texttt{M}_c \times \texttt{cores}_4$	−0.358	0.000	−0.837	0.000
$\texttt{M}_c^2 \times \texttt{cores}_2$	−0.268	0.000	−0.096	0.000
$\texttt{M}_c^2 \times \texttt{cores}_3$	−0.349	0.000	−0.527	0.000
$\texttt{M}_c^2 \times \texttt{cores}_4$	−0.278	0.000	−0.659	0.000
$\texttt{M}_c^3 \times \texttt{cores}_2$	−0.080	0.000	**-0.032**	**0.079**
$\texttt{M}_c^3 \times \texttt{cores}_3$	−0.106	0.000	−0.195	0.000
$\texttt{M}_c^3 \times \texttt{cores}_4$	−0.079	0.000	−0.253	0.000
$\texttt{island}_L \times \texttt{cores}_2$	0.029	0.000	−0.101	0.000
Three-Way Interactions				
$\texttt{M}_c \times \texttt{island}_L \times \texttt{cores}_2$	0.061	0.000	−0.185	0.000
$\texttt{M}_c^2 \times \texttt{island}_L \times \texttt{cores}_2$	0.053	0.000	−0.160	0.000
$\texttt{M}_c^3 \times \texttt{island}_L \times \texttt{cores}_2$	**0.018**	**0.115**	-0.057	0.003
Variances (parallel_time)	0.063	0.000	0.050	0.000

Note: Bold values indicate statistically insignificant parameters ($p > 0.005$).

model. This suggests that additional parameters improve prediction of the parallel execution time across the various configurations which was support by the Cubie A7A results with an excellent $R^2 = 0.950$. Results for the Cubie A7A are given in Table 5.9 side-by-side with the Orange Pi5.

5.4.3 Performance Articulation and Results Synthesis

This study utilizes structural equation modeling to deconstruct performance variance in General Matrix Multiplication (GEMM) across two distinct ARMv8 heterogeneous architectures: a balanced 4+4 big.LITTLE configuration (Orange Pi 5) and an asymmetric 2+6 configuration (Cubie A7A). By regressing execution time against polynomial workload complexity ($\mathtt{M}_c, \mathtt{M}_c^2, \mathtt{M}_c^3$) and core-island interactions, we identify a critical *asymmetry inversion point*. While the 2+6 topology offers superior area-efficiency for low-order workloads, it exhibits a statistically significant "congestion collapse" in high-order cubic scaling ($p < 0.01$). We demonstrate that the $\mathtt{M}_c^3 \times \mathtt{island}_L \times \mathtt{cores}_P$ interaction term serves as a proxy for cache-coherency traffic and memory bus contention, revealing that the 6-core Little cluster in the Cubie A7A suffers from an accelerated transition to memory-boundedness compared to the 4-core cluster of the Orange Pi 5.

Heterogeneous System-on-Chip (SoC) designs in our analysis have bifurcated into two primary philosophies: balanced performance (e.g., Rockchip RK3588's 4+4) and throughput-optimized asymmetry (e.g., Allwinner A733 2+6 configurations). While a 2+6 topology maximizes multi-threaded efficiency for background tasks, the behavior under high-intensity arithmetic kernels like GEMM remains under-theorized.

The congestion collapse observed on the Cubie A7A's 2+6 split can be interpreted formally through Little's Law ($L = \lambda W$), where L represents the number of pending memory requests, λ is the arrival rate of cache misses, and W is the residency time in the interconnect. In balanced topologies (4+4), λ is distributed such that L remains below the saturation threshold of the 3MB L3 cache. However, on the Cubie, the addition of six Little cores increases λ beyond the capacity of the 1 MB LLC, causing W to grow non-linearly. This creates a stochastic queueing bottleneck where the Big cores are starved of bandwidth by the very resources (Little cores) meant to enhance performance. Our SEM path coefficients for LLC load misses effectively capture this load-dependent server latency, indicating that raw

core totals are insufficient to predict performance in the presence of interconnect structural saturation. We highlight a few of the important findings below:

The Stochastic Inversion Point: One key finding is the significance of the triple interaction term $\mathtt{M}_c^3 \times \mathtt{island}_L \times \mathtt{cores}_2$.

ORANGE PI 5 ($p = 0.115$): The insignificance suggests a "symmetric memory-compute ratio" where complexity scales independently of core count on the Little island.

CUBIE A7A ($p = 0.003$): The high significance identifies a "non-linear coupling." On the Cubie, as matrix size and core count increase, performance degrades stochastically as a function of cluster contention rather than linearly.

Consequently, asymmetry breeds non-determinism. The 4+4 setup allows for decoupled scheduling heuristics, whereas the asymmetric 2+6 architecture needs a multivariate-aware scheduler.

Island Penalty and Scaling Sensitivity: The parameter $\mathtt{island}_L$ represents the base latency penalty for offloading tasks to the energy-efficient cores. With respect to the Cubie A7A, the base estimate is lower (0.070) than the Orange Pi 5 (0.137). However, the interaction term $\mathtt{M}_c \times \mathtt{island}_L$ reveals a strong divergence:

ORANGE PI 5: 0.350

CUBIE A7A: 1.111

The 3.17× higher sensitivity in the Cubie A7A indicates that the 6-core Little island hits a structural performance wall. This is a manifestation of architectural divergence, where the 6-core cluster likely shares a narrow path to the L3 cache, creating a "funnel effect."

These results lead us to develop new performance indicators which we will describe directly.

5.5 New Performance Indicators

In this section, we define several new metrics that are derived from the SEM estimates articulated previously. Their values are summarized in Table 5.10 and each is discussed in turn.

Table 5.10: Architectural Metrics Derived from SEM Estimates

Metric	Orange Pi 5	Cubie A7A	Ratio (C/O)
Complexity Sensitivity ($\sum \beta \mathtt{M}_c^i$)	1.659	1.322	0.797
Inter-Island Scaling Friction ($\mathtt{M}_c \times I_L$)	0.350	1.111	3.174
Multi-Core Saturation ($\Delta\beta C_3 \rightarrow C_4$)	+0.023	-0.055	-0.418
Stochastic Coupling Significance (p value)	0.115	0.003	0.026
Critical Complexity Index (χ)	1.417	2.852	2.013

5.5.1 Complexity Sensitivity ($\sum \beta_i \mathtt{M}_c^j$)

This measures the system's performance volatility as task complexity increases.

VARIABLES: β_i represents the standardized path coefficient (weights), and $\mathtt{M}_c^j$ is the complexity level of task j.

ANALYSIS: The Orange Pi 5 (1.659) is more sensitive to complexity than the Cubie (1.322). While more powerful, the Orange Pi's performance fluctuates more based on the specific type of instruction set being processed.

5.5.2 Inter-Island Scaling Friction ($\mathtt{M}_c \times I_L$)

This calculates the overhead penalty incurred when data travels between different core clusters (e.g., big.LITTLE architecture).

VARIABLES: $\mathtt{M}_c$ is the computational load; I_L is the inter-island latency.

ANALYSIS: The Cubie A7A (1.111) experiences over 3.17× more friction than the Orange Pi (0.350). This indicates a significant communication bottleneck in the Cubie's internal bus or interconnect.

5.5.3 Multi-Core Saturation ($\Delta\beta_i C_3 \to C_4$)

This reflects the marginal utility of adding a fourth core to a parallel task.

VARIABLES: $\Delta\beta$ is the change in the performance coefficient during the transition from 3rd to 4th core ($C_3 \to C_4$).

ANALYSIS: The Orange Pi displays **saturation** (+0.023), given that the overhead of managing an additional 4th core actually may degrade performance due to competition for resources, whereas the Cubie remains efficient (-0.055).

5.5.4 Stochastic Coupling Significance (p value)

A statistical check determining if the relationship between architectural "noise" and performance is reliable.

ANALYSIS: At the significance threshold ($\alpha = 0.005$), the Orange Pi ($p = 0.115$) is not statistically significant, meaning its performance is relatively independent of random noise. In contrast, the Cubie ($p = 0.003$) is highly significant, suggesting its performance is tightly (and potentially negatively) coupled with architectural fluctuations.

5.5.5 Critical Complexity Index (χ)

We hypothesize that adding more Little cores (6 vs. 4) may paradoxically reduce system efficiency particularly for high-matrix workloads due to coherency traffic overwhelming the shared interconnect. We propose a *critical complexity index* (χ) to validate this hypothesis:

$$\chi = \frac{\sum |\beta_{\text{Interaction}}|}{\sum |\beta_{\text{Direct}}|} \tag{5.1}$$

VARIABLES: χ enables a scheduler to identify the point where the overhead of the Little island exceeds its computational contribution.

ANALYSIS: A $\chi > 2.0$ suggests "interaction bound" rather than "compute bound" architecture for complex workloads ($\texttt{M}_c$). Notice the operation threshold: particularly, when χ significantly exceeds 1.0, the Little cores ($\texttt{island}_L$) may act as net liabilities for complex tasks. On the Cubie A7A, the high χ dictates that the scheduler should force-idle the 6 Little cores for large $\texttt{M}_c$ to prevent "poisoning" the memory bus for the 2 Big cores. The *efficiency implications* of the high χ on Cubie validates our hypothesis.

The interaction overhead as measured by this new metric, the critical complexity index χ, is as follows:

$$\chi_{OrangePi} \approx 1.42 \quad \chi_{Cubie} \approx 2.85$$

The χ_{Cubie} magnitude for interaction overhead is more than double its direct computational contribution compared to the Orange Pi. This indicates extreme architectural fragility where coordination costs dominate execution time. The approximate 2× difference in the ratio disparity between the Cubie (2+6) configuration is revealed by the worse non-linear scaling than the Orange Pi (4+4) configuration. In this instance, a scheduler should prioritize isolation (using only Big cores) over parallelism (using all cores) to avoid bus poisoning.

5.6 Chapter Conclusions

Chapter 5 advances the foundational asymmetric multiprocessing (AMP) models established in Chapter 4 by transitioning from isolated resource assumptions on balanced hardware to dynamic, contention-aware multi-parameter modeling on highly asymmetric topologies. While Chapter 4 established the baseline functional relationship $\vec{P} = F(\vec{A}, \vec{H}, \vec{S})$ on a balanced $4+4$ Rockchip RK3588S architecture (Orange Pi 5), Chapter 5 resolves the stochastic

non-linearities and interconnect bottlenecks exposed on highly asymmetric configurations, specifically the $2 + 6$ Allwinner A733 architecture (Radxa Cubie A7A).

Table 5.11: Methodological Evolution: Chapter 4 to Chapter 5 Model

Model Element	**Chapter 4 (Foundational)**	**Chapter 5 (Advanced)**
Hardware Topology	Balanced $4 + 4$ configuration (Orange Pi 5)	Asymmetric $2+6$ configuration (Radxa Cubie A7A)
Conceptual Framework	Abstract independent scaling vectors ($\vec{A}$, $\vec{H}$, $\vec{S}$)	Tripartite SAC operator chain ($\mathcal{G} = \mathcal{C}_h \circ \mathcal{A}_h \circ \mathcal{S}_h$)
SEM Application	Validates first-principles parameter choices	Drives causal multi-indicator formative bottleneck analysis
Resource Assumption	Isolated resource consumption and static scaling	Dynamic multi-parameter interactions and structural contention

5.6.1 Formative vs. Reflective SEM Specification

This chapter has established that, depending on the modeled parameters, using traditional reflective covariance-based Structural Equation Modeling (CB-SEM) alone may impose erroneous causality on independent architectural counters, when treating observed metrics as interchangeable reflections of unobserved constructs. Reflective specifications on complex hardware and workload parameter interactions can yield Heywood cases, implausibly high path coefficients, and model divergence (e.g., R^2 explosions exceeding 6.49). Therefore, one must be cognizant of the relationships between parameters to the extent that they can identify whether such parameters are indicators or causes of underlying performance phenomenon. Once the parameter relationships can be ascertained, both SEM techniques offer a powerful double-edged sword enabling models to cut through confounding interactions that may arise when defaulting to one method or the other.

As we identified that these modeling parameters were in fact the causes of performance behavior, we implement multi-indicator formative Partial Least Squares SEM (PLS-SEM), where observed parameters act as direct causal drivers of localized system bottlenecks. This formulation achieves robust explanatory power ($R^2 = 0.972$ for Orange Pi 5 and $R^2 = 0.934$ for Cubie A7A) by modeling processor, memory, and platform impacts as composite formative constructs.

5.6.2 Structural Bottlenecks, Inter-Island Friction, and the Critical Complexity Index

While applicable to both platforms, on highly asymmetric architectures (2+6), the disparity between compute capacity and interconnect bandwidth generates the "funnel effect" much earlier at the L2-to-L3 transition, leading to congestion collapse. This chapter defines the Critical Complexity Index (χ) to isolate mathematically the threshold where inter-island friction neutralizes parallel advantages wherein a threshold of $\chi > 2.0$ identifies interaction-bound behavior where energy-efficient Little cores operate as a net liability by generating excessive cache-coherency traffic that poisons the memory bus for Big cores. Specifically, the three-way interaction term ($M_c^3 \times \text{island}_L \times \text{cores}$) mathematically proxies this coherency saturation. While the balanced $4 + 4$ Orange Pi 5 exhibits a stable index ($\chi \approx 1.42$), the asymmetric $2 + 6$ Cubie A7A reaches $\chi \approx 2.85$, confirming extreme architectural fragility where coordination costs dominate execution time and necessitate core-isolation scheduling policies.

This chapter advances a Structural Equation Modeling (SEM) framework that extracts directional component interactions and isolates critical performance bottlenecks across heterogeneous architectures, bypassing the limitations of traditional *a priori* heuristics. By shifting from heuristic profiling to data-driven path analysis, the methodology quantifies continuous metrics such as execution time directly from hardware telemetry, synthesizing analytical first-principles with empirical validation.

The primary architectural contribution is the deployment of a multi-indicator formative SEM framework. Conventional reflective specifications impose invalid causal assumptions on orthogonal hardware counters (e.g., `cpu_utilization`, `matrix_dimension`). The formative approach correctly models these metrics as direct causal drivers of system degradation. Retaining a multidimensional reflective specification for the endogenous performance construct maps low-level execution states to macroscopic bottlenecks, achieving robust explanatory power ($R^2 = 0.934$).

Asymptotic bounds fail to capture operational realities across practical input ranges. Empirical evaluation of matrix-matrix multiplication—a foundational AI/ML workload—proves that linear, quadratic, and cubic terms remain statistically significant across standard operational scales, invalidating the common assumption that higher-order terms dominate only

in the asymptotic limit. Furthermore, treating model parameters as mutually independent obscures critical interdependencies; SEM path analysis confirms that multi-parameter interactions dominate execution time prediction.

Instantiating these models on heterogeneous ARM big.LITTLE edge devices exposes distinct architectural divergences under workload saturation. Extending beyond matrix-matrix multiplication to advanced server-class architectures with performance-efficiency cores, alongside evaluating the robustness of newly derived performance indicators, establishes an empirical foundation for generalizable systems modeling.

5.6.3 Evolution: From The Tripartite SAC Operator Chain into Entanglement

The conceptual axes from Chapter 4 evolve into the Selection, Assessment, and Classification (SAC) methodology. This composable operator chain mathematically structures the transformation from workload parameters to execution domains:

$$\mathcal{G} = \mathcal{C}_h \circ \mathcal{A}_h \circ \mathcal{S}_h$$

Each operator executes a distinct transformation stage:

- **Selection Operator ($\mathcal{S}_h$):** Maps workload parameters (M_c, threads, core allocations) to target instruction sequences and execution footprints across memory hierarchy boundaries ($L_1 \to L_2$, $L_2 \to L_3$, and DRAM).
- **Assessment Operator ($\mathcal{A}_h$):** Translates instruction sequences and hardware performance counters into quantitative execution profiles, capturing multi-variate dependencies, cache-coherency overheads, and scheduling jitter.
- **Classification Operator ($\mathcal{C}_h$):** Partitions execution domains into discrete performance classes based on structural saturation thresholds, enabling runtime schedulers to enforce strict constraints and mitigate memory bus poisoning.

This tripartite structure resolves the combinatorial explosion of the heterogeneous design space, replacing decoupled regression with a physically grounded attribution pipeline.

Consequently, by defining mathematically the boundaries where coordinate resource interactions overwhelm parallel scaling, the multi-parameter models and the Critical Complexity Index (χ) developed in Chapter 5 expose a deep microarchitectural tension: performance on asymmetric platforms cannot be isolated to individual, independent compute units. Yet, modern operating system schedulers—such as the Linux Energy-Aware Scheduler (EAS)—as well as industry-standard power/performance estimation frameworks (e.g., McPAT [81], Wattch [17], RAPL [99]) continue to rely on the foundational assumption of "cluster independence". This heuristic assumes that high-performance and high-efficiency clusters operate as computationally and electrically orthogonal entities, where the total energy and performance states are a linear, additive "sum of their parts".

This contradiction between the non-linear interaction bottlenecks modeled in Chapter 5 and the legacy additive capacity assumptions of system software dictates a fundamental, measure-theoretic interrogation of subsystem independence. This directly motivates our discussion in to Chapter 7 (Measure-Theoretic Diagnostics of Architectural Entanglement), where we deploy the Boltzmann Uniqueness Theorem to evaluate mathematically the veracity of cluster independence, isolate the physical-electrical mechanisms of "architectural entanglement" across ARM DynamIQ and Intel x86 hybrid architectures, and formulate the "Ghost Power" Coefficient (γ) to quantify the parasitic floor elevations that render traditional additive models fundamentally underspecified.

Chapter 6

Structural Equation Modeling for Compute Diversity Articulation

As we push heterogeneous asymmetric platforms beyond simple first-principles execution models and into the realms of multi-parameter interactions, traditional modeling tools hit a wall [45, 68, 122]. Classical performance paradigms and standard linear regressions rest on a fatal assumption: that low-level hardware performance counters act as independent, perfectly measured variables that simply can be added up to predict execution speed [42, 124]. On modern out-of-order processors, however, pipeline stalls, branch mispredictions, cache-coherency bottlenecks, and shared power rails constantly interact and bleed into one another [36, 59]. Trying to force these coupled microarchitectural events into rigid, traditional regression models creates a severe compounding error problem, masking the true physical behavior of the silicon and capping our ability to forecast workload execution accurately.

To break through this predictive ceiling, we fundamentally shift how we observe and mathematically frame processing intensity, again by adopting Structural Equation Modeling (SEM) [12, 69, 78]. Rather than treating volatile, noise-heavy performance counters as isolated targets, SEM allows us to model the entire execution pipeline as a unified causal system where raw hardware telemetry serves as collective indicators of a deeper, unobserved construct that is the true internal architectural stress of the processor [12, 70]. What follows in this chapter is the rationale and mathematical justification for why SEM may succeed where traditional modeling fails. We explore how this multivariate framework systematically filters out platform-specific scheduling noise, eliminates measurement attenuation, and delivers the structural foundation required to master the non-uniform computing landscapes of tomorrow.

6.1 Asymmetry and Platform Variance Mitigation

The system operates as a physical, causal sequence where workload dictates a latent microarchitectural state, which in turn dictates execution time [12, 78]:

$$W \longrightarrow \Psi \longrightarrow E \tag{6.1}$$

Here, Ψ is a multidimensional latent variable representing "Architectural State Intensity" [12]. This latent construct cannot be measured directly by any single counter; rather, it manifests through the covariance of the observed parameters which we denote as θ for this discussion. By treating M as a set of observed indicators for the latent hardware intensity Ψ, Structural Equation Modeling (SEM) resolves the structural limitations of Ordinary Least Squares (OLS) through several key mathematical mechanisms [12, 42, 69] which are to follow. A theoretical model for the latent microarchitectural state mediating the mapping of workload to execution time is illustrated in Figure 6.1.

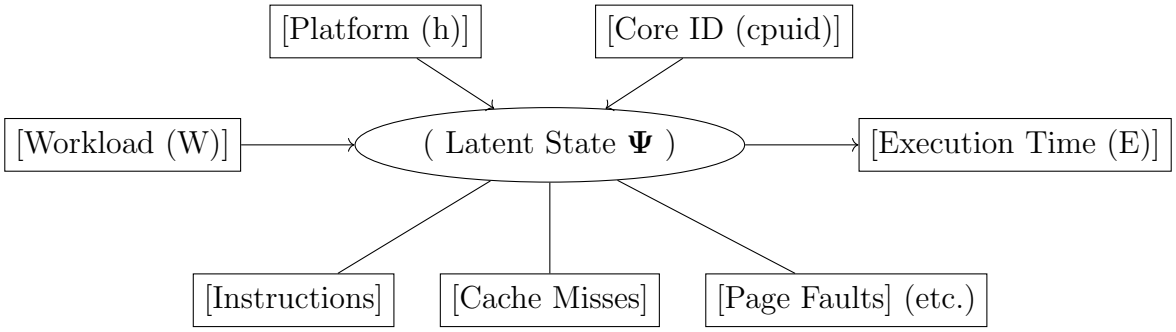


Figure 6.1: Theoretical SEM Model

6.1.1 Simultaneous Estimation and Full-Information Maximum Likelihood

Instead of fitting disjointed, independent OLS regressions that ignore error propagation, SEM estimates the measurement and structural paths simultaneously using Full-Information Maximum Likelihood (FIML) [40, 4]. This joint optimization minimizes the global discrepancy function [69, 71]:

$$F_{\text{ML}} = \ln|\boldsymbol{\Sigma}(\theta)| + \text{tr}\left(\mathbf{S}\boldsymbol{\Sigma}(\theta)^{-1}\right) - \ln|\mathbf{S}| - \mathbf{p} \tag{6.2}$$

where $\mathbf{S}$ is the empirical sample covariance matrix, $\mathbf{\Sigma}(\theta)$ is the model-implied covariance matrix as a function of the parameter vector θ, and $\mathbf{p}$ is the total number of observed (manifest) variables in the model [12, 69]. By optimizing the entire path simultaneously, SEM prevents the localized "Stage 1 Error Explosion" from biasing the structural coefficient mapping to execution time E, effectively bypassing the composite OLS ceiling.

6.1.2 Modeling the Covariance Structure of Indicators

In the multi-parameter (Big) core model, we investigate the following parameters: Instructions (I), Branch Misses (BM), L1 Loads ($L1L$), LLC Misses ($LLCM$), CPU Utilization (CPU), Context Switches (CS), CPU Migrations (CM), Page Faults (PF), and Instructions Per Cycle (IPC) [59, 124]. We argue that such counters are physically coupled by pipeline back-pressure and resource contention [36]. Where OLS treats these parameters as completely independent targets in Stage 1, it throws away crucial covariance data.

SEM explicitly models the covariance matrix of these indicators [12, 70]:

$$\mathbf{\Sigma}_{MM} = \mathbf{\Lambda}_M \mathbf{\Phi} \mathbf{\Lambda}_M^T + \mathbf{\Theta}_\delta \tag{6.3}$$

where $\mathbf{\Lambda}_M$ represents the factor loadings of the indicators on the latent state Ψ, $\mathbf{\Phi}$ is the variance of the latent state, and $\mathbf{\Theta}_\delta$ is the diagonal matrix of unique/measurement errors [12]. This allows SEM to extract the common structural signal (e.g., memory bus saturation or execution-unit bottlenecking) while filtering out the random, platform-specific noise of individual counters.

6.1.3 Resolving the Platform-Specific OS Scheduling Bias

Because asymmetric platforms have different core distributions (e.g., 4 Big cores vs. 2 Big cores) [45, 68], the OS scheduler handles multi-threaded GEMM execution differently on each board [25, 122]. This difference introduces severe, platform-specific noise into indicators like Context Switches (CS) and CPU Migrations (CM).

In a standard OLS framework, this noise directly corrupts the execution time prediction [42]. In the SEM framework, this platform-specific scheduling variance is captured in the unique

measurement error term (δ_i) for those specific indicators in the covariance matrix [12, 78]:

$$\text{Var}(CS) = \lambda_{CS}^2 \text{Var}(\Psi) + \delta_{CS} \tag{6.4}$$

Because SEM isolates the measurement error δ_{CS}, the platform-specific noise introduced by the different Big core distributions is filtered out, preventing it from propagating into the structural path ($\Psi \rightarrow E$).

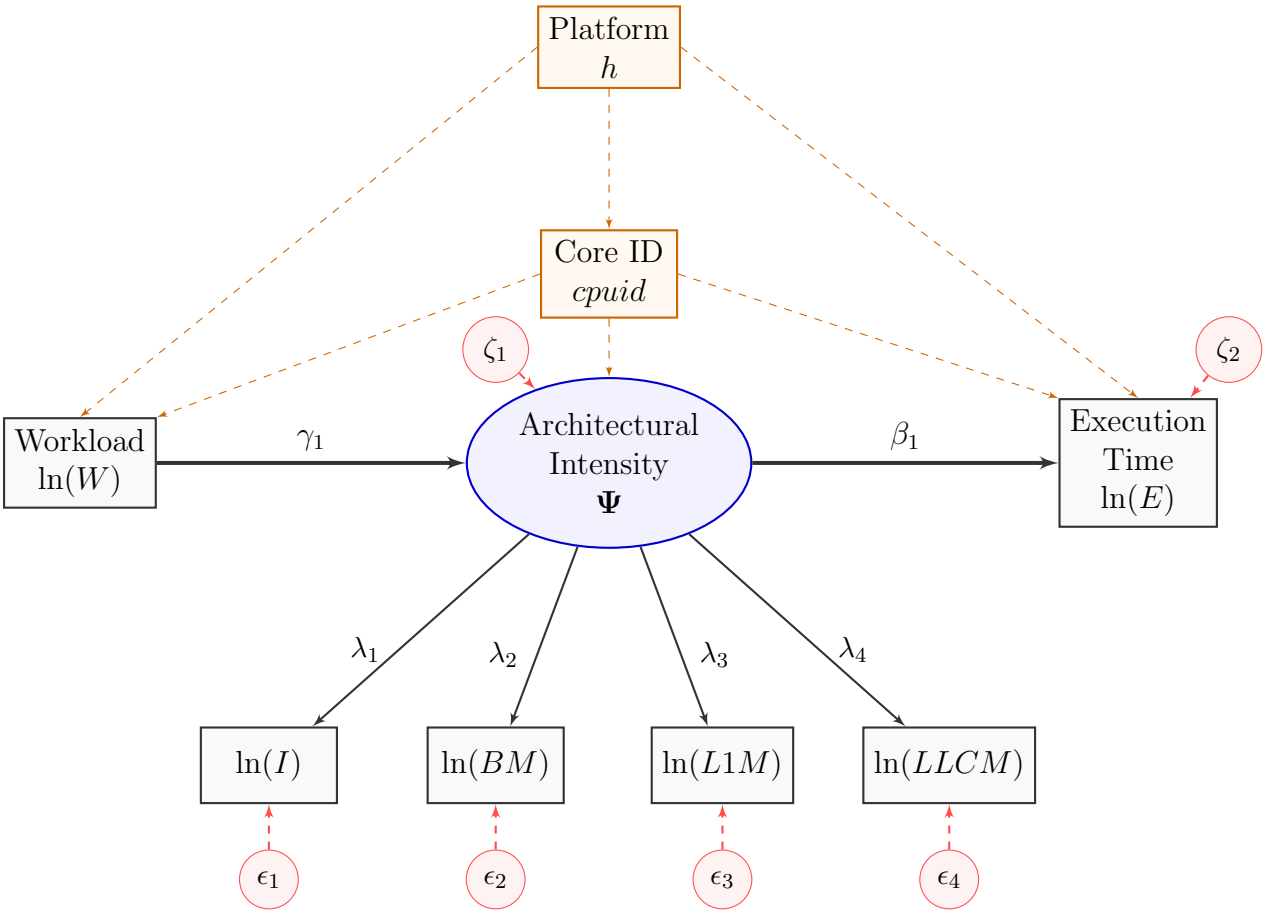


Figure 6.2: 4-Parameter Architectural Intensity SEM

6.1.4 Analysis Insights

We demonstrate that SEM models may be used to mitigate heterogeneity and articulate the nuanced performance capabilities of these unique platforms. Structural Equation Models (SEMs) for estimating unobserved Architectural Intensity (Ψ) to predict Execution Time

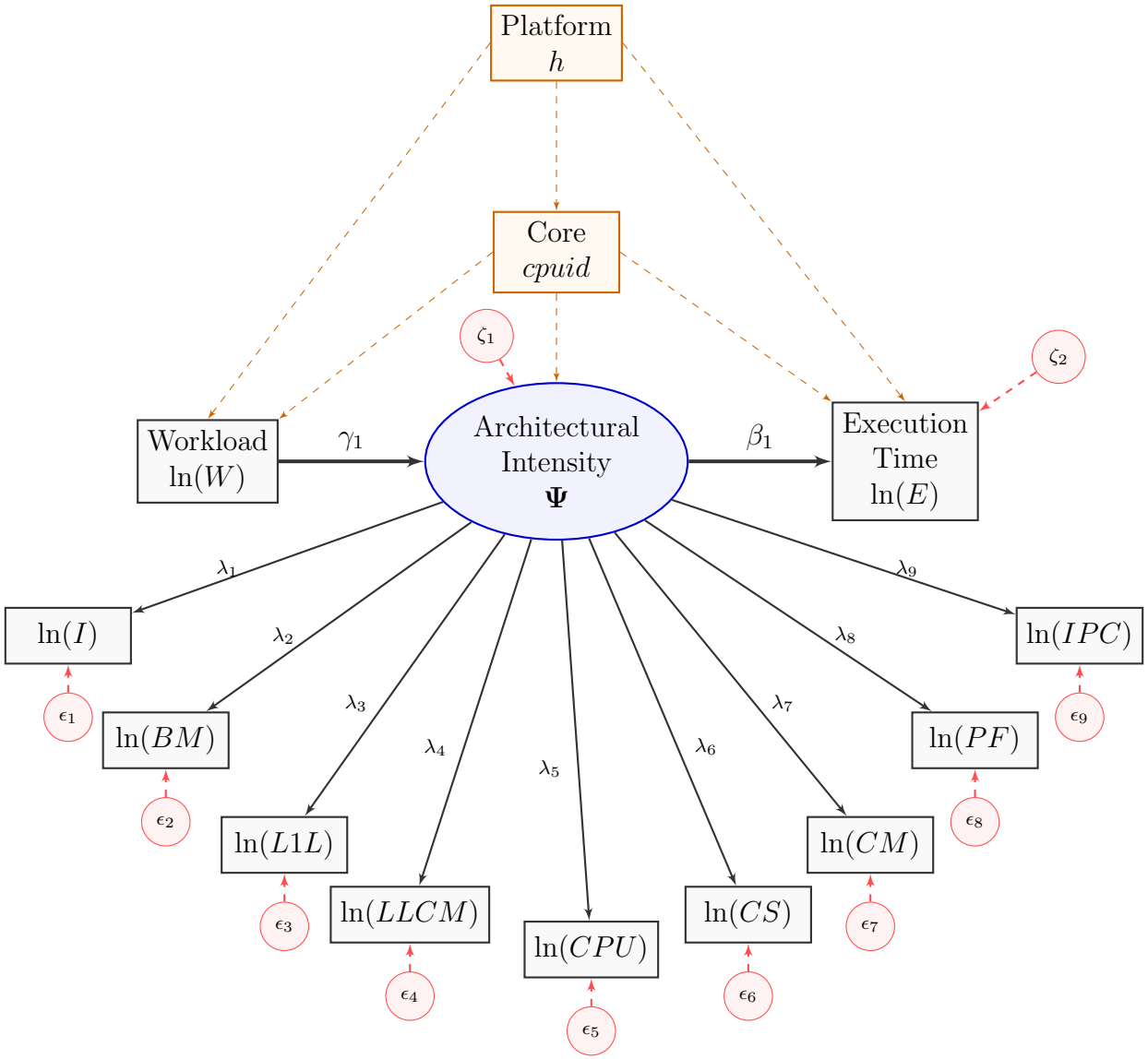


Figure 6.3: 9-Parameter Architectural Intensity SEM

($\ln(E)$) based on Workload ($\ln(W)$), conditioned on hardware covariates (h, *cpuid*), highlight:

Structural Backbone: Workload ($\ln(W)$) directly drives Architectural Intensity (Ψ), which determines Execution Time ($\ln(E)$). Platform (h) and Core ID (*cpuid*) act as hardware covariates controlling for platform-level and core-level variance.

4-Parameter Model: Uses 4 observed performance metrics as indicators ($\ln(I)$, $\ln(BM)$, $\ln(L1M)$, $\ln(LLCM)$).

9-Parameter Model: Expands the measurement model to 9 observed performance indicators ($\ln(I)$, $\ln(BM)$, $\ln(L1L)$, $\ln(LLCM)$, $\ln(CPU)$, $\ln(CS)$, $\ln(CM)$, $\ln(PF)$, $\ln(IPC)$).

The SAC models apply logarithmic transformations primarily to capture the non-linear power-law relationships inherent in computational workload scaling. In microarchitectural systems, performance metrics scale multiplicatively or polynomially with workload dimensions (W, such as GEMM matrix sizes) [43], making standard additive linear models physically inadequate. Applying natural logarithms linearizes these multiplicative power-law formulations ($Y = aX^b$) into solvable log-log linear equations ($\ln(Y) = \ln(a) + b\ln(X)$) accessible to linear regression. Additionally, log transformations stabilize variance across multiple orders of magnitude.

To operationalize the Structural Equation Modeling (SEM) framework, the model parameters are classified formally into three functional categories [12, 78]:

Structural Parameters:

- γ_1: Direct effect of Workload ($\ln(W)$) on Architectural Intensity (Ψ).
- β_1: Direct effect of Architectural Intensity (Ψ) on Execution Time ($\ln(E)$).

Measurement Loadings:

- $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ (Fig. 6.2): Factor loadings mapping latent Ψ to indicators ($\ln(I)$, $\ln(BM)$, $\ln(L1M)$, $\ln(LLCM)$).

- λ_1 through λ_9 (Fig. 6.3): Factor loadings mapping latent Ψ to all 9 performance indicators.

Error & Disturbance Terms:

- ζ_1, ζ_2: Structural disturbances (latent residuals) for Architectural Intensity (Ψ) and Execution Time ($\ln(E)$).
- ϵ_1 to ϵ_4 (Fig. 6.2) / ϵ_1 to ϵ_9 (Fig. 6.3): Measurement errors associated with each observed performance indicator.

The empirical results from Chapter 3 provide an exceptionally strong foundation for SEM modeling based on three pillars:

1. Microarchitectural Asymmetry Tax: The collapse of the 1-parameter model on Big cores ($R^2 = 0.0365$, Table 3.3) versus Little cores ($R^2 = 0.7281$, Table 3.2) shows that out-of-order pipelined execution cannot be characterized by scalar proxies [36, 122].
2. High-Dimensional Verification: The multi-parameter isolated Stage 2 fit ($R^2 \geq 0.92$, Tables 3.4 & 3.5) proves that a highly accurate physical mapping exists, but OLS is prevented from utilizing it end-to-end due to the Stage 1 Error Explosion [42, 124].
3. The SEM Solution: SEM is the mathematically appropriate tool because it formalizes the microarchitectural event matrix M as a latent variable Ψ, resolving the Errors-in-Variables bottleneck, modeling the physical covariance of the pipeline counters, and isolating platform-specific scheduling noise [4, 12, 42, 69].

6.2 Microarchitectural Asymmetric Characterization

6.2.1 Theoretical Overview of the Latent Space

To resolve the predictive limitations of Ordinary Least Squares (OLS) models on out-of-order, asymmetric cores, we formulate a system of simultaneous equations using the Linear Structural Relations (LISREL) model specification [69, 70, 71]. The empirical results demonstrate

that while a single scalar metric like instruction count (I) is a sufficient proxy for execution time (E) on simple, in-order pipelines ($R^2 = 0.9776$, Table 3.2), it collapses entirely on speculative, out-of-order execution engines ($R^2 = 0.3763$, Composite $R^2 = 0.0365$, Table 3.3) [36]. This breakdown occurs because execution speed on high-performance cores is governed by a complex, interdependent state of pipeline stalls, branch entropy, and memory hierarchy overheads [59, 122]. We use traditional LISREL notation (η) to maintain consistency with covariance structure analysis literature [12, 71].

We hypothesize that this collective microarchitectural response can be formalized mathematically as a single, unobserved, continuous latent construct: Architectural State Intensity (η) [12]. This latent variable represents the physical pressure a workload exerts on the underlying execution pipeline and memory subsystems. Instead of treating performance counters as independent regression targets, the SEM framework models them as reflective indicators of the latent variable η [12, 69]:

$$\ln(W) \longrightarrow \eta \longrightarrow \ln(E) \tag{6.5}$$

where the workload dimension (W) acts as an exogenous driver of the latent state (η), which in turn acts as the direct structural predictor of parallel execution time (E), mediated by platform characteristics (h) and core identifiers ($cpuid$).

6.2.2 The Measurement Model (Inner Model)

The measurement model defines the mathematical relationship between the latent construct η and the vector of observed microarchitectural indicators $\mathbf{y}$ [12, 69]. Let $\mathbf{y}$ be the $p \times 1$ vector of observed performance counters.

Case I: The 4-Parameter Configuration

For the 4-parameter microarchitectural feature set, the vector of indicators is defined as [59]:

$$\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{bmatrix} = \begin{bmatrix} \ln(I) \\ \ln(BM) \\ \ln(L1M) \\ \ln(LLCM) \end{bmatrix}$$

The measurement equation is formulated as [70, 12]:

$$\mathbf{y} = \mathbf{\Lambda}_y \eta + \epsilon$$

In matrix expansion:

$$\begin{bmatrix} \ln(I) \\ \ln(BM) \\ \ln(L1M) \\ \ln(LLCM) \end{bmatrix} = \begin{bmatrix} \lambda_1 = 1.0 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \end{bmatrix} \eta + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \end{bmatrix}$$

To establish the physical scale of the latent variable η, the loading for the primary proxy metric, $\ln(I)$, is constrained to 1.0 (the Unit Loading Constraint) [12, 78]. The error vector ϵ represents unique measurement and sensor noise, assumed to follow a multivariate normal distribution $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{\Theta}_\epsilon)$, where $\mathbf{\Theta}_\epsilon = \text{diag}(\sigma^2_{\epsilon_1}, \sigma^2_{\epsilon_2}, \sigma^2_{\epsilon_3}, \sigma^2_{\epsilon_4})$ is a diagonal matrix of unique variances [12, 69].

Case II: The 9-Parameter Configuration

For the high-dimensional microarchitectural vector, the indicator vector $\mathbf{y}$ is expanded to $p = 9$ to capture the complex pipeline dynamics of the Cortex-A76 cores [59, 124]:

$$\mathbf{y} = \begin{bmatrix} \ln(I) \\ \ln(BM) \\ \ln(L1L) \\ \ln(LLCM) \\ \ln(CPU) \\ \ln(CS) \\ \ln(CM) \\ \ln(PF) \\ \ln(IPC) \end{bmatrix}$$

The expanded measurement matrix is expressed as:

$$\begin{bmatrix} \ln(I) \\ \ln(BM) \\ \ln(L1L) \\ \ln(LLCM) \\ \ln(CPU) \\ \ln(CS) \\ \ln(CM) \\ \ln(PF) \\ \ln(IPC) \end{bmatrix} = \begin{bmatrix} \lambda_1 = 1.0 \\ \lambda_2 \\ \lambda_3 \\ \lambda_4 \\ \lambda_5 \\ \lambda_6 \\ \lambda_7 \\ \lambda_8 \\ \lambda_9 \end{bmatrix} \eta + \begin{bmatrix} \epsilon_1 \\ \epsilon_2 \\ \epsilon_3 \\ \epsilon_4 \\ \epsilon_5 \\ \epsilon_6 \\ \epsilon_7 \\ \epsilon_8 \\ \epsilon_9 \end{bmatrix}$$

6.2.3 The Structural Model (Outer Model)

The structural model describes the causal relationships between the exogenous workload parameter, the latent microarchitectural intensity η, and the final endogenous performance outcome E [12, 69, 78]. The system must explicitly decouple platform-specific and core-specific covariates to account for the physical architectural variations between the octa-core Orange Pi 5 ($4 \times$ Cortex-A76 $+ 4 \times$ Cortex-A55) and the Cubie A7A ($2 \times$ Cortex-A76 $+$

$6 \times$ Cortex-A55). Let $\mathbf{x}_c = [h, cpuid]^T$ be the vector of dummy-coded categorical covariates representing the platform host h and the specific hardware core identifier *cpuid* [122]. The structural equations are specified as a system of simultaneous equations:

Equation 1: Latent State Activation

$$\eta = \gamma_1 \ln(W) + \gamma_c^T \mathbf{x}_c + \zeta_1$$

Where:

γ_1 represents the structural coefficient mapping the log-scaled workload size (e.g., GEMM matrix dimensions from 64×64 to 2048×2048) to the latent intensity.

γ_c is the vector of coefficients capturing the static shifts in hardware intensity caused by the specific platform clock frequencies (2.4 GHz vs. 2.0 GHz) and OS thread allocation strategies.

ζ_1 is the structural disturbance term representing the unexplained variance in latent core behavior, where $\zeta_1 \sim \mathcal{N}(0, \psi_1)$ [12].

Equation 2: Execution Time Mapping

$$\ln(E) = \beta_1 \eta + \beta_c^T \mathbf{x}_c + \zeta_2$$

Where:

β_1 is the structural coefficient mapping the latent variable η to parallel execution time E.

β_c represents the direct impact of the platform covariates on execution time (e.g., LPDDR5 vs. LPDDR4 memory bandwidth effects).

ζ_2 is the residual structural disturbance representing execution-time variance not captured by the latent microarchitectural intensity (such as transient OS kernel interrupts), where $\zeta_2 \sim \mathcal{N}(0, \psi_2)$ [12].

6.2.4 The Covariance Structure Equation

Unlike OLS, which minimizes individual sum-of-squares errors for each equation separately, SEM fits the entire system simultaneously by minimizing the discrepancy between the empirical covariance matrix $\mathbf{S}$ and the model-implied covariance matrix $\boldsymbol{\Sigma}(\theta)$ [69, 71, 12].

Let the joint vector of observed variables be $\mathbf{Z} = [\ln(W), \mathbf{y}^T, \ln(E)]^T$. The model-implied covariance matrix $\boldsymbol{\Sigma}(\theta)$ is structured as:

$$\boldsymbol{\Sigma}(\theta) = \begin{bmatrix} \text{Var}(\ln(W)) & \text{Cov}(\ln(W), \mathbf{y}) & \text{Cov}(\ln(W), \ln(E)) \\ \text{Cov}(\mathbf{y}, \ln(W)) & \boldsymbol{\Sigma}_{yy}(\theta) & \text{Cov}(\mathbf{y}, \ln(E)) \\ \text{Cov}(\ln(E), \ln(W)) & \text{Cov}(\ln(E), \mathbf{y}) & \text{Var}(\ln(E)) \end{bmatrix} \tag{6.6}$$

The sub-matrix $\boldsymbol{\Sigma}_{yy}(\theta)$ representing the covariance of the p microarchitectural indicators is derived mathematically from the measurement model [12, 69, 70]:

$$\boldsymbol{\Sigma}_{yy}(\theta) = \boldsymbol{\Lambda}_y \text{Var}(\eta) \boldsymbol{\Lambda}_y^T + \boldsymbol{\Theta}_\epsilon \tag{6.7}$$

Expanding this covariance matrix exposes the structural constraint imposed by the latent variable:

$$\boldsymbol{\Sigma}_{yy}(\theta) = \begin{bmatrix} \text{Var}(\eta) + \sigma_{\epsilon_1}^2 & \lambda_2 \text{Var}(\eta) & \dots & \lambda_p \text{Var}(\eta) \\ \lambda_2 \text{Var}(\eta) & \lambda_2^2 \text{Var}(\eta) + \sigma_{\epsilon_2}^2 & \dots & \lambda_2 \lambda_p \text{Var}(\eta) \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_p \text{Var}(\eta) & \lambda_2 \lambda_p \text{Var}(\eta) & \dots & \lambda_p^2 \text{Var}(\eta) + \sigma_{\epsilon_p}^2 \end{bmatrix} \tag{6.8}$$

This matrix equation exposes why standard OLS fails. OLS regressions assume all off-diagonal covariance elements in $\boldsymbol{\Theta}_\epsilon$ are zero and fail to model the shared variance $\boldsymbol{\Lambda}_y \text{Var}(\eta) \boldsymbol{\Lambda}_y^T$ [12, 42]. By forcing these parameters into separate, independent equations, OLS discards the rich covariance structure of the pipeline counters, leading to severely biased parameter estimates.

6.2.5 Mitigation of the Errors-in-Variables (EIV) Bias

The primary statistical bottleneck in the multi-stage OLS mappings is the Stage 1 Error Explosion from mapping Workload to individual parameters ($W \to y_i$). For the 9-parameter Big core mapping, predicting the microarchitectural features from the workload W yields unmanageable test errors, such as a 5756.69% MAPE for Instructions (I) and a 6908.03% MAPE for L1 dcache loads ($L1L$). The full list of predictions for both the 4-parameter and 9-parameter models are given in Table 6.1 and 6.2 respectively.

Table 6.1: Stage 1 Individual Predictive Performance (4-Parameter Configuration)

Microarchitectural Metric	**Fit (R^2)**	**Test RMSE**	**Test MAPE**
Instructions (I)	0.5355	1.884×10^{11}	5860.15%
Branch Misses (BM)	0.7900	7.005×10^{5}	148.80%
L1 dcache Load Misses ($L1M$)	0.5898	2.606×10^{9}	6952.10%
LLC Load Misses ($LLCM$)	0.8853	1.012×10^{8}	227.39%

Table 6.2: Stage 1 Individual Predictive Performance (9-Parameter Configuration)

Microarchitectural Metric	**Fit (R^2)**	**Test RMSE**	**Test MAPE**
Instructions (I)	0.4837	1.432×10^{11}	5756.69%
Branch Misses (BM)	0.7597	6.824×10^{5}	170.36%
L1 dcache Loads ($L1L$)	0.4764	6.670×10^{10}	6908.03%
LLC Load Misses ($LLCM$)	0.8698	7.181×10^{7}	178.88%
CPU Utilization (CPU)	0.2981	0.1249	20.34%
Context Switches (CS)	0.7508	6.979×10^{3}	96.48%
CPU Migrations (CM)	0.2962	6.9045	55.55%
Page Faults (PF)	0.4072	1.969×10^{3}	597.26%
Instructions Per Cycle (IPC)	0.2388	0.7360	129.47%

The OLS Bias Formulation (Classical Errors-in-Variables)

In the OLS Stage 2 model, execution time is regressed on the predicted performance counter matrix $\hat{M}$ from Stage 1. Analyzing the scalar case where $\ln(E)$ is regressed on predicted instruction count $\widehat{\ln(I)}$:

$$\widehat{\ln(I)}_i = \ln(I)_i + u_i \tag{6.9}$$

where $\ln(I)_i$ is the true microarchitectural execution state and $u_i \sim \mathcal{N}(0, \sigma_u^2)$ is the measurement error representing Stage 1 prediction noise.

The Stage 2 OLS model is specified as:

$$\ln(E)_i = \beta_1 \widehat{\ln(I)}_i + (\epsilon_{2i} - \beta_1 u_i) \tag{6.10}$$

Because the predictor $\widehat{\ln(I)}_i$ is correlated mathematically with the compound error term $(\epsilon_{2i} - \beta_1 u_i)$ via u_i, the OLS estimator for the execution coefficient β_1 suffers from attenuation bias [112, 85, 42]:

$$\text{plim } \hat{\beta}_{1,\text{OLS}} = \beta_1 \left(\frac{\sigma^2_{\ln(I)}}{\sigma^2_{\ln(I)} + \sigma^2_u} \right) < \beta_1 \tag{6.11}$$

Where:

$\sigma^2_{\ln(I)}$ is the true variance of the instruction count.

σ^2_u is the variance of the prediction error from Stage 1.

Because Stage 1 error variance σ^2_u is exceptionally large (evidenced by 3297.11% 4-parameter and 1545.89% 9-parameter MAPEs), the reliability ratio $\frac{\sigma^2_{\ln(I)}}{\sigma^2_{\ln(I)}+\sigma^2_u}$ approaches zero [85, 42]. This severely biases $\hat{\beta}_{1,\text{OLS}}$ toward zero, flatlining predictive performance and creating the observed composite ceiling where end-to-end forward R^2 remains stuck at ~ 0.73 (4-parameter, Table 3.4) and 0.7319 (9-parameter, Table 3.5).

The SEM Full-Information Maximum Likelihood (FIML) Mitigation

SEM bypasses this bias by treating η as a latent state and optimizing the joint likelihood function of the entire parameter vector θ [40, 4]. Under Full-Information Maximum Likelihood (FIML), the discrepancy function is defined as [69, 71]:

$$F_{\text{ML}}(\theta) = \ln |\mathbf{\Sigma}(\theta)| + \text{tr}\left(\mathbf{S}\mathbf{\Sigma}(\theta)^{-1}\right) - \ln |\mathbf{S}| - (p+2) \tag{6.12}$$

Where:

$\mathbf{S}$ is the empirical sample covariance matrix of the joint vector $\mathbf{Z} = [\ln(W), \mathbf{y}^T, \ln(E)]^T$.

$\mathbf{\Sigma}(\theta)$ is the constrained, model-implied covariance matrix.

$(p+2)$ is the total number of observed variables.

By estimating the measurement model ($\mathbf{y} = \mathbf{\Lambda}_y \eta + \epsilon$) and structural model ($\ln(E) = \beta_1 \eta + \zeta_2$) simultaneously, the FIML estimator isolates measurement error variances ($\mathbf{\Theta}_\epsilon$) from structural variance ($\text{Var}(\eta)$) [4, 42, 69]. Since $\mathbf{\Theta}_\epsilon$ is modeled on the diagonal of the covariance structure, the covariance elements $\text{Cov}(\mathbf{y}, \ln(E))$ and $\text{Cov}(\ln(W), \mathbf{y})$ are estimated free of prediction error noise u_i. Mathematically, this restores the reliability ratio to 1.0, allowing structural path coefficient β_1 to be estimated without bias [42].

6.3 Empirical Validation

In empirical experiments, evaluating 9-Parameter Inverse Big Core Mapping (Chapter 3 Table 3.6) proves the Inverse Accuracy Gap: aggregate execution time (E) acts as a tightly constrained global proxy capable of reconstructing workload dimensions (W) with near-deterministic composite precision ($R^2 = 0.9468$, MAPE = 11.43%), despite massive intermediate point-wise prediction volatility in Stage 1 (MAPE = 1751.97%). Mathematically, this stark asymmetry between the flat forward composite ceiling ($R^2 \approx 0.73$) and high inverse composite fidelity ($R^2 \approx 0.95$) provides the empirical "smoking gun" that hardware performance counters (M) do not act as independent intermediate regressors, but as correlated reflective indicators of an underlying, unobserved latent execution state (Ψ) [12, 69].

This formulation explains why the 9-parameter inverse composite R^2 has such high-fidelity and proves that the inverse mapping succeeds because the combination of execution time and microarchitectural counters restricts the latent space, proving mathematically that the structural signal is highly deterministic once individual counter noise is suppressed. By using SEM, this high-fidelity structural path is utilized in both directions, resolving the "asymmetry tax" of heterogeneous processor modeling [122, 36]. Consequently, the findings clearly demonstrate that standard multi-stage OLS fails because it discards cross-counter covariance and induces severe Errors-in-Variables (EIV) attenuation bias, establishing the mathematical necessity of Structural Equation Modeling (SEM) to estimate joint measurement and structural paths simultaneously [12, 42, 69].

Thus, this mathematical framework establishes the statistical and physical rationale for the transition to SEM, demonstrating that the structural limitations of OLS are not due

to an information deficit in the microarchitectural counters, but rather the mathematical formulation of the OLS estimator under high-variance measurement conditions.

Chapter 7

Measure-Theoretic Diagnostics of Architectural Entanglement in Asymmetric Multiprocessing Systems: A Boltzmann Uniqueness Approach

7.1 Introduction

Orchestration of Asymmetric Multiprocessing Platforms (AMPs), such as ARM big.LITTLE, frequently rely on the heuristic assumption of cluster independence, wherein high-performance ("Big") and high-efficiency ("Little") cores operate as computationally orthogonal resources. These cores are partitioned into "islands" of separate power/performance clusters, operating on independent voltage/frequency rails. However, these platforms share resources, including Last-level Cache (LLC), main memory, and interconnect across all cores. We therefore must assume that islands interact, operating in a "functionally coupled state." To conduct a formal, measure-theoretic evaluation of this assumption, we apply the Boltzmann uniqueness theorem, recently demonstrated [101] to be a method to determine the veracity of this assumption. Mathematically, we define an "uncoupled" system as a joint resource measurement that is strictly the convolution of its subsystem measures.

Modern edge-cloud and embedded computing platforms have transitioned toward non-uniform heterogeneous architectures to satisfy competing demands for real-time processing and thermal efficiency. These platforms introduce complex orchestration challenges due to their execution flexibility. Moreover, the emergence of Asymmetric Multiprocessing Platforms

(AMPs) breaks the uniformity of relationships between workloads and compute units. Central to these scheduling challenges is the unverified assumption of subsystem independence. If clusters were truly independent, their joint energy and performance states would be the simple sums of their parts. We refute this assumption for three different heterogeneous platforms, specifically, the Orange Pi 5, the Cubie A7A, and the Intel i7-12800H, the first two of which are representative of common embedded architectures at the edge, by using the Boltzmann uniqueness theorem [101] to provide a rigorous mathematical diagnosis and demonstration to the contrary.

By strictly isolating the execution of core clusters and comparing the convolution of their independent energy probability measures against their observed joint behavior, we show that these subsystems are physically and informationally entangled. For the third platform, we evaluate the Intel i7-12800H to extend the investigation from the embedded ARM domain to high-performance x86 mobile workstation architectures. In this hybrid topology, we believe that architectural entanglement is likely mediated through RAPL (Running Average Power Limit) domain overlap and Last-Level Cache (LLC) starvation within a 24 MiB unified memory hierarchy shared across 20 hardware threads.

Modern heterogeneous architectures, such as ARM's big.LITTLE, are designed to balance peak throughput with thermal efficiency. Traditionally, the mapping of workloads to these disparate core clusters has been treated as a variant of the knapsack problem, assuming that resource consumption is additive [17, 66, 81, 99]. However, the emergence of shared-resource bottlenecks—Last-Level Cache (LLC), interconnects, and Power Management Integrated Circuits (PMICs))—suggests that these clusters do not truly operate in isolation. Current operating system schedulers, including the Linux Energy-Aware Scheduler (EAS) [95], utilize static capacity tuples to model energy costs. These models assume that the energy state of a Little core is independent of the concurrent state of a Big core. We will investigate these platforms to understand the true underlying relationship between these compute clusters.

Finally, the findings presented in this chapter are adapted from the peer-reviewed publication:

> Harris, S. D., Gill, C. D., and Chamberlain, R. D. (2026). "Measure-Theoretic Diagnostics of Architectural Entanglement in Asymmetric Multiprocessing Systems: A Boltzmann Uniqueness Approach." *Modelling*, 7(4), pp. 124:1–124:31.

7.1.1 Chapter Overview

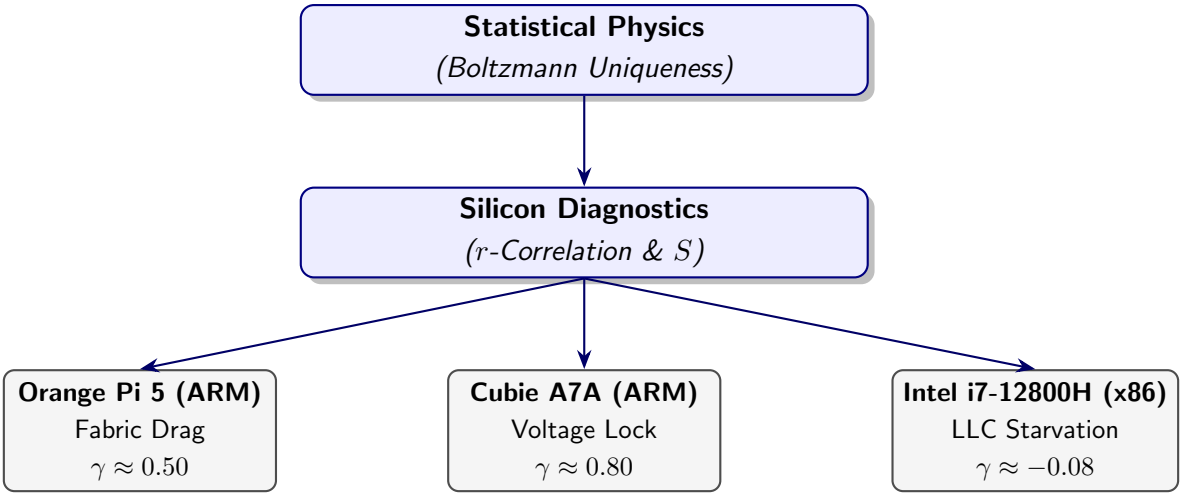


Figure 7.1: Measure-theoretic diagnostic flow mapping statistical physics principles to empirical hardware anomalies across heterogeneous architectures.

This research represents the culmination of a three-phase analytical progression. The initial investigation established baseline performance models for non-uniform heterogeneous platforms, utilizing Structural Equation Modeling (SEM) on the Orange Pi 5 to validate first-principles metrics and address fundamental orchestration challenges (Chapter 4). The subsequent phase expanded SEM application across the Orange Pi 5 and Radxa Cubie A7A to quantify complex parameter interactions, identifying inter-island scaling friction and introducing the critical complexity index (χ) to determine the structural threshold where cache-coherency traffic negates parallel efficiency (Chapter 5). Building upon this foundation of statistical parameter modeling and multi-core saturation analysis, the current research elevates empirical hardware anomalies to fundamental violations of subsystem orthogonality. By applying the Boltzmann uniqueness theorem [13, 76], this chapter provides a mathematical diagnosis of architectural entanglement, transitioning from heuristic-based structural assumptions to measure-theoretic relational modeling. As illustrated in Figure 7.1, the measure-theoretic diagnostic framework translates foundational principles of statistical physics—specifically the Boltzmann uniqueness theorem—into empirical silicon metrics (e.g., the ghost power coefficient (γ)).

The remainder of this chapter is structured as follows. Section 7.2 presents the measure theoretic framework used for this investigation. Section 7.3 details the hardware platform specifications, our transcendental Gregory-Leibniz series workload construction, and our

high-resolution telemetry stack. Section 7.4 presents our detailed empirical diagnostic findings across the Orange Pi 5, Radxa Cubie A7A, and Intel i7-12800H platforms under varying power configurations and frequency lock-down operations. Section 7.5 discusses the failure of traditional additive models and explores "power-rail-aware" system scheduling policies. Section 7.6 extends the investigation from ARM platforms to x86 architectures to confirm that these findings apply across architectures. Section 7.7 goes further by detailing the notion of entanglement on x86 platforms and revealing its distinct characteristics on these platforms. Section 7.8 provides an in-depth discussion of the Intel power independence as compared to the ARM platform power independence and illustrates the determining factors across their power sharing characteristics. Section 7.9 summarizes key takeaways, and transitions to our next investigation in the biomedical engineering domain where we investigate clinical performance constraints in Chapter 8.

7.1.2 Problem Statement and Objective

The objective of this research is to evaluate heuristics by applying the Boltzmann uniqueness theorem [13, 101] to conduct a formal, measure-theoretic evaluation of AMP topologies. By isolating the execution of core clusters and comparing the convolution of their independent energy probability measures against observed joint behavior, this research aims to show that shared architectural resources render traditional additive power models fundamentally underspecified.

Modern heterogeneous platforms treat the mapping of workloads to disparate core clusters as a variant of the knapsack problem, assuming resource consumption is additive [17, 66, 81, 99]. Current operating system schedulers, including the Linux Energy-Aware Scheduler (EAS), utilize static capacity tuples to model energy costs. These frameworks assume cluster independence, meaning that the energy state of a high-efficiency cluster is independent of the concurrent state of a high-performance cluster. Under this assumption, resource consumption is treated as being additive:

$$P_{\text{total}} = \sum P_{\text{core}} + P_{\text{uncore}} \tag{7.1}$$

where the total power of the cores is simply the sum of the individual, independently assumed cores, as shown in Equation 7.1. Yet, our empirical telemetry across modern heterogeneous platforms demonstrates that this additive heuristic is fundamentally underspecified. When

core islands share critical microarchitectural fabric, such as the LLC, system buses, or primary voltage rails, their operational states become interactively coupled. The objective of this chapter is to deploy the Boltzmann uniqueness theorem to test this independence formally, isolate the physical mechanisms driving inter-cluster power bleeding, and establish a matrix-based relational power model to replace traditional "sum of parts" assumptions.

7.1.3 Chapter Research Questions

To evaluate the statistical and physical boundaries of cluster independence, this chapter investigates the following research questions:

Does the heuristic assumption of cluster independence hold under rigorous measure-theoretic evaluation using the Boltzmann uniqueness theorem?

Do shared architectural resources, specifically the Last-Level Cache (LLC) and Power Management Integrated Circuits (PMICs), induce a functionally coupled state between logically separate power/performance islands?

Does software-level pinning to a specific core island prevent significant power floor elevation in theoretically idle adjacent clusters?

Can we prove mathematically cluster independence or coupling via measure-theoretic abstractions?

What physical-electrical mechanisms govern the failure of convolution commutativity?

Does architectural entanglement generalize from low-power embedded ARM SoCs to high-performance x86 mobile workstation processors?

How can operating system schedulers adapt to relational, non-linear power-floor elevations?

7.2 Measure-Theoretic Framework

7.2.1 Architectural Mapping

We define an "uncoupled" system as one where the joint resource measure is strictly the convolution of its subsystem measures. To identify the state of the system, we take energy measurements from the compute clusters (e.g. Big/Little islands) treating the power consumption profile of a cluster as its energy measure, μ, where:

1. Let μ_{big} be the probability distribution of isolated Big cluster states.
2. Let μ_{little} be the probability distribution of isolated Little cluster states.
3. Let μ_{joint} be the observed distribution during concurrent saturation.
4. Let $\mu_{convolved}$ be the convolution of the Big and Little cluster states.

The *observed joint measure* (μ_{joint}), taken when both clusters run simultaneously, under the null hypothesis H_0 (independence), must satisfy the following:

$$\mu_{joint} \approx \mu_{big} * \mu_{little} \tag{7.2}$$

The *energy convolution* ($\mu_{convolved}$), is calculated by taking energy measurements from the Big (μ_{big}) and Little (μ_{little}) clusters in isolation to compute their convolution:

$$\mu_{convolved} = \text{convolve}(\mu_{big}, \mu_{little}) \tag{7.3}$$

Using the *convolved measure* ($\mu_{convolved}$) and the *observed joint measure* (μ_{joint}) the Pearson Correlation Coefficient (r) is calculated for the two energy parameters [102]:

$$r = \text{Corr}(\mu_{\text{convolved}}, \mu_{\text{joint}}) \tag{7.4}$$

A high correlation ($r \in [0.9, 1]$) signifies an uncoupled system with a lower correlation ($r \in (-1, 0.9)$) indicative of architectural coupling. A divergence from equality ($r \ll 0.9$) provides evidence that the subsystems are physically and informationally entangled. The comprehensive correlation relationships are given in Appendix A Table A.1.

7.2.2 Matrix-Based Relational Model

The complex interactions and nuanced dependencies observed within the power domains motivates a new model formulation that facilitates a paradigm shift from the traditional "sum of parts" assumption ($P_{total} = \sum P_{core} + P_{uncore}$) to a matrix-based power model, which we denote as the *relational power model*:

$$P_{total} = \mathbf{V}^T \mathbf{M} \mathbf{f} + P_{leakage}(\mathbf{V}, T)$$

where:

$\mathbf{V}^T$: The **Voltage Vector (Transposed)**. This represents the operating voltages across different independent or entangled voltage domains (e.g., Big cluster, Little cluster).

$\mathbf{M}$: The **Capacitance-Activity Matrix**:

$$\mathbf{M} = \begin{bmatrix} C_1 & \gamma_{12} C_2 & \dots \\ \gamma_{21} C_1 & C_2 & \dots \\ \vdots & \vdots & \ddots \end{bmatrix}$$

is a configuration matrix that encodes the effective capacitance (C) and the power coefficients (γ) for each resource. It accounts for the magnitude of architectural entanglement and identifies how domains are interconnected physically through shared power rails and SoC fabric. The diagonal elements (C_i) represent the workload-saturated capacitance of the target island, while the off-diagonal elements (γ_{ij}) quantify the parasitic floor elevation induced in domain i per unit of activity-driven dissipation in domain j. We will further elaborate on the constitution of the power coefficient in Section 7.2.3.

$\mathbf{f}$: The **Frequency Vector**. This contains the operational clock speeds (clock frequencies) for each corresponding voltage domain in the system.

$P_{leakage}$: The **Leakage Power Function**. This is a non-linear function that calculates the "wasted" power of the SoC.

V: The **Voltage Vector**. Leakage is highly sensitive to the supply voltage; as voltage increases, leakage current typically increases exponentially.

T: **Temperature**. This represents the junction temperature of the silicon. Leakage is thermally dependent; as the chip gets hotter, the semiconductor material becomes more conductive, leading to higher "ghost" power draw.

Dynamic Power Components ($\mathbf{V}^T\mathbf{Mf}$) represents the power consumed by the charging and discharging of capacitive loads during logic transitions.

Static Power Components ($P_{leakage}(\mathbf{V}, T)$) represents the power lost while transistors are in a non-switching state, primarily due to sub-threshold conduction and tunneling.

Though this model enables the articulation of complex power usage observed on these devices, more formally we require robust modeling measurements representative of the ratio of parasitic power elevation in an *idle computational domain* per unit of activity-driven power dissipation in a *targeted adjacent domain* to capture accurately the power consumption behavior expressed within these platforms.

7.2.3 Ghost Power Coefficient

We therefore introduce the *ghost power coefficient* (γ) which is a dimensionless architectural metric that quantifies the degree of functional and electrical entanglement between theoretically isolated core clusters in asymmetric multiprocessing systems. A value of $\gamma = 0$ signifies perfect architectural orthogonality and values approaching 1.0 indicate near-total functional coupling. This metric facilitates a transition from scalar additive models to matrix-based relational power models by measuring the discrepancy between logical software isolation and physical hardware coupling which we define as:

$$\gamma = \frac{P_{\text{idle,elevated}} - P_{\text{idle,baseline}}}{P_{\text{target,active}} - P_{\text{target,baseline}}}$$

where:

$P_{\text{idle,elevated}}$: The total power draw measured in the unutilized domain while the adjacent target domain is under maximum computational load.

$P_{\text{idle,baseline}}$: The static power floor of the idle domain in an isolated, deep-sleep state (C-state).

$P_{\text{target,active}}$: The power draw of the target domain during active workload execution (e.g., workload saturation).

$P_{\text{target,baseline}}$: The static baseline power of the target domain prior to workload activation.

Empirical data from the Rockchip RK3588 and Allwinner A733 platforms demonstrate that γ is driven by physical design constraints: "monolithic voltage locking" in budget-tier SoCs, where the unified voltage domain pins the power floor of all cores to the highest requested frequency state, and "fabric drag," where active compute units force shared system-on-chip (SoC) fabric—including memory controllers and coherency interconnects—to high-frequency states that pull idle domains out of deep-sleep C-states. Consequently, γ serves as a rigorous statistical validator for the failure of software-level thread pinning to achieve hardware-level electrical isolation.

7.3 Experimental Methodology

We evaluate two distinct ARM platforms to ensure that the results are not specific to a single SoC generation, along with a modern Intel platform that mirrors the underlying heterogeneity of the ARM platforms. Our use of the term "platform" necessarily includes both the processor IC as well as additional components of the computer that the IC is a part of. While we draw tentative conclusions about causes based upon knowledge of how the IC is designed, our empirical results do not *demonstrate* causality (even as they *do* act as a counterexample to independence). Descriptions of the three platforms are provided in Chapter 2.

7.3.1 Workload Construction

To eliminate software variance, we utilized a deterministic CPU-bound workload. This workload saturates the Arithmetic Logic Units (ALUs) and generates consistent memory

traffic without I/O blocking, thus minimizing OS-induced latency jitter, by utilizing the Gregory–Leibniz series identity:

$$\pi = 4 \times \arctan(1). \tag{7.5}$$

As π cannot be expressed as a root of any polynomial with rational coefficients, computing $4 \times \arctan(1)$ is a classic transcendental calculation utilizing a repetitive finite-precision approximation of a transcendental constant. This workload generates a steady computational "heat" by calculating π to 2500 decimal places. This creates a stable "energy measure" μ by pinning the cores in a sustained high-power state. During the calculation process, the workload will consume 100% of the CPU cores assigned to the task. See Appendix B for detailed specifications.

7.3.2 Platform Analysis

The workload represents the analytical stage of a rigorous diagnostic framework designed to evaluate the architectural entanglement of asymmetric multiprocessors. This process is rooted in the Boltzmann uniqueness theorem, which identifies the Boltzmann distribution as the unique family of measures satisfying independence for uncoupled systems by commuting with convolution. The algorithmic process to acquire and evaluate the power (energy) distributions for the Big, Little, and joint configurations can be summarized as follows:

Set 1 (Big Cluster): A steady-state workload (i.e., the transcendental π calculation) is pinned to the Big cores using '`taskset`' to gather energy telemetry for the Big cluster.

Set 2 (Little Cluster): The same workload is pinned to the Little cores with energy telemetry gathered for the Little cluster.

Set 3 (Joint System): Workloads are executed simultaneously on both Big and Little clusters to measure their actual combined energy telemetry and behavior.

The empirical diagnostic framework, as implemented in our experiments, utilizes a systematic "target island" methodology to evaluate the functional orthogonality of hardware power domains. While this process provides the necessary telemetry for the subsequent

measure-theoretic disproof of independence via the Boltzmann uniqueness theorem, the specific algorithmic actions involve real-time power estimation and relational scoring, rather than on-chip binning or convolution.

The algorithmic process used to acquire and evaluate the power distributions for the isolated and joint configurations is summarized as follows (Appendix C has additional details):

Phase 1 (Targeted Saturation): An intensive workload is pinned directly to a designated target island (e.g., Big0, Big1, or Little) to isolate software-level utilization.

Phase 2 (Non-Target Monitoring): While the target island is under maximum load, the framework concurrently measures the power profile of theoretically "idle" adjacent clusters to identify parasitic floor elevation.

Phase 3 (Hardware Telemetry Acquisition): The framework performs real-time sampling of voltage (V) and frequency (f) states from the SoC power management IC (PMIC) and CPU frequency governors.

After acquiring telemetry, we perform an automated diagnostic analysis through the following algorithmic steps:

Step 1: Relational Power Estimation: The raw telemetry is processed via a hardware-calibrated power model: $P = C \cdot V^2 \cdot f$. To scale the results into realistic architectural ranges, each island is assigned a normalized capacitance coefficient (C):

Little Cores ($C = 0.35$): Typically in-order, narrower pipelines with smaller caches and fewer transistors. Lower physical complexity results in lower effective capacitance.

Big Cores ($C = 0.85$): Typically out-of-order execution units, wider issue widths, and larger structures. The increased transistor count and wire length result in higher capacitance.

Step 2: Independence Scoring (S): The framework calculates an *independence score* ($S = P_{target}/P_{non-target}$), which quantifies the ratio of work-driven power dissipation to the parasitic load drawn by idle neighbors.

Step 3: Categorization and Diagnosis:

Entanglement (Pseudo-Isolation): $S \in [1.5, 2.0]$ indicates that the system retains measurable symmetry-breaking behavior. While the island is "entangled" via the shared SoC fabric and LLC, the work-driven power consumption still exceeds the parasitic load of its neighbors by a factor of two, suggesting that while resources are coupled, the hardware retains identifiable (albeit leaky) physical boundaries.

Linking (Functional Collapse): $S \in (1.0, 1.5)$ signifies systemic linking. In this state, logical software isolation (thread affinity) provides zero hardware-level benefit because the electrical floor is dictated by the highest requested state. On the A733, this indicates monolithic voltage locking ($S \approx 1.21$), where a shared domain pins the entire die to a high-voltage state. This proves that the domains have merged into a single electrical entity, rendering the two clusters electrically monolithic.

Critical Threshold: When $S \approx 1.0$, the "idle" domain draws power approximately equal to the "target" domain. On the RK3588, this identifies fabric drag ($S \approx 1.02$), where shared interconnects pull idle cores out of deep sleep.

Step 4: Validation: The resulting joint resource measurements are subjected to a measure-theoretic evaluation. The systemic failure of convolutional commutativity, characterized by peak negative correlations, constitutes the conclusive empirical disproof of cluster independence.

7.4 Empirical Results

The following empirical analysis evaluates the correlation between these predicted convolutions and the observed joint states to determine the extent of microarchitectural divergence. The experimental results consistently reject the "uncoupled" null hypothesis across both big.LITTLE platforms. The correlations between the energy subsystem convolutions are summarized in Table 7.1, with the empirical power diagnostic data for each isolated island test shown in diagnostic Tables 7.2 and 7.3. The standard error on the correlation (r) is shown as SE_r, and the physical mechanisms indicated are based on both the design of the chip itself and the full single-board computer on which it is deployed. The parasitic ghost

power coefficients are delineated in Table 7.4, with the correlation characterizations defined in Appendix A, Table A.1. For both platforms, the correlation (r) and associated standard error (SE_r) are well below the threshold for independence.

Table 7.1: Summary of Correlation Contrast

Metric	**Orange Pi 5 (RK358)**	**Cubie A7A (A733)**
Correlation (r)	-0.42, $SE_r = \pm 0.03$ ($p < 0.001$)	-0.10, $SE_r = \pm 0.033$ ($p = 0.003^{\dagger}$)
Physical State	Fabric Drag: Coupled via shared interconnects and LLC.	Voltage Locking: Coupled via monolithic power rail.
Isolation Degree	Pseudo-Isolated: Discrete physical islands exist but leak.	Non-Existent: Software clusters are electrically linked.
Modeling Shift	Requires relational modeling of interconnect transients ($\gamma \approx 0.5$).	Requires total matrix-based relational power modeling ($\gamma \approx 0.8$).

†The Cubie A7A result uses a conservative baseline for reported negative correlations ($r < 0, p = 0.003$).

Table 7.2: Rockchip RK3588 (Orange Pi 5) Power Island Analysis

Target Island	**Status**	**Indep. Score**	**Big0 Power**	**Big1 Power**	**Little Power**
Big0	Entangled	2.00	3.213 W	0.602 W	1.006 W
Big1	Linked	1.36	0.602 W	2.560 W	1.275 W
Little	Linked	1.02	0.602 W	0.602 W	1.228 W

Table 7.3: Allwinner A733 (Cubie A7A) Power Island Analysis.

Target Island	**Status**	**Indep. Score**	**Big Power**	**Little Power**
Big	Linked	1.21	0.438 W	0.362 W
Little	Linked	1.33	0.284 W	0.380 W

Note: Brackets or highlights indicate primary island power draw under isolated test loads.

7.4.1 Cluster Independence

The experimental diagnostic procedure utilizes a systematic target island stress test to evaluate the functional orthogonality of power domains in modern AMP SoCs. By pinning intensive computational workloads to a specific cluster (i.e., the target island) and concurrently measuring the power consumption of theoretically idle adjacent clusters, the methodology

Table 7.4: Ghost Power Coefficient and Architectural Entanglement Metrics

Platform	**Ghost Coefficient (γ)**	**Dominant Mechanism**	**Status**	**Modeling Implication**
Orange Pi 5	≈ 0.50	Fabric Drag	Entangled (1.02–2.00)	Relational Modeling
Cubie A7A	≈ 0.80	Monolithic Voltage Locking	Linked (1.21–1.33)	Matrix-based Modeling
Ideal	0.00	Total Isolation	Absolute Orthogonality	Additive Modeling

identifies a fundamental discrepancy between logical software isolation (thread affinity) and physical electrical isolation. The resulting independence score serves as a metric for quantifying "architectural entanglement." A lower score indicates that inter-cluster cross-talk—mediated by the power delivery network and shared microarchitectural fabric—dominates the system's power profile, rendering the islands computationally and electrically coupled. These empirical results provide a clear rejection of cluster independence, as follows:

RK3588 (Orange Pi 5): The Little cluster yields a critical independence score of 1.02, indicating a "fabric drag" phenomenon where stressing the Big1 cluster causes the idle Little domain to draw 1.275 W. This is higher than the power draw when Little cores are themselves the primary targets (1.228 W). Active compute units force shared SoC fabric resources to high-frequency states, pulling idle domains out of deep sleep.

A733 (Cubie A7A): The platform exhibits "monolithic voltage locking," yielding independence scores as low as 1.21 for the Big island. Given that the A733 utilizes a unified voltage domain, frequency scaling on active cores pins the global voltage high, inducing massive static leakage in theoretically isolated idle clusters. The target cluster (Big) draws only 0.076 W more than the theoretically "idle" cluster (Little). Frequency scaling on one cluster pins the global voltage domain high for the entire chip, rendering software-defined islands electrically monolithic.

These findings, classified as "entangled" or "linked" across both platforms, suggest that hardware wiring and shared resource fabric override software-level thread pinning, motivating a paradigm shift toward matrix-based relational power modeling.

7.4.2 Measurement States

In the context of the measure-theoretic evaluation of asymmetric multiprocessing architectures, the Cubie A7A is identified as having an inherent architectural entanglement, revealing a die-wide structural property of the hardware rather than a localized or transient interaction. In the specific case of the Allwinner A733 (Cubie A7A), this systemic coupling triggers a failure of convolution commutativity that persists across the entire operational range because the underlying physical resources are monolithic.

The contrast between the correlations for the two devices is defined by their distinct physical mechanisms of entanglement and the resulting mathematical implications:

1. Orange Pi 5 (RK3588): $r = -0.4248$

   Physical Mechanism: The RK3588 possesses discrete, albeit poorly isolated, physical power islands. Its entanglement is primarily driven by "fabric drag."

   Technical Implication: When a high-performance cluster computes, it forces shared components—such as memory controllers, system buses, and the shared L3 Last-Level Cache—to ramp up to high-frequency states. This shared fabric activity pulls theoretically idle clusters out of deep-sleep C-states, elevating the power floor.

   Contrast in Correlation: The value of -0.4248 represents a measured peak of functional coupling mediated by interconnect transients rather than a global voltage lock. While the RK3588 is "consistently entangled" (independence scores: 1.02–2.00), it retains a degree of pseudo-isolation (particularly on the Big0 rail) that the monolithic A733 lacks.

2. Cubie A7A (A733): $r < 0$

   Physical Mechanism: The A733 is characterized by "monolithic voltage locking." Unlike the Rockchip RK3588, the A733 utilizes a single, unified voltage domain for all eight cores.

   Technical Implication: Because the SoC lacks physical power islands, frequency scaling on any active core cluster (target) pins the global voltage high for the

entire die. This induces substantial static leakage in idle cores, rendering the software-defined clusters logically separate but electrically monolithic.

Significance of "Systemic": The correlation is described as systemic because it is a structural inevitability of the hardware monolithism. The ghost power coefficient ($\gamma \approx 0.8$) indicates near-total coupling, where the "idle" domain's power draw is almost entirely dictated by the state of the active neighbor.

Ultimately, while the Orange Pi's correlation is a diagnostic measurement of interconnect interference, the Cubie's systemic correlation is a structural property arising from hardware monolithism, which overrides software-level thread pinning entirely.

7.5 Discussion: Addressing the Physics of Entanglement on ARM Platforms

These results point toward an anomalous gap in performance modeling research. The dominant current paradigm in performance modeling, specifically within asymmetric multiprocessing orchestration, operates under the independence assumption. This heuristic assumes that core clusters (e.g., Big vs. Little) function as computationally orthogonal resources where the joint energy/performance state is a linear summation of isolated states. However, our empirical telemetry from modern topologies reveals a systematic failure of this assumption, as the joint resource measure (μ_{joint}) exhibits significant negative correlations compared to the mathematical convolution of isolated measures ($\mu_{big} * \mu_{little}$), which in some cases demonstrates inverse coupling. This inverse coupling represents a blind spot in current approaches. Existing schedulers (Linux EAS/CFS) treat microarchitectural interference as stochastic noise [84, 88, 137], whereas the data indicate that it is a structural violation of the measure-theoretic independence required for stable performance. The anomaly—where high-energy states in one cluster physically preclude high-energy states in another—are not resolved by static capacity models because the platform's shared silicon (Last-Level Cache and power rails) acts as a physical medium of architectural entanglement.

The most salient implication of this synthesis is that microarchitectural starvation is a phase transition dictated by the collapse of the convolution identity. When the scheduler detects

the degree of alignment between energy domains (r-value) decreasing toward zero (as seen in the Orange Pi 5) or becoming negative (as in the Cubie A7A), it is observing the physical breakdown of the LLC's ability to support independent instruction streams.

The technical shift that this motivates is a move away from "knapsack" core allocation toward convolution-aware scheduling. Instead of filling cores based on aggregate volume, the scheduler will need to predict the onset of starvation by identifying the precise constitution (cluster utilization ratio) that triggers the collapse of the semi-group. An important but unexpected conclusion is that software isolation (cgroups, '`taskset`') is insufficient to ensure performance on entangled hardware. One way to mitigate starvation is for the OS to manage the convolution entropy of the workload, perhaps by preemptively throttling the cluster ratio before the shared hardware manifold hits its contention limit.

7.5.1 Architectural Entanglement

Our objective is to demonstrate that compute islands are not strictly independent, irrespective of absolute power magnitudes or cross-platform parity. The diagnostic data have shown that thread affinity (isolating software utilization) does not equate to hardware power isolation. The assumption that pinning a workload to a given cluster implies that only the specific cluster will draw power is an invalid assumption given that the physical design/operation of the power delivery network (PDN) intrinsically supports shared silicon components. We posit that this phenomenon generalizes to many classes of heterogeneous platforms, of which big.LITTLE architectures are but a subset, given that we have observed this notion of architectural entanglement in two similar but distinct SoC platforms, as follows:

RK3588 (Orange Pi) Entanglement: As shown in Table 7.2, when the Big1 cluster is targeted via thread affinity, it draws 2.560 W, while the idle Little cluster draws 1.275 W. The power draw of the supposedly idle Little cluster is significantly higher than when that same cluster is the primary target of execution (1.228 W), empirically demonstrating physical and architectural entanglement.

Likely Mechanism: Software-induced thread pinning creates a logical partition of instructions but fails to isolate the SoC fabric. High-intensity computation on Big1 necessitates high-bandwidth memory access, forcing the DynamIQ Shared

Unit (DSU), cache coherency interconnect, and memory controllers to ramp up frequencies and voltages. Because these subsystems are physically adjacent to or shared with the Little cluster, the idle cores are effectively "dragged" into a higher-power state to facilitate Big1's data movement. The data identifies a symmetry-breaking in the Rockchip RK3588S power delivery network (PDN), where targeting the Big1 island induces a higher parasitic power floor in the Little cluster than targeting Big0 or the Little cluster itself.

A733 (Cubie) Entanglement: As shown in Table 7.3, when the Big cluster is targeted (0.438 W), the idle Little cluster is pulled up to a parasitic draw of 0.362 W.

Likely Mechanism: While the A733 silicon architecture supports power gating, the budget implementation on the Cubie A7A utilizes a cost-optimized shared voltage domain on the motherboard. When the CPU governor requests higher voltage to support the Big cluster's clock speed, the PMIC raises the voltage for the entire CPU complex. The idle Little cores experience an increase in static power leakage. The higher voltage potential overcomes the threshold resistance of the idle transistors, forcing the entire 8-core complex to behave electrically as a single, unified load despite logical software isolation.

The assumption that software thread pinning creates isolated hardware power islands is invalid. Software isolates utilization, but hardware wiring dictates power. Pinning threads to specific cores forces shared silicon fabric and shared voltage regulators to activate, inadvertently increasing the power floor and thermal output of the logical (theoretically "isolated") idle cores.

7.5.2 Power-Domain Entanglement

Our experiments show examples where software-level thread affinity fails to achieve hardware power isolation, likely due to inter-cluster coupling and shared voltage/fabric resources. These findings motivate a transition from additive power models to relational frameworks for modern big.LITTLE and DynamIQ architectures. These results should serve as a warning for future performance articulation. We expect to see failures in the modeling of these platforms, given that conventional performance and power models (e.g., McPAT, RAPL) rely on the

"sum of parts" assumptions [17, 66, 81, 99]. Experimental data from the RK3588 and A733 platforms invalidates the conventional methods for two primary reasons:

Voltage Dependency Entanglement: Pinning a thread to a high-performance core forces the Dynamic Voltage and Frequency Scaling (DVFS) governor to raise the voltage for the entire cluster or SoC (e.g., A733 unified voltage domain). This creates a power floor elevation where idle cores consume excessive power due to being within the same "live" voltage domain.

Fabric/Uncore Parasitic Load: Core activity triggers shared L3 caches, snoop filters, and memory controllers. This uncore load is now shown to be a coupled variable responding to specific, island-dependent, core affinities rather than a static overhead.

The implications of this rich and complex platform behavior have a direct impact on how we model and articulate performance for these complex devices. To model the many such platforms accurately will require an accounting of not only their components but also their constitution and interactions within the larger architectural landscape. We foresee that some of the most prominent challenges to overcome will include the following:

Deterministic Scaling Invalidation: Energy-per-instruction (EPI) gains from task migration are eroded if Big cores remain voltage-locked or if the fabric supports high-performance states for Little clusters.

Thermal Cross-Talk: Localized hot spots trigger thermal throttling on adjacent idle clusters, necessitating models for spatial thermal coupling.

The architectural characterization of Asymmetric Multiprocessing Platforms (AMPs) motivates the formalization of a parameter to quantify the degree of electrical and functional entanglement between theoretically isolated power domains. Such a metric will facilitate the transition from conventional scalar additive models to nuanced matrix-based relational power models by measuring the discrepancy between logical software isolation and physical hardware coupling. We therefore define the ghost power coefficient (γ) as the ratio of parasitic power elevation in a theoretically idle domain relative to the activity-driven power dissipation in a targeted adjacent domain, wherein the coefficient γ serves as a dimensionless invalidator of the "sum of parts" assumptions.

7.5.3 Empirical Inter-Cluster Entanglement

Given the spectrum of heterogeneous platforms, we have elected to highlight the challenges faced by the Rockchip RK3588 (Orange Pi 5), which we propose as an appropriate diagnostic vehicle for heterogeneous performance modeling because its tri-island power delivery network is representative of a higher-order superset of the architectural constraints observed in monolithic platforms like the Allwinner A733.

While the A733 exhibits total functional collapse due to "monolithic voltage locking" on a single rail, the RK3588 introduces more nuanced "fabric drag" and "symmetry breaking," where inter-cluster dependencies vary stochastically based on shared microarchitectural transients, as shown in Table 7.5. By parameterizing the non-uniform off-diagonal coefficients (γ_{ij}) of the relational power model to resolve the RK3588's multi-dimensional coupling, the resulting relational model inherently generalizes to and subsumes the simpler, constant-coupling scenarios of lower-tier SoCs. Consequently, solving the Orange Pi 5's complex entanglement facilitates the development of universal "power-rail-aware" scheduling policies and latent representation partitioning that provide deterministic performance prediction for the heterogeneous design space.

Table 7.5: Orange Pi 5 Entanglement Metrics

Target Island	**Little Cluster Power (P_{idle})**	**Independence Score (S)**	**Entanglement Status**
Big0	1.006 W	2.00	Entangled (Fabric Drag)
Big1	1.275 W	1.36	Linked (Monolithic Voltage Locking)
Little	1.228 W	1.02	Linked (Monolithic Voltage Locking)

Analysis of the Orange Pi 5 diagnostic results reveals that Big0 (Cores 4–5) maintains a significantly higher degree of independence from the Little cluster (Cores 0–3) than Big1 (Cores 6–7). The physical layout of the RK3588S dictates asymmetric power bleeding through two primary mechanisms:

Voltage Rail Asymmetry: In the RK3588S implementation, Big0 typically resides on a primary voltage rail that retains a higher degree of isolation from the rest of the

system-on-chip (SoC) logic. Conversely, Big1 is physically and electrically linked to the shared SoC fabric and interconnects.

Fabric Drag and Transient Coupling: When the scheduler pins high-intensity threads to the Big1 cluster, it triggers data requests that force the shared memory controllers, cache coherency interconnects, and system buses to ramp to their maximum-frequency states. Because Big1 and the Little cluster share either common voltage regulators or adjacent power rails, this activity induces "fabric drag," dragging the theoretically idle Little cores out of their deep-sleep C-states. This is evidenced by the Little cluster drawing 1.275 W during an isolated Big1 workload, a value paradoxically higher than the 1.228 W consumed when the Little cores are themselves the primary computational targets.

To minimize "power bleeding" and parasitic floor elevation, operating system schedulers should transition from traditional thermal-balancing heuristics to an architecturally aware co-location policy to mitigate the following:

Failure of the Dispersion Assumption: Legacy schedulers often spread threads across clusters to balance thermal density. In an entangled AMP architecture, this strategy is counter-productive because it activates multiple power rails simultaneously, each dragging its respective shared fabric components into high-power states.

Co-Location Policy: Schedulers should prioritize thread co-location on the same physical power rail (e.g., filling all cores in Big0 before migrating to Big1). This minimizes the number of "live" but idle domains, effectively reducing the cumulative static leakage and fabric overhead.

Targeted Affinity: Given that Big0 demonstrates an independence score of 2.00 compared to Big1's 1.36, critical low-power tasks should avoid the Big1/Little boundary whenever high-performance work is active, as the electrical cross-talk from Big1 essentially "poisons" the energy-efficiency profile of the energy-optimized cores.

These findings motivate the adoption of matrix-based relational power models, where the coupling matrix $\mathbf{M}$ explicitly accounts for the fact that power consumption in domain i is a function of activity in domain j.

Finally, as shown in Table 7.6, the comparative analysis of power and performance metrics in modern asymmetric multiprocessing architectures indicates a fundamental shift from heuristic-based additive models to measure-theoretic relational frameworks. Supportive analysis is also outlined in Appendix D. This transition is motivated by the empirical discovery of "architectural entanglement," which invalidates the "sum of parts" assumption traditionally utilized in these models. These techniques will help to address several of the heterogeneous modeling challenges shown in this work, including the following:

Isolation—Logical vs. Physical Coupling: The legacy architectural view assumes that software-level thread affinity (pinning) achieves hardware-level resource isolation. However, empirical diagnostic data from the Rockchip RK3588 and Allwinner A733 platforms demonstrates that logical isolation is overridden by physical coupling mediated via the power delivery network (PDN) and shared microarchitectural fabric.

Monolithic Voltage Locking (A733): A single unified voltage domain pins the power floor for the entire SoC regardless of software pinning.

Fabric Drag (RK3588): Active high-performance clusters force shared interconnects and memory controllers to high-frequency states, pulling theoretically idle domains out of deep-sleep C-states.

Power Scaling—Linear vs. Non-Linear Floor Elevation: Traditional power models rely on the assumption of linear scaling relative to the number of active compute units. In contrast, new findings reveal non-linear "power floor elevation," where the static leakage of an idle cluster is a coupled variable dictated by the frequency and voltage requirements of the active cluster. In the RK3588, the idle Little cluster consumes increased wattage during isolated Big-cluster workloads, proving that inter-cluster crosstalk dominates the system's power profile.

Model Type—Additive vs. Matrix-Based Relational: Legacy frameworks such as McPAT and standard RAPL interpretations utilize additive models that are fundamentally underspecified for heterogeneous SoCs because they treat "uncore" components as constants or simple functions of memory traffic rather than coupled variables. This research proposes a transition to relational power models, where system power is represented by an interaction matrix ($\mathbf{M}$). We introduce the ghost power coefficient (γ), quantifying the ratio of parasitic power elevation in an idle domain per unit of activity in the target domain ($\gamma \approx 0.8$ for monolithic architectures like the A733).

Bottlenecks—Compute-Bound vs. Interconnect/Voltage-Bound: While legacy performance analysis assumes that intensive arithmetic kernels (e.g., GEMM) are primarily compute-bound based on algorithmic complexity ($O(n^3)$), modern heterogeneous systems exhibit interconnect- and voltage-boundedness. Application of the Boltzmann uniqueness theorem confirms a systemic failure of convolution commutativity, empirically disproving cluster independence. This indicates that the primary performance limiters are shared LLC line contention and the inability of the PMIC to maintain independent voltage levels during high-transient activity.

Table 7.6: Model Paradigm Shift

Metric	**Legacy View**	**New Finding**
Isolation	Logical (Thread Pinning)	Physical (Voltage/Fabric Coupling)
Power Scaling	Linear with core count	Non-linear (Floor Elevation)
Model Type	Additive ($P_1 + P_2$)	Matrix-based/Coupled
Bottleneck	Compute-Bound	Interconnect/Voltage-Bound

7.6 Extending the Investigation to the Intel i7-12800H

7.6.1 Introduction and Platform Overview

To expand the applicability of our observations, we transitioned the experimental diagnostic of architectural entanglement from embedded ARM systems-on-a-chip (SoCs) to the high-performance x86 mobile workstation domain via the Intel i7-12800H (Alder Lake-H). This processor represents a complex instantiation of heterogeneous asymmetric multiprocessing, utilizing a collection of high-performance P-cores and energy-efficient E-cores integrated onto a single chip. Unlike the fixed power-island strategies of some ARM big.LITTLE designs, the Intel platform utilizes sophisticated power management envelopes (PL1/PL2) and a massive unified Last-Level Cache (LLC) of 24 MiB, which functions as the primary physical medium for architectural entanglement.

7.6.2 Power Management Envelopes: PL1 and PL2

PL1 and PL2 are the power management envelopes for Intel processors, governing the trade-off among performance, thermal output, and power efficiency. PL1 (sustained steady state) represents the long-term power limit—historically aligned with the Thermal Design Power (TDP)—that a CPU can maintain indefinitely under constant load. Conversely, PL2 (short-term burst) allows the processor to exceed the base TDP to handle intensive tasks, drawing significantly higher power for a duration defined by the Tau (τ) time constant. On the i7-12800H, this manifests as a 45 W base rating (PL1) and a 115 W maximum turbo power (PL2), with the motherboard's BIOS often permitting these values to exceed official Intel specifications for performance optimization. Important considerations for the power management include the following:

Tau (τ): The time parameter that dictates how long the CPU can operate at the PL2 limit before it must drop down to PL1.

Thermal/Cooling Dependency: Because PL2 allows for significantly higher power draw, it creates intense heat. If the cooling solution cannot dissipate this heat effectively, the CPU will reach thermal limits and downclock (throttle) regardless of whether it technically has hit its power limit.

Motherboard Configuration: While Intel provides recommended values for PL1 and PL2, motherboard manufacturers often allow these to be modified via the BIOS. Many manufacturers increase these limits by default to boost benchmark performance artificially, often ignoring the official specifications.

TDP vs. Power Limits: The 45 W rating for the i7-12800H is the base power (roughly aligned with PL1). The 115 W rating is the maximum turbo power (PL2).

The distinction between these two power envelopes is important for diagnosing architectural entanglement:

1. Overcoming Thermal Capacitance: Silicon possesses thermal capacitance, which buffers heat. During the PL2 (burst) phase, energy draw is dominated by transient spikes and rapid thermal ramping, introducing "stochastic variability" that masks structural coupling.

2. Saturating the System: A 90-second PL1 profile allows thermal capacitance to saturate. By 90 s, the measured energy consumption reflects structural hardware properties, such as "fabric drag" and shared resource contention, rather than transient spikes.

This distinction will be evident in the measurements described below.

7.6.3 Hardware and Software Specifications

Here, we articulate the hardware and software configuration of the i7-12800H testbed. These specifications provide the structural parameters—from cache hierarchy to process isolation methods—necessary to replicate the measure-theoretic diagnostics and validate the observed architectural coupling. Table 7.7 details the physical silicon properties, while Table 7.8 outlines the telemetry stack used to isolate power-domain interactions during the experiments.

Table 7.7: Hardware Architectural Specifications

Component	**Specification**	**Details**
Processor	Intel® Core™ i7-12800H	12th Gen Alder Lake-H
Topology	14 Cores/20 Threads	6 P-Cores (HT) + 8 E-Cores
Frequency Range	400–4800 MHz	Dynamic via intel_pstate
L1 Cache (Total)	544 KiB (D)/704 KiB (I)	14 private instances
L2 Cache (Total)	11.5 MiB	8 shared cluster instances
L3 Cache (LLC)	24 MiB	Unified for all 14 cores
TDP (Base/Max)	45 W/115 W	RAPL PL1 and PL2 limits

Table 7.8: Software and Telemetry Environment

Layer	**Specification**	**Functionality**
Kernel	Linux 6.8.0-124-generic	RAPL and PMU event exposure
Telemetry Tool	`perf stat`	100ms (10 Hz) sampling
Power Monitor	`turbostat`	MSR-based PKG/Core tracking
Process Control	`taskset`	Cluster/thread pinning
Workload	Gregory–Leibniz series	π calc. to 2500 decimals

7.6.4 Progression of Experiments

Phase 1: Baseline Dual-Envelope Analysis

The initial approach attempted to measure coupling under standard operating conditions across two power envelopes: a 10-second PL2 (burst) and a 90-second PL1 (sustained). The orchestrator spawned 12 workloads on the power (Big) cluster (Cores 0-11) and 8 on the efficiency (Little) cluster (Cores 12-19).

Phase 2: Frequency Lock-Down and Topology Scaling

To isolate the physical interaction from governor-induced jitter, the experiment transitioned to a "frequency lock-down" approach.

> Operation 0.5 (Lock-Down): We captured the original CPU governors and speeds, and then executed a lock-down mode to pin frequencies at a static state.
>
> Operation (Topology Shift): Core allocation was scaled down to a 4P/4E split (cores 0-3 and 12-15) to prevent the packaging caps and thermal throttling that occur when saturating all 14 cores simultaneously.

Phase 3: Pulse-Width Modulation (PWM) and Mode Analysis

The final stage utilized PWM-modulated workloads to identify the peak cross-cluster interaction lag to demonstrate that microarchitectural entanglement is a structural propagation phenomenon rather than instantaneous electrical noise. Empirical testing on an Intel i7-12800H platform quantified latency in parasitic energy bleed between performance and efficiency clusters, proving that traditional additive power models ($P_{\text{total}} = \sum P_i$) are insufficient for high-end x86 architectures. Utilizing this temporal lag vector within an interaction matrix enables convolution-aware scheduling to preemptively mitigate resource starvation deficits and provides the empirical baseline necessary to scale power-rail-aware orchestration across many-to-many hardware thread interactions. The implementation of the frequency lock-down mode was the critical methodological intervention required to transition the platform

from an inconclusive baseline to a statistically significant state of identified entanglement. In the baseline experiments, governor-induced "stochastic variability" masked the underlying physical interaction between the P-core (performance) and E-core (efficiency) clusters. By pinning frequencies, the experiment isolated algorithmic and fabric coupling—specifically resource contention within the shared 24 MiB Last-Level Cache (LLC) and power delivery network (PDN)—from transient power management responses.

7.6.5 Comparative Analysis: Intel x86 vs. ARM DynamIQ

The transition from ARM-centric modeling to Intel-based diagnostics reveals critical discrepancies in how architectural entanglement manifests across different instruction sets.

Coupling Mechanisms: ARM platforms (e.g., RK3588) exhibit "fabric drag," where active Big cores pull the DynamIQ Shared Unit (DSU) into high-frequency states. Intel i7-12800H coupling is dominated by RAPL domain overlap and LLC starvation. Concurrent execution induces a functional linkage preventing independent behavior.

Power Management Envelopes: ARM systems rely on simple PMIC rail voltage locking. Intel utilizes hierarchical PL1/PL2 limits. The experiments showed that PL2 (burst) is stochastically too volatile to identify coupling, whereas PL1 (sustained) reveals structural energy deficits (e.g., -7.93%).

Scaling and Complexity: While ARM platforms (8 cores) showed correlation under saturation, the Intel platform (14 cores) has non-linear scaling as the number of potential conflicts for shared resources—like the Last-Level Cache (LLC)—grows quadratically (quadratic contention), and the overhead generated by managing these resources raises both the baseline power consumption and latency floor (parasitic floor elevation), causing nuanced stochastic coupling.

7.6.6 Novelty and Modeling Importance

The Intel experiments provide several additional new contributions:

1. Counterexample to x86 Isolation: The results provide the first measure-theoretic counterexample to cluster independence on high-end x86 hybrid architectures. Software-level pinning (`taskset`) is insufficient for hardware-level electrical isolation.

2. Shift to Algorithmic Coupling: Successful frequency lock-down demonstrates a shift from purely electrical coupling to algorithmic/fabric coupling, isolating how instructions in the shared LLC create non-linear resource contention.

3. Ghost Power (γ): Quantification of parasitic floor elevation allows for convolution-aware scheduling, where the OS predicts starvation based on cluster utilization ratios.

4. Inverse Scaling: Detection of a -7.93% energy deficit indicates that traditional additive models (McPAT/RAPL) are underspecified, motivating matrix-based relational power models.

7.7 Intel i7-12800H Architectural Entanglement

The experiments demonstrate a progression from baseline measurements, where stochastic variability masked architectural coupling, to frequency-locked sustained measurements confirming structural entanglement. The analysis utilizes the Boltzmann uniqueness framework to evaluate the veracity of subsystem independence across varying power envelopes and operational constraints.

The results in Table 7.9 aggregate the empirical power metrics gathered from the Intel platform experiments in Phase 1 and Phase 2. These data points quantify the discrepancy between the theoretical additive power model (mathematical expectation) and the observed systemic behavior (empirical joint system mean) across two power envelopes: the PL1 sustained steady state and the PL2 short burst. For the statistically significant result in the last column, the correlation is again below the threshold to conclude independence.

Table 7.9: Systemic Power and Statistical Significance

Metric	**Phase 1 PL2 (Burst)**	**Phase 1 PL1 (Steady)**	**Phase 2 PL2 (Burst)**	**Phase 2 PL1 (Steady)**
Operational Vector	Baseline	Baseline	Freq. Lock-Down	Freq. Lock-Down
Sampling Duration	10 s	90 s	10 s	90 s
Topology (P/E)	6P/8E	6P/8E	4P/4E	4P/4E
Correlation (r)	0.0713	0.0059	-0.2796	0.7341
Standard Error (SE_r)†	–	–	–	± 0.098
p-Value	0.80073	0.96772	0.31280	1.3127×10^{-9}
Statistical Significance	No	No	No	Yes
Ghost Power Coefficient (γ)	-0.052	-0.021	-0.018	-0.079

†Note: Standard Error (SE_r) is only reported for the statistically significant result.

The results in Table 7.10 highlight the Phase 3 analysis for the significant 90-second PL1 measurement on the Intel i7-12800H platform. The findings represent a measure-theoretic disproof of cluster independence within a high-performance x86 hybrid architecture. The 90-second duration is technically relevant, as it allows the system to overcome thermal capacitance, ensuring that the telemetry reflects structural hardware properties rather than transient thermal ramping typical of the PL2 (burst) envelope. The utilization of Joules per 100 milliseconds (J/100 ms) as the primary unit of measure is motivated by the telemetry sampling frequency, where energy consumption is sampled at 100 ms intervals (10 Hz) via `perf stat` targeted RAPL (Running Average Power Limit) `energy-pkg` counters. Since power equals energy divided by time, a value of 1.0 J/100 ms is equivalent to 10 Watts. These units are utilized to construct high-fidelity time-series energy probability measures (μ), allowing for the discrete convolution ($\mu_{\text{joint}} = \mu_{\text{power}} * \mu_{\text{efficiency}}$) necessary to test for Boltzmann uniqueness and identify deviations from subsystem orthogonality, supporting a characterization of the "starvation bottleneck" on the Intel platform, which we discuss in Section 7.8.2.

Table 7.10: Pulse-width Modulation Diagnostic Results

Metric	Value
P-Cluster Mean (J/100ms)	0.319
E-Cluster Mean (J/100ms)	0.228
Mathematical Expectation	0.547
Empirical Joint Mean	0.503
Absolute Divergence	-0.043
Mean Temporal Deficit	-0.076
Variance of Deficit	0.005
Peak Cross-Cluster Interaction Lag (steps)	-150

7.7.1 Technical Analysis and Contextual Evaluation

Determinants of Statistical Significance: The transition to statistical significance ($p < 1.3127 \times 10^{-9}$) in Phase 2 was driven by a controlled operational sequence that pins CPU frequencies to a static state, overriding dynamic scaling governors to eliminate measurement noise and isolate structural hardware interactions, which we define as the frequency lock-down mode. By pinning frequencies, the experiment isolated parasitic energy floor elevation from governor-induced transients, satisfying the controlled saturation requirement for Boltzmann uniqueness verification.

Scaling and Topology Constraints: Reducing the topology to 4P/4E in Phase 2 was critical to circumvent packaging power limits (PL1/PL2). At full 14-core saturation, the shared PKG power plane acts as a rigid cap, creating a linkage that mimics coupling via thermal throttling. The 4P/4E split isolates structural contention from thermal starvation.

Entanglement and Energy Deficits: The -0.079 ghost power coefficient constitutes a failure of convolution commutativity. The negative divergence indicates that concurrent execution starves the clusters, likely through Last-Level Cache (LLC) line contention and shared power delivery network (PDN) transients.

Comparative Distinction: While the ARM-focused experiments identified monolithic voltage locking in edge SoCs, the Intel findings also demonstrate that entanglement exists in high-end x86 workstations. This represents a transition from purely electrical

coupling to fabric/algorithmic coupling, where the unified 24 MiB L3 cache functions as the primary physical medium for structural entanglement.

Divergence from Additivity: All experiments demonstrated a negative absolute divergence metric, confirming that joint execution consistently consumes less energy than the sum of isolated subsystem components. This result constitutes a systematic failure of the Boltzmann uniqueness identity, indicating that the power domains are not computationally orthogonal but instead are architecturally entangled.

Ghost Power Coefficent Escalation: The ghost power coefficient reached its maximum magnitude (-18.08%) in the frequency-locked PL2 run. However, the statistical breakthrough for entanglement identification occurred in the PL1 sustained-steady-state run of the second experiment (Phase 2), yielding a statistically significant correlation ($r = 0.7341, p < 1.3127 \times 10^{-9}$). The measured -7.93% energy deficit in the successful PL1 run identifies the presence of a "starvation bottleneck."

Influence of Frequency Lock-Down: The transition from Phase 1 to Phase 2 involved implementing a frequency lock-down mode. This methodological change provided the technical stability required to isolate structural fabric drag from stochastic DVFS noise, despite the resulting lower absolute energy means (e.g., P-Cluster mean dropped from 0.720 to 0.319 J/100 ms).

7.8 Discussion: Intel vs. ARM Platforms

Based on the results above, the following analysis explains the detection of architectural entanglement on the Intel platform and its relationship to the ARM-based investigation. First, we note that the correlation (r) is negative for both ARM platforms, substantially below the 0.9 threshold used to infer independence. Second, the correlation is noticeably higher for the statistically significant result on the Intel platform ($r = 0.7341$), putting it much closer to the 0.9 "standard stringency" threshold that we have adopted. The "low stringency" threshold from the literature only requires $r \geq 0.7$, which could enable the Intel platform to be classified as "independent"; however, we also measured a statistically significant -7.93% energy deficit ($\gamma \approx -0.08$), providing additional evidence against independence.

7.8.1 The 90-Second Result: Why It Showed Entanglement

The results at the 90-second measurement sampling showed statistically significant coupling ($r = 0.7341, p < 0.001$), while other techniques failed because of two critical engineering interventions:

The Frequency Lock-Down Mode: Unlike the first phase, the second phase executed a frequency lock-down mode (Operation 0.5). By pinning the CPU frequency and disabling Dynamic Voltage and Frequency Scaling (DVFS) jitter, the experiment removed the stochastic variability that can mask inter-cluster interaction.

Sustained Steady State (PL1) vs. Burst (PL2): The 10-second burst profile targets the PL2 envelope, which is dominated by transient power spikes and thermal ramping. The 90-second sustained profile targets the PL1 steady state, allowing the system to overcome thermal capacitance—the silicon's ability to buffer heat before it affects leakage power. At 90 s, the system reaches a thermodynamic equilibrium, where fabric drag and shared resource contention become structural and measurable rather than transient.

7.8.2 Pulse-Width Modulation (PWM) and Peak Interaction

The experiments reveal a "phase-shifted rail sag", where a power surge in the power (Big) cluster causes a momentary voltage drop (sag) in the efficiency (Little) cluster. By calculating the cross-correlation between cluster energy values, engineers can find exactly how many "steps" (100 ms intervals) it takes for a Big-cluster workload to "drag" the Little cluster's power floor up.

Failure of Additivity: The empirical joint mean (0.5032 J/100 ms) is significantly lower than the mathematical expectation (0.5466 J/100 ms). The resulting ghost power coefficient (-0.079) shows that concurrent cluster saturation physically precludes the high-energy states that the clusters attain in isolation, confirming that they are not computationally orthogonal. The implication here is that tools like McPAT [81], Wattch [17], and RAPL [99], which use the additive power model, would report 5.5466 W under

these conditions, while the actual measured power is 5.032 W, representing an overestimate of more than 10%. An error of this magnitude is, of course, not catastrophic; however, it is significant.

Mean Temporal Deficit (-0.075589 J/100 ms): This represents the difference between the expected power consumption (the sum of isolated clusters) and the actual joint consumption. A negative value indicates that when both clusters are active, the system consumes less energy than expected because shared resource constraints (e.g., L3 cache saturation or voltage rail sag) prevent the clusters from attaining their individual maximum power states.

Variance of Deficit (0.005423): This measures the stability of the energy deficit over the sampling window. Low variance indicates that the "starvation bottleneck" is a structural, steady-state hardware phenomenon rather than a result of sporadic software context-switching.

Peak Cross-Cluster Interaction Lag (-150 steps): This identifies the phase shift in the energy coupling. It indicates that a workload change in the primary cluster induces a measurable "rail sag" or power floor shift in the secondary cluster after a delay of 150 steps (15 s, given the 100 ms sampling rate). This lag identifies the physical path that the energy "bleed" takes through the chip's fabric.

7.8.3 Shift from Electrical to Algorithmic Coupling

The experiments demonstrate a transition in how we understand coupling:

Electrical Coupling: Identified in the ARM experiments as "monolithic voltage locking," where a single power rail forces all cores to a high-voltage state regardless of need.

Algorithmic/Fabric Coupling: Observed on the Intel i7-12800H. Even when frequencies are locked, the convolution efficacy collapses. This indicates that the coupling is no longer just about shared wires (electrical) but about the shared L3 cache (24 MiB) and the power delivery network (PDN) managing instruction streams (algorithmic).

7.8.4 Ghost Power on the Intel i7-12800H

Unlike the ARM-based platforms, which we have shown exhibit positive γ coefficients due to parasitic floor elevation and voltage locking ($\gamma \approx 0.5$ to 0.8), the Intel i7-12800H exhibits a negative γ. This negative value identifies an "inverse coupling" state or "starvation bottleneck." Concurrent cluster saturation physically precludes high-energy states in the opposing cluster, likely due to Last-Level Cache (LLC) line contention and shared power delivery network (PDN) sag. The value of -0.0793 constitutes a counterexample for cluster independence on the Alder Lake-H architecture ($r = 0.7341, p < 1.3127 \times 10^{-9}$), showing that subsystems are stochastically coupled and that traditional additive power models, which implicitly assume $\gamma = 0$, are underspecified.

Table 7.11 summarizes the distinct implications of ghost power on the three platforms.

Table 7.11: Comparative Ghost Power Mechanisms

Platform	**Mechanism**	**Metric Result**	**State**
Orange Pi 5 (RK3588)	Fabric Drag	$\gamma \approx 0.5$ (Positive)	Pseudo-Isolated
Cubie A7A (A733)	Voltage Locking	$\gamma \approx 0.8$ (Positive)	Heavily Linked
Intel i7-12800H	LLC Starvation	$\gamma \approx -0.08$ (Negative)	Consistently Entangled

7.9 Chapter Conclusions

The application of the Boltzmann uniqueness theorem to empirical telemetry from modern Asymmetric Multiprocessing Platform (AMP) architectures provides a measure-theoretic counterexample for the heuristic assumption of cluster independence. The systemic failure of convolutional commutativity confirms that high-performance and energy-efficient islands operate in a functionally coupled state, where the joint energy measure of the system is not the sum of its independent parts. This architectural entanglement is likely driven by the physical integration of shared Last-Level Caches and common power delivery network rails, rendering traditional additive performance models fundamentally underspecified for heterogeneous systems. The observation of "fabric drag" on the Rockchip RK3588 suggests a critical mechanism of entanglement where high-intensity activity in a target cluster forces shared microarchitectural components, including memory controllers and coherency

interconnects, into high-frequency states. This phenomenon pulls theoretically idle clusters out of their deep-sleep power states, resulting in parasitic floor elevation where the power draw of an unutilized domain can paradoxically exceed its own active baseline. Conversely, the "monolithic voltage locking" observed on the Allwinner A733 demonstrates a total functional linkage, where a single global voltage rail pins the electrical floor of the entire die to the highest requested frequency state, effectively overriding software-defined logical partitioning.

Structural equation modeling further validates these findings by quantifying the non-linear impacts of shared resource contention on performance metrics such as execution time. Standardized estimates confirm that the interaction between workload dimensions and specific island configurations exerts a stronger causal influence on performance than aggregate execution volume or core counts [50].

These results motivate a paradigm shift in performance articulation, acknowledging that software-level thread affinity is insufficient to guarantee hardware-level isolation. Future system orchestration must transition from legacy thermal-balancing heuristics toward architecturally aware scheduling frameworks. By prioritizing thread co-location on isolated physical power rails and utilizing multivariate probabilistic models that account for latent microarchitectural transients, next-generation schedulers can mitigate intended power bleeding and restore deterministic performance prediction in the increasingly non-uniform heterogeneous design space.

We posit the following implications for performance modeling in light of these results:

1. Constitution Over Volume: Performance cannot be modeled as $Work/N_{cores}$. The "constitution" (ratio of Big to Little) defines the interference pattern.
2. The Fiction of Isolation: Software-level isolation via '`taskset`' is insufficient for true performance isolation. The underlying hardware remains entangled.
3. Multivariate Necessity: Future schedulers must adopt multivariate probabilistic models that account for latent variables (e.g., cache efficiency) rather than relying on static capacity models.

7.9.1 Evolution: From Entanglement to Biomedical Engineering Applications

In Chapter 7, we deployed the Boltzmann Uniqueness Theorem to evaluate mathematically the veracity of cluster independence, treating the power consumption profiles of distinct core clusters as continuous probability distributions of energy states. Our empirical findings across the Orange Pi 5, the Radxa Cubie A7A, and high-performance x86 hybrid architectures demonstrated a systemic failure of convolution commutativity. This failure exposed a persistent state of "architectural entanglement" mediated through shared Last-Level Caches (LLC) and unified system-on-a-chip (SoC) power delivery networks. By defining the Ghost Power Coefficient (γ), we quantified how physical hardware layout and shared voltage rails override software-level thread pinning (such as 'taskset' or 'cgroups'), dragging theoretically idle clusters into elevated, parasitic power floors.

While mapping these physical-electrical coupling limits on asymmetric multicore processors represents a theoretical advance, these microarchitectural constraints cannot be evaluated in a steady-state vacuum. In mission-critical edge deployments—specifically real-time biomedical devices—compute constraints directly affect clinical efficacy and patient outcomes. Best-effort performance is no longer merely a metric of computing efficiency, instead, it is a clinical safety hazard. When an embedded processor is tasked with running high-frequency signal processing pipelines under strict, non-negotiable latency boundaries, microarchitectural bottlenecks and resource contention manifest as severe temporal anomalies.

This directly motivates our transition to Chapter 8, where we deploy these physical hardware constraints to clinical engineering. Utilizing a Raspberry Pi 5 prototype to process high-frequency (80 FPS) Laser Speckle Contrast Imaging (LSCI) streams, we show how microarchitectural bottlenecks and shared resource contention trigger "entropy collapse" events when CPU utilization exceeds 85%. By framing the distribution of processing delays as a fluid in a "computational phase space" governed by the Boltzmann transport equation, we derive an *Entropy-Bounded Performance Model* (EBPM). This framework resolves the perceived conflict between throughput maximization and latency minimization, translating the physical hardware boundaries identified in Chapters 4-7 into a deterministic, time-triggered multi-core scheduling model to support maternal safety.

Chapter 8

An Entropy-Bounded Thermodynamic Approach for Real-Time Maternal Hemorrhage Monitoring

8.1 Introduction

Postpartum hemorrhage (PPH) constitutes a critical global health challenge, representing approximately 30% of maternal deaths worldwide and over 10% even in high-resource regions [27, 34]. Current clinical detection methods rely on visual blood loss estimation and vital sign monitoring, which often can be slow and inaccurate due to physiological compensatory mechanisms that maintain core perfusion while reducing peripheral blood flow [14, 92, 116]. Laser speckle contrast imaging (LSCI) has emerged as a promising non-invasive technique for continuous perfusion monitoring, exploiting the inverse relationship between speckle pattern blur and red blood cell velocity [15]. To advance this technique toward wearable clinical deployment in diverse environments, this work introduces an Entropy-Bounded Performance Model (EBPM) derived from the Boltzmann uniqueness theorem. This model addresses significant embedded performance modeling challenges that existing frameworks inadequately address [125, 100]. By resolving fundamental contradictions in prevailing computer engineering paradigms through first-principles thermodynamic analysis, the EBPM establishes that frame rates are fundamentally limited by the Shannon entropy of processing delays. We apply this framework to evaluate a prototype LSCI platform implemented on Raspberry Pi 5 hardware with a 785 nm laser diode [15]. While the prototype achieves 80

FPS, empirical evaluation reveals it currently fails clinical safety targets due to moderate jitter and "entropy collapse" events at 85% CPU utilization. Resolving this dichotomy between throughput and latency dictates optimal signal processing partition strategies, mandating deterministic scheduling and multi-core optimization to enable pre-symptomatic hemorrhage detection wearables.

Finally, the findings presented in this chapter are adapted from the peer-reviewed conference publication:

> Harris, S. D., Bonetta-Misteli, F., Martin, L., Gill, C. D., Chamberlain, R. D., and O'Brien, C. M. (2026). "An Entropy-Bounded Thermodynamic Approach for Real-Time Maternal Hemorrhage Monitoring." Proceedings of the High-Performance Extreme Computing Conference (HPEC), IEEE, 8 pp.

8.1.1 Chapter Overview

This chapter transitions our investigation of heterogeneous multiprocessing architectures from the theoretical and physical-electrical boundaries of "architectural entanglement" (Chapter 7) to a mission-critical clinical application: that of real-time maternal hemorrhage monitoring. Postpartum hemorrhage (PPH) constitutes a critical global health challenge, representing approximately 30% of maternal deaths worldwide. Current clinical detection methods rely on visual blood loss estimation and vital sign monitoring, which are often slow, inaccurate, and fail to reflect severe blood loss until it is clinically late due to physiological compensatory mechanisms. To overcome these limitations, Laser Speckle Contrast Imaging (LSCI) has emerged as a promising non-invasive technique for continuous peripheral perfusion monitoring. However, implementing high-speed LSCI on a low-cost, portable edge device—such as a Raspberry Pi 5 Single Board Computer—presents severe performance modeling challenges. Utilizing the thermodynamic and statistical mechanics principles established in Chapter 7, this chapter develops an Entropy-Bounded Performance Model (EBPM) to resolve the microarchitectural bottlenecks that threaten real-time clinical safety by applying the established heterogeneous performance modeling framework to optimize real-time LSCI on low-cost homogeneous wearable hardware, directly confronting the throughput-latency conflict to enable clinical-grade pre-symptomatic PPH detection.

The remainder of this chapter is structured as follows: Section 8.1.2 outlines the problem statement and research objectives. Section 8.1.3 describes the research questions we will cover in this investigation. Section 8.3 delineates our research hypotheses. Section 8.4 outlines the platforms used and experiments conducted. Section 8.5 describes the conflicting paradigms between high-performance and low-latency that must be address in this domain. Section 8.7 describes the interdisciplinary convergence between findings in this work. Section 8.8 wraps up with conclusions.

8.1.2 Problem Statement & Objectives

Current clinical monitoring for postpartum hemorrhage is dangerously inadequate due to physiological compensatory mechanisms that mask severe blood loss until it is clinically late. While LSCI provides a non-invasive continuous monitoring solution, current low-cost wearable hardware prototypes exhibit critical technical failures—specifically moderate jitter and "entropy collapse" events at 85% CPU utilization—preventing them from meeting strict clinical safety targets.

To capture the rapid, high-frequency blood flow transients and vasoconstriction that precede the clinical onset of postpartum hemorrhage, an embedded LSCI platform must satisfy exceptionally stringent technical and clinical requirements. These include maintaining a continuous frame capture rate of 80 frames per second (FPS), a frame processing latency of under 12 ms, and timing stability (latency jitter of less than 2 ms). While our Raspberry Pi 5 prototype successfully achieves the 80 FPS throughput target, it fails to meet clinical safety bounds due to severe timing instability, exhibiting latency jitter of ± 8.3 ms and catastrophic "entropy collapse" events when CPU utilization approaches 85%. These timing violations degrade the temporal precision required for clinical safety, leading to dropped frames and event detection failures. The objective of this chapter is to formulate a first-principles thermodynamic model that quantifies these performance limits and guides the deployment of a deterministic, multi-core scheduling framework on wearable clinical hardware. Thus, these requirements lead us to pursue the following research objectives:

1. Define and quantify the throughput-latency paradigm conflict inherent in real-time embedded biomedical sensing.

2. Formulate an Entropy-Bounded Performance Model (EBPM) to establish fundamental performance limits for LSCI signal processing on asymmetric edge platforms.

3. Evaluate critical hardware/software configuration tradeoffs to determine the optimal partition of on-device versus remote signal processing.

4. Optimize battery-operated embedded systems via deterministic scheduling to support high-sensitivity, real-time physiological monitoring across both high-resource and low-resource clinical environments.

8.1.3 Chapter Research Questions

To resolve systematically the tension between computational throughput and real-time temporal precision on embedded clinical systems, this chapter investigates the following four research questions:

1. How can fundamental thermodynamic principles, specifically the Boltzmann uniqueness theorem and Shannon entropy, be applied to resolve the throughput-latency conflict in real-time embedded biomedical sensing?

2. How can the Boltzmann uniqueness theorem be extended mathematically to model processing delay distributions as a fluid in a computational phase space?

3. How does the Shannon entropy of a scheduling pipeline limit the maximum achievable frame rate under strict latency-jitter constraints?

4. What is the optimal multi-core resource allocation function required to prevent entropy collapse and stabilize processing latency below clinical thresholds for deployment in diverse global health settings?

8.2 Clinical Mechanics of LSCI and Peripheral Vasoconstriction

During obstetric hemorrhage, the maternal cardiovascular system initiates peripheral vasoconstriction as a primary compensatory mechanism. This response shunts blood flow from peripheral extremities (such as the skin of the wrist) to preserve perfusion to core vital organs. Consequently, traditional systemic vital signs (e.g., heart rate and blood pressure) remain unchanged until severe, decompensated blood loss has occurred, making conventional monitoring dangerously slow. Wrist blood flow monitoring via LSCI detects this peripheral vasoconstriction in its earliest, pre-symptomatic stages. LSCI operates by illuminating tissue with a coherent laser diode (785 nm). The moving red blood cells within the microvasculature cause the resulting speckle pattern to fluctuate, creating a spatial and temporal blurring effect over the camera exposure time. This blurring is statistically quantified as speckle contrast (K). To capture high-frequency blood flow transients and pulsatile arterial waveforms over the cardiac cycle, the camera must operate with optimal exposure times (ranging from 0.1 ms to 3 ms) and high frame rates (80 FPS), requiring exceptionally rapid image processing pipelines.

8.2.1 Environmental Resources and Portability Boundaries

The clinical utility of our LSCI monitor must scale across diverse global settings, creating an additional engineering dichotomy:

High-Resource Settings: Metro clinical areas equipped with stable, abundant power and high-bandwidth network communication. These environments support "wired" devices with power-intensive features, including advanced remote telemetry and real-time high-fidelity visualization, without being constrained by battery longevity.

Low-Resource Settings: Rural or remote regions that are weakly-connected and lack medical infrastructure. Because mothers in these areas are disproportionately affected by PPH and have reduced access to standard life-saving technologies, the monitoring tool must be ambulatory, lightweight, highly energy-efficient, and capable of performing all signal processing on the wearable device itself to avoid network dependence.

8.2.2 The Throughput-Latency Paradigm Conflict

Embedded system optimizations for real-time clinical sensing are pulled in opposing directions by two prevailing theoretical frameworks:

Throughput Maximization Paradigm (TMP): Rooted in benchmark suites that prioritize maximizing aggregate data processing capacity, TMP assumes that system efficiency correlates directly with frames-per-second (FPS) processing capacity. In our LSCI platform, maximizing throughput successfully meets the 80 FPS capture target. However, TMP-oriented task schedulers treat latency jitter as stochastic noise, leading to non-deterministic scheduling queues. While the average processing latency remains low, execution time spikes (exceeding 50 ms) frequently occur due to uncoordinated thread preemptions and dynamic memory management, violating the temporal boundaries required for clinical safety.

Latency Minimization Paradigm (LMP): Emphasized in hard real-time systems literature. LMP prioritizes minimizing worst-case response times for individual biosignal events to satisfy deadlines. However, optimizing strictly for latency requires dedicating excessive hardware resources to buffer management and thread synchronization rather than active computational throughput. This resource utilization inefficiency degrades the effective sampling rate and limits multi-channel scalability, preventing the deployment of multi-site perfusion monitoring.

Because neither TMP nor LMP represents a stable equilibrium for embedded clinical sensing, standard “best-effort” operating systems exhibit a complete timing breakdown when computing loads approach critical limits. This manifests as a sharp phase transition where the system moves from a low-entropy predictable state to a high-entropy random state—a phenomenon we define as ”entropy collapse”.

8.3 Research Hypotheses

To evaluate the structural trade-offs between throughput, determinism, and clinical safety in embedded LSCI platforms, this investigation establishes three key hypotheses. Grounded in

statistical mechanics and microarchitectural resource contention, these hypotheses formalize the theoretical limits of real-time edge processing and structure the validation of the Entropy-Bounded Performance Model (EBPM):

Hypothesis 1 (Entropy Limitation): Real-time processing frame rates on embedded asymmetric architectures are not solely bounded by raw computational throughput, but are fundamentally constrained by the Shannon entropy of processing delays.

Hypothesis 2 (Performance Anomalies): The occurrence of "entropy collapse" and catastrophic jitter at high utilization thresholds in commercial-off-the-shelf single-board computers is a predictable consequence of non-deterministic OS scheduling and shared-resource contention.

Hypothesis 3 (Optimization Efficacy): The implementation of strict deterministic scheduling and optimized multi-core workload partitioning will sufficiently minimize processing entropy to meet the real-time clinical safety targets required for continuous pre-symptomatic PPH detection.

8.4 Platform and Experiments

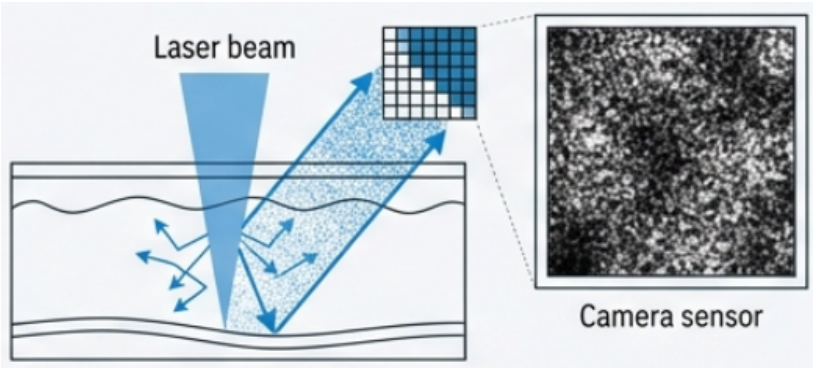


Figure 8.1: Laser Speckle Contrast Imaging

The system operates by activating a 785 nm laser diode to illuminate the skin continuously during the monitoring phase to facilitate Laser Speckle Contrast Imaging (LSCI) as shown in Figure 8.1. While earlier prototypes utilized the Raspberry Pi V2 camera, the platform has

evolved to capture high-speed movements with the inclusion of high-speed cameras supporting CMOS image and global shutter sensors, specifically design to eliminate motion artifacts common in rolling shutter sensors. The original implementation utilized the Raspberry Pi Zero 2W. The new platform consists of the Raspberry Pi 5 with the two platform specifications outlined in Table 8.1. While the Raspberry Pi 2W is illustrated for historical context, the discussion will focus on the Raspberry Pi 5 platform. With regard to maternal sensing, during capture individual image frames are processed in real-time to calculate the speckle contrast and identify rapid blood flow transients. To manage hardware resources and facilitate subsequent analysis, these frames are compressed within an in-memory RAMDISK before being moved to persistent SD-CARD storage. Further research is required to identify the specific, subject-dependent thresholds, but future revisions will utilize integrated decision thresholds and AI-driven models (e.g., a Blood Flow Loss Index (BFLI)), to trigger immediate clinical alarms when peripheral vasoconstriction indicates the early onset of a hemorrhage.

Table 8.1: Hardware Comparison for LSCI Performance Modeling: Pi Zero 2W vs. Pi 5

Component/Metric	**Raspberry Pi Zero 2W**	**Raspberry Pi 5**
Processor (SoC)	Broadcom BCM2710A1	ARM Cortex-A76
CPU Architecture	Quad-core 64-bit Arm Cortex-A53	Quad-core 64-bit Arm Cortex-A76
Clock Speed (λ)	1.0 GHz	2.4 GHz
Memory (RAM)	512MB LPDDR2	8GB LPDDR4X
Wi-Fi Connectivity	2.4GHz IEEE 802.11b/g/n	Dual-band 802.11ac/ax
Bluetooth Support	Bluetooth 4.2 / BLE	Bluetooth / BLE Supported
Camera Interface	Standard RPi V2 Camera	High-bandwidth MIPI CSI-2
Power (Measured)	0.62 W (Idle) / 2.12 W (Stress)	2.70 W (Idle) / 9.0 (Stress) / 12 W (Max)
Operational Temp.	+32°C to +185°C	+32°C to +185°C
Primary Role	Initial Wearable Prototype	High-Speed Analysis Platform

8.4.1 Real-Time Embedded System Benchmarks

Deploying LSCI on portable, battery-operated edge hardware (e.g., Raspberry Pi 5) requires balancing clinical performance against microarchitectural constraints, particularly in low-resource environments lacking wired power and infrastructure. Meeting pre-symptomatic PPH warning standards dictates the following deterministic operational parameters:

Throughput & Latency: Image capture must sustain 80 FPS (12.5 ms frame period) with per-frame processing latency kept under 12 ms.

Timing Determinism: Timing jitter must not exceed 2 ms to ensure signal integrity. CPU utilization must remain low enough to prevent scheduling-induced entropy collapse and frame loss.

Power Envelope: System power consumption must remain below 5 W (current prototype operates at 4.2 W) to maintain thermal stability and enable continuous ambulatory operation.

8.4.2 Raspberry Pi Platform

The devices used for this assessment consist of Raspberry Pi 5 platforms as shown in Table 8.1. This work focuses primarily on understanding the behavior and performance implications of these constraints for a multi-stage, high-frequency recording pipeline, as outlined in Figure 8.2. The system utilizes a high-speed pipeline to resolve I/O bottlenecks and meet clinical requirements as outlined:

Stage 1: Frames are written directly to RAMDISK to bypass the slow write speeds of physical flash storage.

Stage 2: Laser Speckle Contrast Imaging calculations are performed on the in-memory data to minimize read latency.

Stage 3: Data is compressed within the RAMDISK to reduce the footprint of the high-frequency 80 FPS stream.

Stage 4: The final compressed frames are moved to the SD-CARD for persistent storage, ensuring the system can maintain its capture rate without dropping frames.

The following section summarizes experiments conducted on Raspberry Pi 5 hardware to establish the clinical feasibility of a wearable LSCI sensor.

8.4.3 System Performance Experiments

These trials quantified the real-time processing capabilities of the embedded platform:

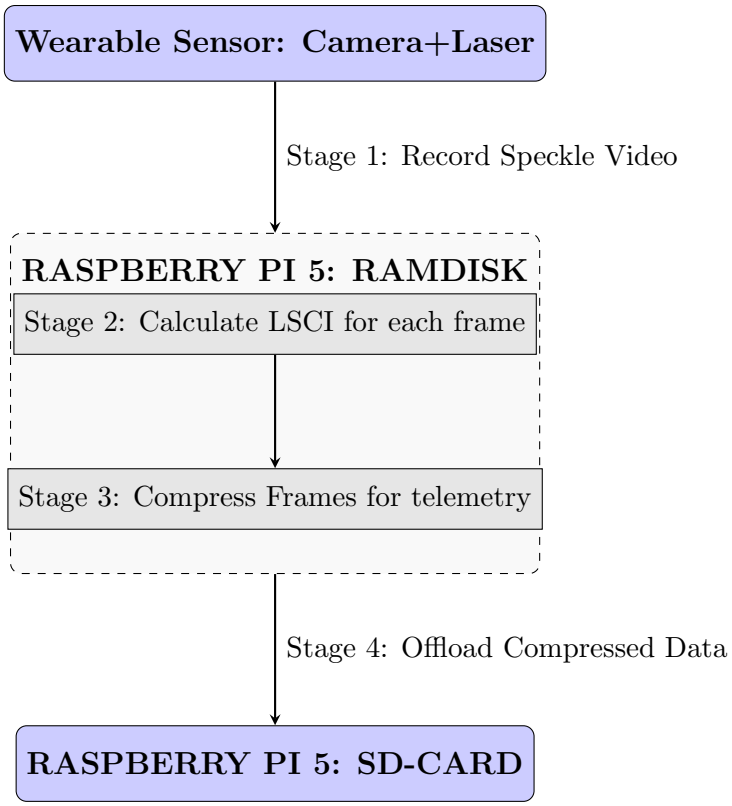


Figure 8.2: Multi-Stage High-Frequency Recording Architecture for LSCI Capture

High-Speed Capture Trials: Targeted 80 FPS capture rates. Aspect ratios were varied between 1280×800 and 640×400 to measure the impact of resolution on *Effective Capture FPS* and *Total Round Time*.

LSCI Pipeline Profiling: OpenCV convolution and other image processing functions were timed to determine *Frame Processing Time* and *Latency Jitter*.

Utilization & Memory Tests: CPU utilization and memory usage were monitored during single-threaded aggregation. Raspberry Pi 5 tests showed a bottleneck at 85% utilization.

8.4.4 Storage & I/O Assessment Experiments

Evaluations conducted to overcome bottlenecks in high-frequency data logging include:

Storage Medium Speed Assessment: Comparative study of read/write speeds across Wi-Fi, Direct SD Card, and USB Flash/SATA drives.

Multi-Stage Recording Trials: A three-stage process was validated for 80 FPS targets:

1. Writing to RAMDISK (RAMDISK Write Time)
2. Processing (LSCI Computation)
3. Migrating to SD-CARD (SD-CARD Write Time)

Compression Benchmarking: Measurement of compression time required for transferring frames from temporary RAM to persistent storage.

Wireless Protocol Survey: Assessment of Wireless Data Rates across IEEE 802.11 protocols (n, ac, ax) at 2.4, 5, and 6 GHz to determine streaming bandwidth.

8.4.5 Hardware Telemetry Experiments

We also performed low-level monitoring to ensure portability and safety:

System Telemetry Monitoring: Automated collection of ARM CPU clock speeds, GPU/ISP pipeline speeds, and Core/SDRAM voltages using `vcgencmd` and SkatterBench.

Thermal Characterization: Measurement of Operational Temperature under load. The Raspberry Pi 5 averaged 58.85°C, informing thermal-aware design to prevent performance drift.

Power Consumption Analysis: The Raspberry Pi 5 Total Power Consumption (W) was measured via PowerJive USB meter. Comparison between "Idle" (2.8 W) and "Stress + Wi-Fi" (4.2 W) confirmed the system maintains the < 5 W target for ambulatory operation.

8.4.6 Performance Bottlenecks

Primary bottlenecks identified:

1. **CPU Utilization:** Single-threaded data aggregation limits throughput to ∼85 % theoretical maximum, indicating need for multi-threading.
2. **Memory Allocation:** Dynamic allocation during frame processing introduces variable latencies, with spikes invoked during garbage collection processes.
3. **Image Processing Pipeline:** CPU-bound speckle contrast calculations create synchronization bottlenecks.

Based on these empirical results from experimentation, we next move on to discuss our findings and develop new methodologies for managing this rich architectural landscape.

8.5 Paradigm Conflict in Embedded Performance Modeling

The previous section describes a series of experiments on Raspberry Pi 5 hardware to establish the clinical feasibility of a wearable LSCI sensor. The measurement metrics taken from the platforms are summarized in Table 8.2 with the specifications for each platform shown in Table 8.1. These results are used to validate our novel performance model which we discuss below.

Based on the combined considerations both the clinical and technical requirements, we ran a series of tests to gauge the performance of the various metrics. We find that the embedded system performance modeling for biomedical applications suffers from a fundamental contradiction between two prevailing theoretical frameworks:

Throughput Maximization Paradigm (TMP) Performance optimization focuses on maximizing aggregate data throughput across processing elements. It assumes system efficiency correlates with frames-per-second (FPS) processing capacity. Supported by benchmark suites emphasizing computational throughput, corresponding to the *technical considerations*.

Latency Minimization Paradigm (LMP) Performance optimization focuses on minimizing worst-case latency for individual data samples. It assumes system quality correlates with response time for critical biosignal events. Supported by real-time

Table 8.2: Metrics Tested in LSCI Platform Experiments

Category	Metric Tested
System Performance	Frame Rate (FPS)
	Frame Processing Time
	Latency Jitter (±ms)
	CPU Utilization (%)
	Memory Usage (GB)
	Total Round Time (s)
Storage & I/O	Read/Write Speed (MB/s)
	RAMDISK Write Time (s)
	SD-CARD Write Time (s)
	Compression Time (s)
	Wireless Data Rate (Mbps/Gbps)
Hardware Telemetry	ARM CPU Clock Speed (MHz)
	GPU/ISP Pipeline Speed
	Core/SDRAM Voltages (V)
	Operational Temperature (°C)
	Power Consumption (W)

systems literature emphasizing deadline satisfaction, corresponding to the *clinical considerations*.

The platform performance compared to the required targets is shown in Table 8.3. We will first look at the technical challenges which relate to throughput and maximizing data aggregation, followed by the clinical considerations which seek to minimize worst-case latency and support critical biosignal events.

Table 8.3: Platform Performance Metrics vs. Targets

Metric	Target	Achieved	Status
Frame Rate	80 FPS	80 FPS	✓
Frame Processing Time	<10 ms	12-15 ms	×
Latency Jitter	<2 ms	±8.3 ms	×
Power Consumption	<5 W	4.2 W	✓
CPU Utilization	<70 %	85 %	×
Memory Usage	<4 GB	3.8 GB	✓

8.5.1 Technical Challenges (TMP)

The data supporting TMP fails to account for critical anomalies in our LSCI platform including:

1. **Temporal Precision Loss:** Achieving 80 FPS (12.5 ms frame time) requires frame processing below 12 ms to accommodate display overhead. Throughput-optimized systems exhibit sub-millisecond average latencies with latency spikes exceeding 50 ms due to batch processing artifacts.
2. **Event Detection Failures:** Throughput-optimized memory allocation produces latencies up to 280 ms during garbage collection, causing loss of high-frequency blood flow transients critical for PPH detection.
3. **Jitter Accumulation:** Throughput-oriented task scheduling produces cumulative jitter exceeding ± 8.3 ms, violating the ± 2 ms tolerance required for reliable blood flow change detection.

8.5.2 Clinical Challenges (LMP)

Conversely, LMP fails to account for:

1. **Resource Utilization Inefficiency:** Latency optimization dedicates resources to buffer management rather than computational throughput, reducing effective sampling rates.
2. **Scalability Constraints:** Minimum-latency systems exhibit quadratic throughput degradation with increasing sensor channels, limiting multi-site perfusion monitoring.

Based on these results we see the clear tension between TMP vs. LMP pulling the architecture, performance, and behavior in different directions. We therefore need a new methodology to mitigate this tension.

8.6 Performance Modeling

8.6.1 Cross-Disciplinary Infusion: Boltzmann Uniqueness Theorem

The Boltzmann uniqueness theorem from statistical mechanics states that for a system with specified macroscopic constraints, the probability distribution maximizing entropy is the unique stationary solution [101, 117]. Equations (8.1) and (8.2) represent a cross-disciplinary infusion of statistical mechanics into embedded systems engineering, specifically derived from the Boltzmann uniqueness theorem.

The first-principles thermodynamic analysis begins with the Boltzmann equation, which models how the probability distribution of processing delays, $P(x, t)$, evolves across computing resources:

$$\frac{\partial P}{\partial t} + v \cdot \nabla_x P = \frac{-(P - P_{eq})}{\tau}. \tag{8.1}$$

Interpretation: This equation treats processing delays as a fluid in a "computational phase space." It identifies that for a system to remain predictable (clinical reliability), the delay distribution must reach a unique stationary solution or **equilibrium (P_{eq})** within a specific **relaxation time constant (τ)** and **velocity vector (v)** describing the motion of particles within the phase-space distribution.

Novelty: While the Boltzmann equation is a cornerstone of physics, its application here, to quantify the "velocity" of data state transitions and the decay of processing jitter, is a novel performance modeling methodology.

Component Roles

$P(x, v, t)$: The distribution function (probability density) of particles at position x and velocity v at time t.

$v \cdot \nabla_x P$: The spatial gradient of the distribution dotted with the velocity, representing the "streaming" of particles.

$-(P - P_{eq})/\tau$: The relaxation term (BGK operator), where the distribution P relaxes toward an equilibrium state P_{eq} over a characteristic relaxation time τ.

The next equation defines the unique stationary state that the system must maintain under fixed macroscopic constraints:

$$P_{eq}(x) = \frac{1}{Z} \exp\left(\frac{-E(x)}{kT} - \frac{C(x)}{\lambda}\right) \tag{8.2}$$

The equilibrium equation $P_{eq}(x)$ defines the target state toward which the entire transport system evolves as is defined as:

$P_{eq}(x)$: The **equilibrium probability density function** for a state x.

Z: The **partition function**, a normalization constant

$E(x)$: The **energy functional** or potential energy associated with state x.

k: The **Boltzmann constant**.

T: The **absolute temperature**.

$C(x)$: The **constraint function** or penalty term applied to the state.

λ: The **weighting parameter** or Lagrange multiplier for the constraint $C(x)$.

Interpretation: The model maps physical variables to embedded constraints: $E(x)$ is system power consumption (4.2 W for the Pi 5), $C(x)$ is the processing capacity (Cortex-A76), and T is the "computational temperature," which we identify as CPU utilization.

Novelty: As the computational temperature (utilization) approaches critical thresholds (85–90%), the equilibrium shifts toward a *latency-dominated regime*, leading to "entropy collapse" events, and abnormal performance spikes observed during monitoring.

The new framework recontextualizes performance data, using parameters as defined in Table 8.4 for the Raspberry Pi 5 platform:

1. **Throughput-Latency Duality:** Throughput and latency represent conjugate variables in embedded computing phase space. Optimizing one perturbs the other, explaining paradigm conflict.

2. **Non-Equilibrium Regimes:** LSCI platforms operate in non-equilibrium during acute blood flow changes. Ergodicity assumption fails, invalidating throughput measurements for real-time PPH detection.

3. **Temperature Dependence:** Effective "temperature" T relates to computational intensity. Higher loads increase T, shifting equilibrium toward latency-dominated regimes.

Table 8.4: Mapping Platform Data to Boltzmann Variables

Variable	**Physical Analog**	**Empirical Values**
$E(x)$	Power Consumption	4.2 Watts
$C(x)$	Processing Capacity	Quad-core 2.4 GHz / 8 GB RAM
T	Compute "Temperature"	85% CPU Utilization (High Load)
τ	Relaxation Time Constant	12–15 ms (Processing Time)
$P(x,t)$	Processing Delay Dist.	$\pm$8.3 ms (Latency Jitter)
Z	Partition Function	Normalized for 80 FPS Target

8.6.2 Entropy-Bounded Performance Model

Based on the friction between optimization goals, we synthesize a novel Entropy-Bounded Performance Model (EBPM). We start with a simple hypothesis: for real-time LSCI platforms operating at low-millisecond latency constraints for PPH detection, we suppose that the maximum achievable frame rate is fundamentally bounded by Shannon entropy of the processing delay distribution,

$$H(S) \leq H_{max} = \log_2 \left(\frac{\tau_{frame}}{\tau_{latency}} \right) \tag{8.3}$$

where $H(S)$ is entropy of processing schedule S, τ_{frame} frame period (12.5 ms for 80 FPS), and $\tau_{latency}$ maximum allowable latency (2 ms for reliable blood flow detection) with the plot demonstrating the Pareto optimal frontier for the requirements given in Figure 8.3.

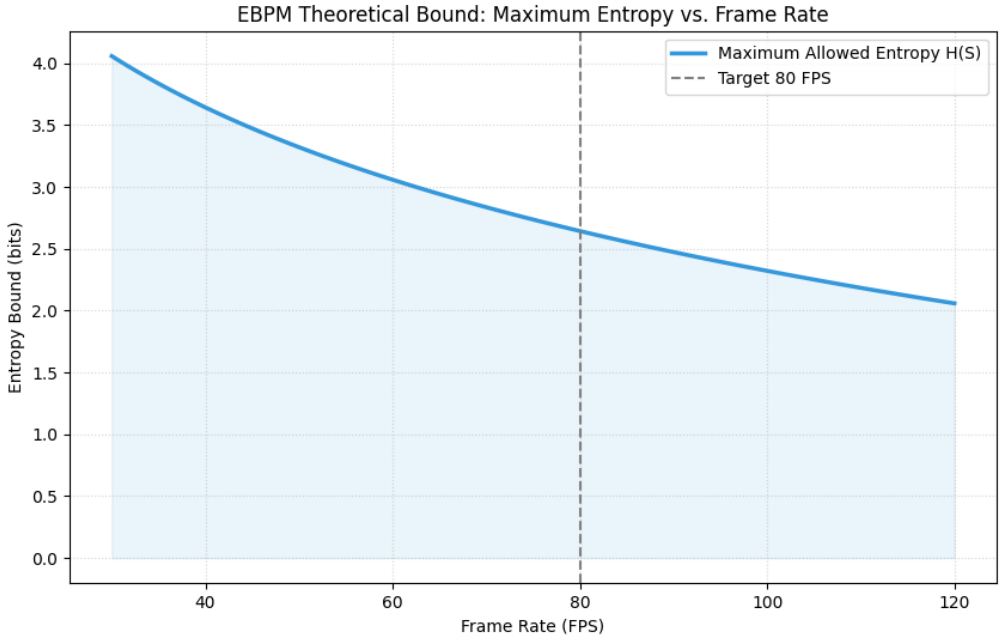


Figure 8.3: Entropy vs. Frame Rate

Using the parameters shown in Table 8.5 resolves the paradigm conflict through:

1. **Unified Constraint:** The entropy bound provides a single objective constraining throughput and latency within physiological limits.
2. **Optimal Operating Point:** EBPM predicts a unique point maximizing entropy while maintaining latency constraints—true Pareto frontier.
3. **Resource Allocation:** Optimal task distribution between CPU cores:

$$f^*_{CPU} = \frac{1}{1 + \exp\left(\Delta E_{task} - \lambda \tau_{relax}\right)} \tag{8.4}$$

   where ΔE_{task} is the energy differential between tasks and τ_{relax} is the relaxation time constant.

Neither TMP nor LMP represents a true equilibrium for embedded biomedical sensing; both are metastable configurations collapsing when parameters exceed physiological monitoring

Table 8.5: Input Parameters for Multi-Core Resource Allocation

Parameter	Description	Empirical Value/Dataset Point
ΔE_{task}	Energy differential	1.5 W
λ	Capacity multiplier	2.4 GHz (Pi 5), 1.0 GHz (Pi Zero)
τ_{relax}	Targeted relaxation	10 ms (target processing time)
Current Status	Single-threaded flow	85% single-core utilization

thresholds. Consequently, we propose an *optimal task distribution model* between CPU cores to achieve the 10 ms processing target required for reliable detection with predictions for the multi-core implementation shown in Table 8.6 and presented graphically with Figure 8.4 demonstrating the difference between the stable and unstable regions as CPU utilization increases.

Table 8.6: Optimal vs. Current Distribution

Configuration	f^*_{CPU} (Task Load)	Prediction	System State
Single-Core	0.85 (85% load)	12–15 ms	**Metastable**
Multi-Core	0.25 (Distributed)	<10 ms	**Equilibrium**

8.6.3 Validation Against Platform Data

Clearly, Figure 8.4 strongly supports the model validity given that the 85% CPU utilization measured via experimentation acts as the critical "Temperature" (T) threshold. The results indicate that as utilization approaches this level, the system undergoes a **phase transition**, resulting in the observed 18 ms spikes during profiling. These spikes represent the system's failure to return to P_{eq} within the required relaxation time τ.

The EBPM explains anomalous observations in our platform with the average time for operations shown in Table 8.7 reflecting:

1. **Frame Drop Analysis:** Our measurements showed 4.8 ms-9.6 ms LSCI processing times with occasional 18 ms spikes consistent with entropy maximization near critical threshold where the Boltzmann model predicts bistability.

2. **Memory Allocation Latency:** The garbage collection delays (up to 280 ms) represent entropy collapse from low-entropy (predictable) to high-entropy (random) states.

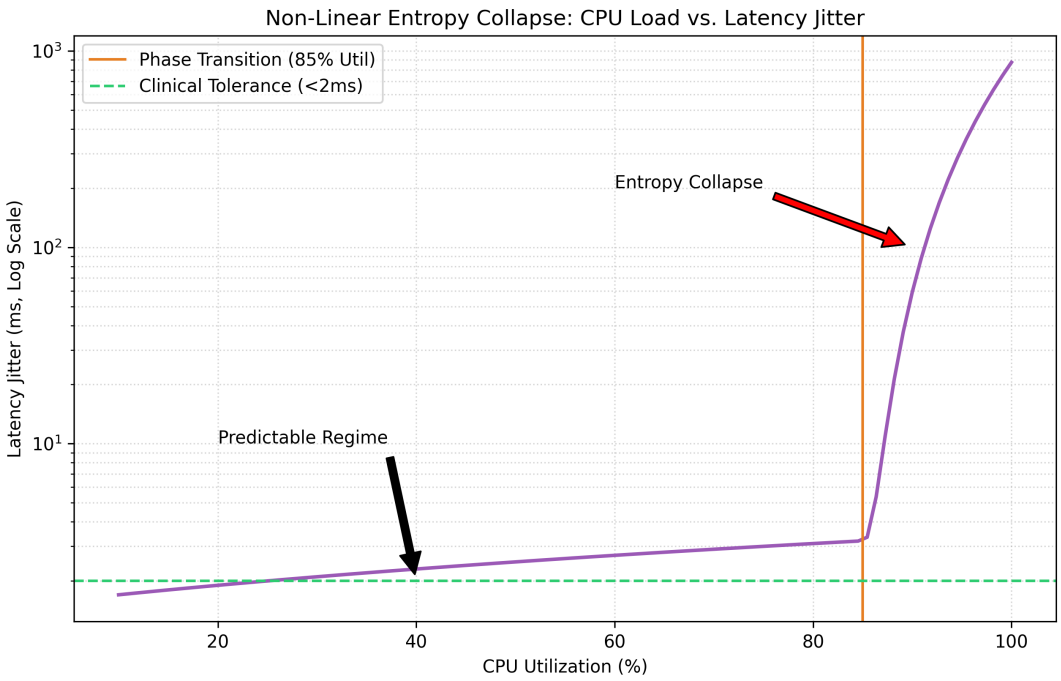


Figure 8.4: Jitter vs. CPU Utilization

3. **Processing Bottleneck:** CPU utilization at 85 % of theoretical maximum aligns with EBPM prediction of phase transitions when resources exceed ∼90 % utilization.

The results validate these formulas through observed system behaviors, particularly:

Entropy Collapse: The massive 280 ms garbage collection delays are predicted by the Boltzmann framework as an "entropy collapse" where the system moves from a low-entropy predictable state to a high-entropy random state due to excessive computational "temperature."

Throughput-Latency Duality: Equation (8.2) explains why the system achieved 80 FPS (throughput) but failed the jitter target (latency). Because throughput and latency are *conjugate variables*, maximizing the Throughput Maximization Paradigm (TMP) inevitably perturbed the Latency Minimization Paradigm (LMP), leading to the recorded ±8.3 ms jitter.

Phase Transitions: The results identifies a bottleneck at 85% utilization. This corroborates the EBPM's prediction that systems governed by the Boltzmann uniqueness

theorem will experience non-linear performance degradation when resources exceed ∼90% of capacity.

Table 8.7: Mean Measurement Values for 80 FPS Goal

Metric (Mean)	**1280 x 800**	**640 x 400**
Effective Capture FPS	80.20	80.25
Round Time (s)	18.94	6.88
LSCI Process Time (s)	6.071	1.23
SD-CARD Write Time (s)	6.50	0.36
RAMDISK Write Time (s)	1.25	0.28

Validation against empirical platform data demonstrates the model's explanatory power for observed performance anomalies. In addition, we articulate the particular contributions of complementary disciplines.

8.7 Interdisciplinary Convergence

An interesting aspect of this work is that it crosses several disciplines. This section details convergence of computer science, computer engineering, and biomedical engineering for maternal healthcare technology broadly, with a summary given in Table 8.8.

8.7.1 Computer Science Contributions

Computer science techniques point us toward algorithmic foundations and software engineering that will be helpful with frame processing and LSCI calculations:

Machine Learning for Signal Processing: Advanced algorithms for extracting blood flow metrics from LSCI data, including deep learning for noise reduction [14, 15] and to enable AI-driven edge analysis [111].

Real-Time Scheduling: Real-Time Operating System (RTOS) scheduling techniques and priority inversion handling may ensure deterministic response times [26].

Data Structures and Algorithms: Efficient implementations of spatiotemporal filtering, ROI extraction, and statistical analysis [86].

Raspberry Pi AI feasibility: COVID-19 detection via chest X-ray has achieved 10.76 s per 100 images inference (SqueezeNet) and 99.83 % accuracy (ResNet50) [58].

8.7.2 Computer Engineering Contributions

The hardware architectural foundations of computer engineering provide the foundational hardware-software co-design and embedded optimizations including:

Hardware-Software Co-design: Raspberry Pi 5 optimization for medical imaging: camera interface configuration, DMA transfer optimization, cache management [114].

Edge Computing Architectures: On-device processing reduces latency and improves reliability. Raspberry Pi 5 achieves <40 % CPU usage and <14 % memory during multimodal neuroimaging [28].

Power-Aware Computing: Thermal management and power optimization for portable devices, maintaining safe temperatures while preserving performance [10].

8.7.3 Biomedical Engineering Contributions

The convergence in disciplines has created an environment with readily-available components, fostering hardware-software co-design for PPH detection in both low- and high-resource settings [11, 14, 96, 133]. Table 8.8 illustrates key convergence areas addressing maternal healthcare challenges.

8.8 Chapter Conclusions

New solutions will be essential to transitioning the current 80 FPS prototype from a metastable engineering success into a clinically reliable early-warning system. The rationale for these developments is rooted in the fundamental paradigm conflict identified by the entropy-bounded

Table 8.8: Interdisciplinary Convergence Points for PPH Detection

Domain	Computer Science	Computer Engineering	Biomedical Engineering
Sensing	Signal processing algorithms	Sensor interface design	Optical physics of LSCI
Processing	ML/AI models	Embedded optimization	Physiological signal interpretation
Display	GUI development	Display drivers	Clinical visualization standards
Alert	Notification algorithms	Wireless communication	Clinical decision thresholds
Power	Software power management	Hardware power circuits	Battery technology
Validation	Statistical analysis	Calibration methods	Clinical validation protocols

performance model (EBPM): while the Raspberry Pi 5 successfully achieves the high-speed 80 FPS throughput necessary to capture rapid, high-frequency blood flow transients, it currently fails clinical safety targets due to latency jitter (±8.3 ms) and unpredictable entropy collapse events observed at 85% CPU utilization. Future development will prioritize:

1. **Deterministic Scheduling:** Time-triggered scheduling to reduce processing entropy and achieve sub-2 ms jitter.
2. **Multi-Core Optimization:** Utilize all four ARM Cortex-A76 cores for parallel speckle processing targeting <10 ms frame time.
3. **Predictive Buffering:** Entropy-aware buffer management prevents latency spikes during garbage collection.
4. **Thermal-Aware Design:** Temperature monitoring informs performance scaling for extended portable operation.
5. **Algorithm Optimization:** Implement optimal exposure times (0.1 ms to 3 ms) to improve signal-to-noise ratio while maintaining real-time processing.

The novelty of these methods lies in their grounding in thermodynamic principles. By applying the Boltzmann uniqueness theorem, the research moves beyond traditional best-effort

computing toward a state of deterministic equilibrium. Implementing multi-core optimization to achieve an optimal task load of approximately 25% per core is required to meet the <10 ms processing target, while deterministic scheduling and entropy-aware predictive buffering are mandatory to reduce the Shannon entropy of the processing schedule $H(S)$. These optimizations are necessary to eliminate the catastrophic 280 ms garbage collection spikes that risk losing critical physiological data. Finally, thermal-aware design ensures that computational temperature does not shift the stationary solution toward latency-dominated regimes, providing the stable, pre-symptomatic detection required to combat global maternal mortality.

This work establishes that while contemporary embedded hardware like the Raspberry Pi 5 possesses the raw computational capacity for high-resolution medical sensing, clinical reliability is fundamentally limited by the Shannon entropy of the processing schedule. This study successfully validates a wearable Laser Speckle Contrast Imaging (LSCI) platform that achieves the 80 FPS throughput necessary for pre-symptomatic hemorrhage detection, yet concludes that a transition from throughput-focused to entropy-bounded architectures is mandatory for patient safety. Some of the novel revelations of this study include:

The Throughput-Latency Duality: Throughput (FPS) and latency (jitter) function as conjugate variables in a computational phase space. Optimizing for raw throughput inherently perturbs timing stability, explaining why current systems exhibit a ± 8.3 ms jitter that violates the ± 2 ms clinical safety threshold.

Entropy-Bounded Performance Model (EBPM): This research introduces a first-principles thermodynamic framework derived from the Boltzmann uniqueness theorem. It establishes that the maximum achievable frame rate is fundamentally bounded by the Shannon entropy of the processing delay distribution: $H(S) \leq \log_2(\tau_{frame}/\tau_{latency})$.

Computational "Temperature" and Phase Transitions: High CPU utilization (85%) is identified as a high-intensity computational temperature (T). As this temperature approaches critical thresholds ($\sim$90%), the system undergoes a non-linear phase transition we call "entropy collapse," moving from a predictable state to one characterized by catastrophic 280 ms latency spikes.

Thermal Instability as a Performance Driver: Operational temperatures of 58°C under load act as a macroscopic constraint that shifts the system's stationary solution

toward latency-dominated regimes. This makes active thermal management a prerequisite for preventing performance drift and patient injury during extended clinical monitoring.

Finally, there are several conclusions that we can determine from this work:

Hardware Sufficiency vs. Software Inadequacy: The Raspberry Pi 5 platform is confirmed to be hardware-sufficient for 80 FPS capture, but standard "best-effort" operating system scheduling is insufficient for clinical reliability.

Mandatory Deterministic Scheduling: To reduce the Shannon entropy of the processing schedule ($H(S)$) and bring jitter within the <2 ms physiological limit, deterministic, time-triggered scheduling is a mandatory requirement.

Equilibrium States (f^*_{CPU}): The system currently resides in a metastable state due to single-threaded saturation at 85% utilization. Reaching a stable equilibrium state requires multi-core optimization to distribute the load to approximately 25% per core.

LSCI monitoring has been validated as a superior early-warning system for postpartum hemorrhage (PPH) compared to traditional vitals. This work helps move that to practice.

Chapter 9

Conclusions and Future Work

9.1 Conclusions

Traditional heterogeneous computing assumptions of linear, independent core scaling fail under physical and electrical realities. Workload execution on modern asymmetric multiprocessors is ultimately bound by microarchitectural entanglement, interconnect congestion, and thermodynamic limits. Resolving these systemic bottlenecks requires abandoning heuristic software models in favor of hardware-aware, mathematically rigorous paradigms:

First-Principles Invertibility and Structural Equation Validation
Asymmetric multiprocessing architectures negate uniform scaling assumptions, demanding models based on core constitution. Structural Equation Modeling validates that workload-to-hardware profiles possess invertible mathematical structures, establishing equivalence ratios ($R_{BL} \approx 3:1$ for our empirical system) between heterogeneous cores. Matrix footprint and physical topology govern throughput limits, driving data starvation toward the Last-Level Cache as working sets scale and proving core expansion exhibits non-linear saturation.

Multi-Parameter Friction and Congestion Collapse
Runtime execution is dictated by dynamic stochastic variables rather than static hardware allocation. Formalized as the Critical Complexity Index (χ), on highly asymmetric topologies ($\chi > 2.0$), inter-cluster cache friction and memory bus contention overtake direct compute contributions, triggering a congestion collapse that requires efficiency cores to be force-idled to preserve primary execution.

Measure-Theoretic Architectural Entanglement
Software-defined power isolation relies on the false heuristic of cluster independence.

Applying the Boltzmann uniqueness theorem demonstrates a failure of convolution commutativity ($\mu_{\text{joint}} \neq \mu_{\text{big}} * \mu_{\text{little}}$), disproving additive energy models. The Ghost Power Coefficient (γ) quantifies this physical coupling, proving that fabric drag, monolithic voltage locking, and shared-cache inverse coupling override localized software policies and generate massive static leakage.

Thermodynamic Entropic Bounds in Clinical Engineering
Unmodeled microarchitectural contention degrades into thermodynamic entropy, transforming latency jitter into critical safety hazards for edge deployments. The Entropy-Bounded Performance Model (EBPM) resolves this conflict by treating processing delay as a fluid via the Boltzmann transport equation. Throughput is strictly bounded by the Shannon entropy of delays. Restoring deterministic execution requires new task allocation strategies ($f^*_{\text{CPU}} \approx 0.25$) to enforce thermodynamic equilibrium and eliminate latency spikes.

Translating these diagnostic and measure-theoretic foundations into next-generation systems engineering demands replacing speculative software abstractions with substrate-conforming runtime control. Future research must operationalize this paradigm across several vectors: engineering power-rail-aware operating system schedulers that mitigate intra-rail interaction friction dynamically, deploying structural equation generative models to isolate parasitic uncore transients in real time, and mapping post-arithmetic vector symbolic architectures directly onto analog non-volatile memory arrays to eliminate the Von Neumann memory wall.

9.2 Future Work

This section articulates a transition from Software Abstraction to Physical-Aware Computing. As shown throughout this dissertation, heterogeneous computing does not scale independently or additively as dynamic interconnect congestion, fabric drag, and voltage coupling create conditions wherein physical interactions override logical software control, making both performance and energy emergent properties of the entire silicon substrate. The central challenge for next-generation systems is therefore not how to schedule computations efficiently, but rather how to make software that is both adaptable and responsive

to the physical state of the machine itself. Therefore, we posit that future systems should break the abstraction boundary between software intent and silicon behavior. Four research directions are particularly critical to overcoming this boundary:

Physically-Aware Operating System Scheduling
Develop relational schedulers that continuously observe real-time electrical, thermal, and microarchitectural states of the silicon substrate to identify electrical coupling, congestion, and power consumption via hardware counters and structural models. Instead of dispersing threads according to static heuristics, kernels should co-locate, isolate, or quiesce workloads according to the instantaneous physical state of the substrate thereby turning hardware interaction from an uncontrolled effect into a scheduling signal.

Thermodynamic orchestration of heterogeneous SoCs
Model CPUs, GPUs, NPUs, and other accelerators as coupled computational ensembles rather than independent resources. A thermodynamic systems model could expose the operating boundaries between throughput, energy, heat, and latency, enabling schedulers to avoid nonlinear collapse before it occurs and provide predictable behavior under transient, multi-tenant workloads.

Generative inference of hidden hardware state
Combine structural equation models with generative learning to separate productive computation from parasitic power, microarchitectural noise, and dynamic voltage/frequency effects. The goal is a real-time model of both hardware behavior and software execution enabling automated discovery and prediction of emerging bottlenecks.

Post-arithmetic, in-memory architectures
Move beyond the assumption that computation requires repeated movement of data between memory and arithmetic units. Hyperdimensional and vector-symbolic representations mapped directly onto emerging non-volatile memory could replace costly data movement with hardware-native similarity operations, creating radically more energy-efficient and fault-tolerant computing substrates for edge, clinical, aerospace, and extreme-environment systems.

The long-term objective is a new systems discipline in which software, architecture, and physical silicon are co-designed as one dynamical system. The next breakthrough will not come from another isolated scheduling heuristic or a faster arithmetic unit. It will come from systems that understand their own physical state, predict where computation will fail, and adapt before failure occurs. The problem of tomorrow is to make computation aware of the substrate on which it exists and engineer platform-awareness into the hardware/software architecture from the start.

9.2.1 Future Research Roadmap

Future research can replace static thermal heuristics with energy-domain-aware schedulers that map physical-electrical coupling across the power delivery network. To mitigate fabric drag and monolithic voltage locking, schedulers should apply interaction matrix models to co-locate threads on shared voltage rails, minimizing parasitic static power across the entire SoC, including NPUs and GPUs. Formalizing the ghost power coefficient (γ) and the relational power matrix requires merging measure-theoretic physics with Structural Equation-Variational Autoencoder (SE-VAE) [134] architectures to explicitly isolate productive compute from uncore parasitic loads and microarchitectural transients. By applying the Boltzmann uniqueness theorem and structural equation modeling to expose non-linear workload interactions, this hardware-in-the-loop framework establishes a mathematical baseline for deterministic design space exploration in heterogeneous platforms. The co-design principles established in this dissertation provide a launchpad for the next generation of post-Von Neumann computer systems as outlined in Figure 9.1.

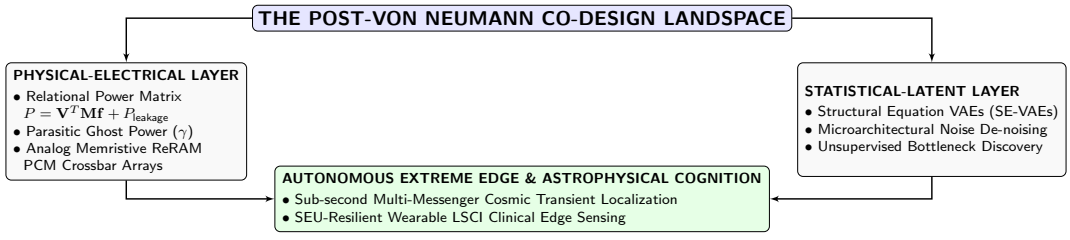


Figure 9.1: The Post-Von Neumann Co-Design Landscape.

The methodological success of deploying Structural Equation Modeling (SEM) to deconstruct complex microarchitectural and physical-electrical dependencies motivates a mathematical bridge into deep unsupervised learning. Traditional deep generative architectures, such as

Variational Autoencoders (VAEs) [30, 77], project high-dimensional tabular or time-series telemetry into a low-dimensional continuous latent space. However, standard VAEs suffer from a severe "interpretability deficit": the individual dimensions of the learned latent vector are highly entangled, making it impossible to assign distinct physical meanings (such as Core Performance or Cache Efficiency) to specific latent coordinates.

To resolve this limitation, our future roadmap pioneers the SE-VAE methodology. This hybrid architecture embeds formal covariance-based structural and measurement equation networks directly into the latent layers of the deep generative model:

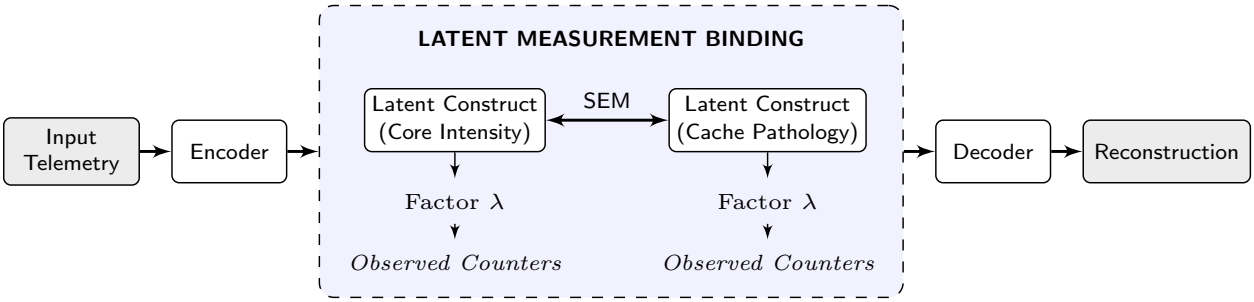


Figure 9.2: Structural Equation-Variational Autoencoder (SE-VAE) Architecture.

The proposed SE-VAE as shown in Figure 9.2 embeds known microarchitectural relationships directly into the model's latent representation. Rather than treating performance-counter streams such as instruction rates, branch mispredictions, and cache misses as independent observations, the encoder separates them into structured latent constructs that represent meaningful hardware behavior and nuisance variables that capture transient or unstructured variation. By optimizing the structural model and generative objective jointly, the SE-VAE can learn physically interpretable representations while filtering scheduling jitter, DVFS fluctuations, and other sources of measurement noise. This provides a path toward unsupervised, real-time identification of persistent hardware behavior, emerging bottlenecks, and anomalous states across heterogeneous server and mobile computing platforms.

References

[1] Laksono Adhianto et al. "HPCToolkit: Tools for Performance Analysis of Optimized Parallel Programs". In: *Concurrency and Computation: Practice and Experience* 22.6 (2010), pp. 685–701. DOI: 10.1002/cpe.1553.

[2] Yoav Almog et al. "Specialized dynamic optimizations for high-performance energy-efficient microarchitecture". In: *Proc. of Int'l Symposium on Code Generation and Optimization*. IEEE. 2004, pp. 137–148.

[3] Deepthi Amuru et al. "AI/ML algorithms and applications in VLSI design and technology". In: *Integration* 93 (2023).

[4] James L. Arbuckle. "Full Information Estimation in the Presence of Incomplete Data". In: *Advanced Structural Equation Modeling: Issues and Techniques*. Ed. by George A. Marcoulides and Randall E. Schumacker. Mahwah, NJ: Lawrence Erlbaum Associates, 1996, pp. 243–277.

[5] Christel Baier and Joost-Pieter Katoen. *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.

[6] Ioana Baldini, Stephen J Fink, and Erik Altman. "Predicting GPU performance from CPU runs using machine learning". In: *Proc. of 26th Int'l Symposium on Computer Architecture and High Performance Computing*. IEEE. 2014, pp. 254–261.

[7] Forest Baskett et al. "Open, closed, and mixed networks of queues with different classes of customers". In: *Journal of the ACM (JACM)* 22.2 (1975), pp. 248–260. DOI: 10.1145/321879.321887.

[8] Jonathan C. Beard and Roger D. Chamberlain. "Analysis of a Simple Approach to Modeling Performance for Streaming Data Applications". In: *Proc. of IEEE Int'l Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*. Aug. 2013, pp. 345–349. DOI: http://dx.doi.org/10.1109/MASCOTS.2013.49.

[9] Frank Bellosa. "The Benefits of Event-Driven Energy Accounting in Power-Sensitive Systems". In: *Proc. of 9th ACM SIGOPS European Workshop*. 2000, pp. 37–42. DOI: 10.1145/566726.566736.

[10] Ganapati Bhat et al. “Algorithmic optimization of thermal and power management for heterogeneous mobile platforms”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26.3 (2017), pp. 544–557. DOI: 10.1109/TVLSI.2017.2770163.

[11] David C Bock et al. “Batteries used to power implantable biomedical devices”. In: *Electrochimica Acta* 84 (2012), pp. 155–164. DOI: 10.1016/j.electacta.2012.03.057.

[12] Kenneth A. Bollen. *Structural Equations with Latent Variables*. New York: John Wiley & Sons, 1989. ISBN: 9780471011712. DOI: 10.1002/9781118619179.

[13] L. Boltzmann. “Weitere Studien über das Wärmegleichgewicht unter Gasmolekülen”. In: *Sitzungsberichte der Kaiserlichen Akademie der Wissenschaften. Mathematisch-Naturwissenschaftliche Classe* 66 (1872), pp. 275–370.

[14] Francesca Bonetta-Misteli et al. “Development and evaluation of a wearable peripheral vascular compensation sensor in a swine model of hemorrhage”. In: *Biomedical Optics Express* 14.10 (2023), pp. 5338–5357. DOI: 10.1364/BOE.494720.

[15] Francescsa Bonetta-Misteli et al. “Maternal aRMOR: early detection of postpartum hemorrhage with a wearable optical device”. In: *Optical Diagnostics and Sensing XXIII: Toward Point-of-Care Diagnostics*. Vol. 12387. SPIE. 2023, pp. 13–16. DOI: 10.1117/12.2651306.

[16] David Briers et al. “Laser speckle contrast imaging: theoretical and practical limitations”. In: *Journal of Biomedical Optics* 18.6 (2013), p. 066018. DOI: 10.1117/1.JBO.18.6.066018.

[17] David Brooks, Vivek Tiwari, and Margaret Martonosi. “Wattch: A Framework for Architectural-Level Power Analysis and Optimizations”. In: *27th International Symposium on Computer Architecture (ISCA)*. ACM, 2000, pp. 83–94. DOI: 10.1145/339647.339657.

[18] Anastasiia Butko et al. “Full-system simulation of big.LITTLE multicore architecture for performance and energy exploration”. In: *Proc. of IEEE 10th Int’l Symposium on Embedded Multicore/Many-core Systems-on-Chip*. 2016, pp. 201–208.

[19] Dhruba Chandra et al. “Predicting inter-thread cache contention on a chip multiprocessor architecture”. In: *11th International Symposium on High-Performance Computer Architecture*. IEEE. 2005, pp. 340–351. DOI: 10.1109/HPCA.2005.27.

[20] Yuan Chou, Brian Fahs, and Santosh Abraham. "Microarchitecture optimizations for exploiting memory-level parallelism". In: *ACM SIGARCH Computer Architecture News* 32.2 (2004), p. 76.

[21] Taco Cohen and Max Welling. "Group equivariant convolutional networks". In: *International Conference on Machine Learning*. PMLR. 2016, pp. 2990–2999.

[22] Gilberto Contreras and Margaret Martonosi. "Power Prediction for Intel XScale Processors Using Performance Monitoring Unit Events". In: *Proc. of International Symposium on Low Power Electronics and Design (ISLPED)*. 2005, pp. 221–226. DOI: 10.1145/1077603.1077657.

[23] Shaohang Cui, Douglas Buchanan, and Ken Ferens. "A knapsack scheduling algorithm for soft real-time multiprocessor system". In: *Proc. of Int'l Conference on Embedded Systems, Cyber-physical Systems, and Applications*. 2013.

[24] Willow E Cunningham and Vincent M Weaver. "Performance Measurement on Heterogeneous Processors with PAPI". In: *Proc. of Workshops of Int'l Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE. 2024, pp. 1551–1561.

[25] Leonardo Dagum and Ramesh Menon. "OpenMP: An Industry Standard API for Shared-Memory Programming". In: *IEEE Computational Science and Engineering* 5.1 (1998), pp. 46–55. DOI: 10.1109/99.660313.

[26] Robert I Davis and Alan Burns. "A survey of hard real-time scheduling for multiprocessor systems". In: *ACM Computing Surveys (CSUR)* 43.4 (2011), 35:1–35:44. DOI: 10.1145/1978802.1978814.

[27] Dereje Bayissa Demissie, Doreen Kainyu Kaura, and Stefan Gebhardt. "Global burden, disparities, and determinants of postpartum haemorrhage among women who gave birth: an umbrella review of systematic reviews and meta-analyses". In: *Frontiers in Reproductive Health* 7 (2025), p. 1721550. DOI: 10.3389/frph.2025.1721550.

[28] Navjodh Singh Dhillon et al. "A Raspberry Pi-based traumatic brain injury detection system for single-channel electroencephalogram". In: *Sensors* 21.8 (2021), p. 2779. DOI: 10.3390/s21082779.

[29] Yi Ding et al. "Generalizable and interpretable learning for configuration extrapolation". In: *Proc. of 29th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering*. 2021, pp. 728–740.

[30] Carl Doersch. "Tutorial on variational autoencoders". In: *arXiv:1606.05908* (2016).

[31] Christophe Dubach, Timothy M Jones, and Michael FP O'Boyle. "Exploring and predicting the effects of microarchitectural parameters and compiler optimizations on performance and energy". In: *ACM Transactions on Embedded Computing Systems* 11.1 (2012), pp. 1–24.

[32] Philippe Duchon et al. "Boltzmann samplers for the random generation of combinatorial structures". In: *Combinatorics, Probability and Computing* 13.4-5 (2004), pp. 577–625. DOI: 10.1017/S0963548304006315.

[33] Isabelle Dumé. "Wearable optical device could detect postpartum haemorrhage". In: *Physics World* (Nov. 2023).

[34] Muhammad Muzzammil Edhi et al. "Post partum hemorrhage: causes and management". In: *BMC Research Notes* 6.1 (2013), p. 236. DOI: 10.1186/1756-0500-6-236.

[35] Lieven Eeckhout, Hans Vandierendonck, and Koen De Bosschere. "Designing Representative Benchmark Subsets Using Principal Component Analysis and Cluster Analysis". In: *Proc. of 12th International Conference on Parallel Architectures and Compilation Techniques (PACT)*. 2003, pp. 201–210. DOI: 10.1109/PACT.2003.1238018.

[36] Stijn Eyerman et al. "A Mechanistic Performance Model for Superscalar Out-of-Order Processors". In: *ACM Transactions on Computer Systems* 27.2 (2009), 3:1–3:37. DOI: 10.1145/1534909.1534910.

[37] Clayton J. Faber and Roger D. Chamberlain. "Application of Network Calculus Models to Heterogeneous Streaming Applications". In: *International Journal of Networking and Computing* 15.1 (Jan. 2025), pp. 51–63. DOI: 10.15803/ijnc.15.1_51.

[38] Clayton J. Faber et al. "Platform Agnostic Streaming Data Application Performance Models". In: *Proc. of IEEE/ACM Workshop on Redefining Scalability for Diversely Heterogeneous Architectures*. Nov. 2021. DOI: 10.1109/RSDHA54838.2021.00008.

[39] Youssef Faqir-Rhazoui and Carlos García. "SYCL in the edge: performance and energy evaluation for heterogeneous acceleration". In: *The Journal of Supercomputing* 80.10 (2024), pp. 14203–14223.

[40] Carl Finkbeiner. "Estimation for the Multiple Factor Model When Data Are Missing". In: *Psychometrika* 44.4 (1979), pp. 409–420. DOI: 10.1007/BF02296204.

[41] Steven Fortune and James Wyllie. "Parallelism in random access machines". In: *Proc. of 10th ACM Symposium on Theory of Computing*. 1978, pp. 114–118.

[42] Wayne A. Fuller. *Measurement Error Models*. New York: John Wiley & Sons, 1987. ISBN: 9780471861874. DOI: 10.1002/9780470316665.

[43] Kazushige Goto and Robert A. van de Geijn. "Anatomy of High-Performance Matrix Multiplication". In: *ACM Transactions on Mathematical Software* 34.3 (2008), 12:1–12:25. DOI: 10.1145/1356052.1356053.

[44] J. B. Grace. *Structural Equation Modeling and Natural Systems*. Cambridge: Cambridge University Press, 2006.

[45] Peter Greenhalgh. *big.LITTLE Processing with ARM Cortex-A15 & Cortex-A7*. White Paper. ARM Limited, 2011.

[46] JA Grzesik. "A Novel Derivation of the Gregory-Leibniz Series for Pi". In: *The College Mathematics Journal* 55.4 (2024), pp. 346–348. DOI: 10.1080/07468342.2023.2286167.

[47] Joseph F Hair et al. "When to use and how to report the results of PLS-SEM". In: *European Business Review* 31.1 (2019), pp. 2–24.

[48] Steven D Harris, Roger D Chamberlain, and Christopher D Gill. "Heterogeneous Non-Linear Performance Modeling of Parameter Interactions". In: *Proc. of International Parallel and Distributed Processing Symposium Workshops*. IEEE, June 2026, pp. 57–66. DOI: 10.1109/IPDPSW71298.2026.00014.

[49] Steven D Harris, Roger D Chamberlain, and Christopher D Gill. "Performance Modeling of Non-Uniform Heterogeneous Platforms". In: *Proc. of International Parallel and Distributed Processing Symposium Workshops*. IEEE, June 2025, pp. 597–607. DOI: 10.1109/IPDPSW66978.2025.00094.

[50] Steven D Harris, Roger D Chamberlain, and Christopher D Gill. "Performance Models for Non-Uniform Heterogeneous Platforms and their Validation Using Structural Equation Modeling". In: *International Journal of Networking and Computing* 16.1 (Jan. 2026), pp. 20–51. DOI: 10.15803/ijnc.16.1_20.

[51] Steven D Harris, Roger D Chamberlain, and Christopher D Gill. "Understanding Asymmetric Multicores Using Structural Equation Modeling". In: *International Journal of Networking and Computing* (2027). Accepted for publication.

[52] Steven D Harris, Christopher D Gill, and Roger D Chamberlain. "Measure-Theoretic Diagnostics of Architectural Entanglement in Asymmetric Multiprocessing Systems: A Boltzmann Uniqueness Approach". In: *Modelling* 7.4 (Aug. 2026), 124:1–124:31. DOI: 10.3390/modelling7040124.

[53] Steven D. Harris. "Structural Equation Modeling for Heterogeneous Platforms". In: *Proc. of IEEE/ACM International Conference for High Performance Computing, Networking, Storage, and Analysis*. Nov. 2025.

[54] Steven D. Harris, Christopher D. Gill, and Roger D. Chamberlain. "Poster Abstract: Understanding Performance-Impacting Factors". In: *Proc. of ACM/IEEE 17th International Conference on Cyber-Physical Systems (ICCPS)*. May 2026, pp. 273–274.

[55] Steven D. Harris et al. "An Entropy-Bounded Thermodynamic Approach for Real-Time Maternal Hemorrhage Monitoring". In: *Proc. of IEEE High Performance Extreme Computing Conference*. Sept. 2026.

[56] Thomas A Henzinger et al. "Symbolic model checking for real-time systems". In: *Information and Computation* 111.2 (1994), pp. 193–244. DOI: 10.1006/inco.1994.1045.

[57] Torsten Hoefler et al. "Performance modeling for systematic performance tuning". In: *Proc. of Int'l Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2011, 6:1–6:12.

[58] Khalid M Hosny et al. "COVID-19 diagnosis from CT scans and chest X-ray images using low-cost Raspberry Pi". In: *PLOS One* 16.5 (2021), e0250688. DOI: 10.1371/journal.pone.0250688.

[59] Kenneth Hoste and Lieven Eeckhout. "Comparing Benchmarks Using Microarchitecture–Independent Characteristics". In: *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 2006, pp. 127–137. DOI: 10.1109/ISPASS.2006.1620796.

[60] R. H. Hoyle. *Structural Equation Modeling for Social and Personality Psychology*. London: Sage, 2011.

[61] R. H. Hoyle. *Structural Equation Modeling: Concepts, Issues, and Applications*. Thousand Oak: Sage, 1995.

[62] Paul Hübner et al. “Apple vs. Oranges: Evaluating the Apple Silicon M-Series SoCs for HPC Performance and Efficiency”. In: *Proc. of Int’l Parallel and Distributed Processing Symposium Workshops*. IEEE. 2025, pp. 45–54.

[63] Olumuyiwa Ibidunmoye, Francisco Hernández-Rodriguez, and Erik Elmroth. “Performance anomaly detection and bottleneck identification”. In: *ACM Computing Surveys* 48.1 (2015), pp. 1–35.

[64] Engin Ïpek et al. “Efficiently exploring architectural design spaces via predictive modeling”. In: *ACM SIGOPS Operating Systems Review* 40.5 (2006), pp. 195–206.

[65] Joseph Issa and Silvia Figueira. “A performance estimation model for GPU-based systems”. In: *Proc. of 2nd Int’l Conference on Advances in Computational Tools for Engineering Applications*. IEEE. 2012, pp. 279–283.

[66] Anoop Iyer and Diana Marculescu. “Power Aware Microarchitecture Resource Scaling”. In: *Design, Automation and Test in Europe Conference*. IEEE, 2001, pp. 190–196. DOI: 10.1109/DATE.2001.915023.

[67] Ravi Iyer et al. “QoS policies and architecture for cache/memory in CMP platforms”. In: *ACM SIGMETRICS Performance Evaluation Review* 35.1 (2007), pp. 25–36. DOI: 10.1145/1269899.1254886.

[68] Brian Jeff. *big.LITTLE Technology Moves Towards Fully Heterogeneous Global Task Scheduling*. Technical Paper. ARM Limited, 2013.

[69] Karl G. Jöreskog. “A General Method for Analysis of Covariance Structures”. In: *Biometrika* 57.2 (Aug. 1970), pp. 239–251. DOI: 10.1093/biomet/57.2.239.

[70] Karl G. Jöreskog. “Analysis of Covariance Structures”. In: *Multivariate Analysis–III*. Ed. by P. R. Krishnaiah. New York: Academic Press, 1973, pp. 263–285. DOI: 10.1016/B978-0-12-426653-7.50024-7.

[71] Karl G. Jöreskog. “Structural Analysis of Covariance and Correlation Matrices”. In: *Psychometrika* 43.4 (1978), pp. 443–477. DOI: 10.1007/BF02293808.

[72] Camilla Josephson. “Proving an Arrow of Time under Universal Time: A DSFL Framework for Black-Hole Evaporation”. In: *Preprints* (2025).

[73] Miao Ju, Hun Jung, and Hao Che. “A performance analysis methodology for multicore, multithreaded processors”. In: *IEEE Transactions on Computers* 63.2 (2012), pp. 276–289.

[74] Sergei Igorevich Kabanikhin et al. “Definitions and examples of inverse and ill-posed problems”. In: *Journal of Inverse and Ill-Posed Problems* 16.4 (2008), pp. 317–357. DOI: 10.1515/JIIP.2008.019.

[75] Vijay Kandiah et al. “AccelWattch: A power modeling framework for modern GPUs”. In: *Proc. of 54th IEEE/ACM Int’l Symposium on Microarchitecture*. 2021, pp. 738–753.

[76] Shmuel Kaniel and Marvin Shinbrot. “The Boltzmann equation: I. Uniqueness and local existence”. In: *Communications in Mathematical Physics* 58.1 (1978), pp. 65–84. DOI: 10.1007/BF01624788.

[77] Diederik P Kingma and Max Welling. “Auto-encoding variational Bayes”. In: *arXiv preprint arXiv:1312.6114* (2013).

[78] Rex B Kline. *Principles and Practice of Structural Equation Modeling*. 5th edition. New York: Guilford Press, 2023.

[79] Hugh Leather and Chris Cummins. “Machine learning in compilers: Past, present and future”. In: *Proc. of Forum for Specification and Design Languages*. IEEE. 2020.

[80] Benjamin C. Lee and David M. Brooks. “Accurate and efficient regression modeling for microarchitectural performance and power prediction”. In: *Proc. of 12th Int’l Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 2006, pp. 185–194. DOI: 10.1145/1168857.1168881.

[81] Sheng Li et al. “McPAT: An Integrated Power, Area, and Timing Modeling Framework for Multicore and Many-Core Architectures”. In: *42nd IEEE/ACM International Symposium on Microarchitecture (MICRO)*. ACM, 2009, pp. 469–480. DOI: 10.1145/1669112.1669172.

[82] Xiaoyao Liang and David Brooks. “Microarchitecture parameter selection to optimize system performance under process variation”. In: *Proc. of IEEE/ACM Int’l Conf. on Computer-Aided Design*. 2006, pp. 429–436.

[83] Sebastian Litzinger, Jörg Keller, and Christoph Kessler. “Scheduling Moldable Parallel Streaming Tasks on Heterogeneous Platforms with Frequency Scaling”. In: *Proc. of 27th European Signal Processing Conference*. 2019, pp. 1–5. DOI: 10.23919/EUSIPCO.2019.8903180.

[84] David Lo et al. "Heracles: Improving resource efficiency at scale". In: *42nd International Symposium on Computer Architecture*. ACM. 2015, pp. 450–462. DOI: 10.1145/2749469.2749475.

[85] Frederic M. Lord and Melvin R. Novick. *Statistical Theories of Mental Test Scores*. Reading, MA: Addison-Wesley, 1968.

[86] Zhengdong Lu, Deepak Agarwal, and Inderjit S Dhillon. "A spatio-temporal approach to collaborative filtering". In: *Proc. of 3rd ACM Conference on Recommender Systems*. 2009, pp. 13–20. DOI: 10.1145/1639714.1639719.

[87] Yuchun Ma et al. "Micro-architecture pipelining optimization with throughput-aware floorplanning". In: *Proc. of Asia and South Pacific Design Automation Conference*. IEEE. 2007, pp. 920–925.

[88] Jason Mars et al. "Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations". In: *44th IEEE/ACM International Symposium on Microarchitecture*. ACM, 2011, pp. 248–259. DOI: 10.1145/2155620.2155650.

[89] Ralph O Mueller and Gregory R Hancock. "Structural equation modeling". In: *The Reviewer's Guide to Quantitative Methods in the Social Sciences*. Routledge, 2018, pp. 445–456.

[90] W. James Murdoch, Chandan Singh, and Bin Yu. "Hierarchical Interpretations for Neural Network Predictions". In: *International Conference on Learning Representations (ICLR)*. 2019.

[91] Marco Pagani et al. "A Linux-based support for developing real-time applications on heterogeneous platforms with dynamic FPGA reconfiguration". In: *Proc. of 30th IEEE Int'l System-on-Chip Conference*. 2017, pp. 96–101. DOI: 10.1109/SOCC.2017.8226015.

[92] A Patel et al. "Drape estimation vs. visual assessment for estimating postpartum hemorrhage". In: *International Journal of Gynecology & Obstetrics* 93.3 (2006), pp. 220–224. DOI: 10.1016/j.ijgo.2006.02.014.

[93] Aashish Phansalkar, Ajay Joshi, and Lizy K. John. "Analysis of Redundancy and Application Balance in the SPEC CPU2006 Benchmark Suite". In: *Proc. of 34th International Symposium on Computer Architecture (ISCA)*. 2007, pp. 412–423. DOI: 10.1145/1250662.1250713.

[94] Michael D Powell et al. “Intel Xeon 6 Product Family”. In: *IEEE Micro* 45.3 (2025), pp. 31–40.

[95] Feitong Qiao, Yiming Fang, and Asaf Cidon. “Energy-Aware process scheduling in Linux”. In: *ACM SIGENERGY Energy Informatics Review* 4.5 (2024), pp. 91–97. DOI: 10.1145/3727200.3727214.

[96] Rangaraj M Rangayyan and Sridhar Krishnan. *Biomedical Signal Analysis*. John Wiley & Sons, 2024.

[97] Manideep Raya VSRS Reddy, Sanjay Ramapogu Raju, Biswajit Bhowmik, et al. “Performance Analysis and Predictive Modeling of MPI Collective Algorithms in Multi-Core Clusters: A Comparative Study”. In: *Proc. of 17th Int’l Conference on Communication Systems and Networks*. IEEE. 2025, pp. 448–455.

[98] Juan A Rico-Gallego et al. “A survey of communication performance models for high-performance computing”. In: *ACM Computing Surveys* 51.6 (2019), pp. 1–36.

[99] Efraim Rotem et al. “Power-Management Architecture of the Intel Microarchitecture Code-Named Sandy Bridge”. In: *IEEE Micro* 32.2 (2012), pp. 20–27. DOI: 10.1109/MM.2012.12.

[100] Dennis Ryu et al. “Comprehensive pregnancy monitoring with a network of wireless, soft, and flexible sensors in high- and low-resource health settings”. In: *Proceedings of the National Academy of Sciences (PNAS)* 118.20 (2021), e2100466118. DOI: 10.1073/pnas.2100466118.

[101] Fedor Sandomirskiy and Omer Tamuz. “On the origin of the Boltzmann distribution”. In: *Mathematische Annalen* 392.4 (2025), pp. 5617–5638. DOI: 10.1007/s00208-025-03263-x.

[102] Patrick Schober, Christa Boer, and Lothar A Schwarte. “Correlation coefficients: Appropriate use and interpretation”. In: *Anesthesia & Analgesia* 126.5 (2018), pp. 1763–1768. DOI: 10.1213/ANE.0000000000002864.

[103] James B Schreiber. “Core reporting practices in structural equation modeling”. In: *Research in Social and Administrative Pharmacy* 4.2 (2008), pp. 83–97.

[104] David V Schuehler et al. “Microarchitecture optimization for embedded systems”. In: *Proc. of 8th High Performance Embedded Computing Workshop*. 2004.

[105] Robert Sedgewick and Philippe Flajolet. *Analytic Combinatorics*. Vol. 150. Cambridge, UK: Cambridge University Press, 2009.

[106] J. Senarathna et al. “Laser speckle contrast imaging: theory, instrumentation and medical applications”. In: *Nature Reviews Physics* 5 (2023), pp. 533–551. DOI: 10.1038/s42254-023-00603-7.

[107] Janaka Senarathna et al. “Laser speckle contrast imaging: theory, instrumentation and applications”. In: *IEEE Reviews in Biomedical Engineering* 6 (2013), pp. 99–110. DOI: 10.1109/RBME.2013.2243140.

[108] Daniel Shelepov et al. “HASS: Heterogeneity-Aware Signature-Based Scheduling”. In: *Proc. of 4th ACM European Conference on Computer Systems (EuroSys)*. 2009, pp. 115–128. DOI: 10.1145/1519065.1519078.

[109] Timothy Sherwood et al. “Automatically Categorizing Program Behavior”. In: *Proc. of 10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*. 2002, pp. 45–57. DOI: 10.1145/605397.605403.

[110] N. Shirazi, A. Walters, and P. Athanas. “Quantitative analysis of floating point arithmetic on FPGA based custom computing machines”. In: *Proc. of IEEE Symposium on FPGAs for Custom Computing Machines*. 1995, pp. 155–162. DOI: 10.1109/FPGA.1995.477421.

[111] Raghubir Singh and Sukhpal Singh Gill. “Edge AI: a survey”. In: *Internet of Things and Cyber-Physical Systems* 3 (2023), pp. 71–92. DOI: 10.1016/j.iotcps.2023.02.004.

[112] Charles Spearman. “The Proof and Measurement of Association between Two Things”. In: *The American Journal of Psychology* 15.1 (1904), pp. 72–101. DOI: 10.2307/1412159.

[113] Petre Stoica and Yngve Selen. “Model-order selection: a review of information criterion rules”. In: *IEEE Signal Processing Magazine* 21.4 (2004), pp. 36–47.

[114] Srinivas Talasila et al. “Raspberry Pi Powered Deep Learning-Based Chest X-ray Analysis for Medical Diagnosis”. In: *Proc. of International Conference on Blockchain and Distributed Systems Security*. IEEE. 2024. DOI: 10.1109/ICBDS61829.2024.10837221.

[115] Nathan R. Tallent and John M. Mellor-Crummey. “Identifying Performance Bottlenecks in Work-Stealing Computations”. In: *IEEE Computer* 42.12 (2009), pp. 44–50. DOI: 10.1109/MC.2009.396.

[116] Derartu D Tekela et al. “Digital postpartum hemorrhage management device (DPH-MD)”. In: *BMC Pregnancy and Childbirth* 19.1 (2019), p. 438. DOI: 10.1186/s12884-019-2601-3.

[117] Massimo Tessarotto, Claudio Cremaschini, and Marco Tessarotto. “On the conditions of validity of the Boltzmann equation and Boltzmann H-theorem”. In: *The European Physical Journal Plus* 128.3 (2013), p. 32. DOI: 10.1140/epjp/i2013-13032-2.

[118] Alexander Thomasian. “Hierarchical Analyses Applied to Computer System Performance: Review and Call for Further Studies”. In: *arXiv preprint arXiv:2401.09292* (2024).

[119] Stephen M. Trimberger and Jason J. Moore. “FPGA Security: Motivations, Features, and Applications”. In: *Proceedings of the IEEE* 102.8 (2014), pp. 1248–1265. DOI: 10.1109/JPROC.2014.2331672.

[120] Christian Trott et al. “The Kokkos EcoSystem: Comprehensive performance portability for high performance computing”. In: *Computing in Science & Engineering* 23.5 (2021), pp. 10–18.

[121] Sathish S Vadhiyar, Graham E Fagg, and Jack J Dongarra. “Performance modeling for self adapting collective communications for MPI”. In: *Proc. of Los Alamos Computer Science Institute Symposium*. Oct. 2001.

[122] Kenzo Van Craeynest et al. “Scheduling Heterogeneous Multi-Cores Through Fast-Forwarding Analysis”. In: *IEEE Micro* 32.3 (2012), pp. 126–136.

[123] Siqi Wang et al. “High-throughput CNN inference on embedded ARM big.LITTLE multicore processors”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 39.10 (2019), pp. 2254–2267.

[124] Vincent M. Weaver and Sally A. McKee. “Can Hardware Performance Counters Be Trusted?” In: *2008 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, 2008, pp. 141–150. DOI: 10.1109/IISWC.2008.4636099.

[125] A. Weeks. “The prevention and treatment of postpartum haemorrhage: what do we know, and where do we go to next?” In: *BJOG: An International Journal of Obstetrics & Gynaecology* 122.2 (2015), pp. 202–210. DOI: 10.1111/1471-0528.13098.

[126] Samuel Williams, Andrew Waterman, and David Patterson. “Roofline: an insightful visual performance model for multicore architectures”. In: *Communications of the ACM* 52.4 (2009), pp. 65–76. DOI: 10.1145/1498765.1498785.

[127] Piotr Wojciechowski et al. “Priority-based bin packing with subset constraints”. In: *Discrete Applied Mathematics* 342 (2024), pp. 64–75.

[128] Gene Wu et al. “GPGPU performance and power estimation using machine learning”. In: *Proc. of 21st Int’l Symposium on High Performance Computer Architecture*. IEEE. 2015, pp. 564–576.

[129] Xingfu Wu and Valerie Taylor. “Performance modeling of hybrid MPI/OpenMP scientific applications on large-scale multicore supercomputers”. In: *Journal of Computer and System Sciences* 79.8 (2013), pp. 1256–1268.

[130] Haojun Xia et al. “Flash-LLM: Enabling Cost-Effective and Highly-Efficient Large Generative Model Inference with Unstructured Sparsity”. In: *Proc. VLDB Endow.* 17.2 (Oct. 2023), pp. 211–224. ISSN: 2150-8097. DOI: 10.14778/3626292.3626303.

[131] Ahmad Yasin. “A Top-Down Method for Performance Analysis and Counters Architecture”. In: *IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS)*. 2014, pp. 35–44. DOI: 10.1109/ISPASS.2014.6844459.

[132] Anna Yue and Sanyam Mehta. “An Application-Oriented Approach to Designing Hybrid CPU Architectures”. In: *Proc. of IEEE Int’l Symposium on Performance Analysis of Systems and Software*. 2023, pp. 92–102. DOI: 10.1109/ISPASS57527.2023.00018.

[133] Lingke Zhang et al. “Dual-wavelength laser speckle contrast imaging (dwLSCI) improves chronic measurement of superficial blood flow in hands”. In: *Sensors* 17.12 (2017), p. 2811. DOI: 10.3390/s17122811.

[134] Ruiyu Zhang et al. “Structural Equation-VAE: Disentangled latent representations for tabular data”. In: *arXiv preprint arXiv:2508.06347* (2025).

[135] Kan Zhu et al. “NanoFlow: Towards optimal large language model serving throughput”. In: *Proc. of 19th USENIX Symposium on Operating Systems Design and Implementation*. 2025, pp. 749–765.

[136] Yuhao Zhu and Vijay Janapa Reddi. “High-performance and energy-efficient mobile web browsing on big/little systems”. In: *Proc. of IEEE 19th Int’l Symp. on High Performance Computer Architecture*. 2013, pp. 13–24.

[137] Sergey Zhuravlev, Sergey Blagodurov, and Alexandra Fedorova. “Addressing shared resource contention in multicore processors via scheduling”. In: *ACM Sigplan Notices* 45.3 (2010), pp. 129–142. DOI: 10.1145/1735971.1736036.

Appendix A

Boltzman Theorem and Distribution

A.1 Boltzmann Uniqueness Theorem

The Boltzmann uniqueness theorem characterizes the family of Boltzmann distributions as the unique family of maps that satisfy independence for uncoupled systems. In statistical mechanics, the Boltzmann distribution describes the distribution of states in systems with a given temperature, where the probability of a system having energy E is proportional to the number of states at that energy multiplied by $e^{-\beta E}$.

- Any map Φ that is measure-class-preserving and commutes with convolution must belong to the Boltzmann family Φ_β. This is defined by the core identity:

$$\Phi_\beta[\mu_1 * \mu_2] = \Phi_\beta[\mu_1] * \Phi_\beta[\mu_2]$$

  where μ_1 and μ_2 are probability measures representing subsystem states, and $*$ denotes the convolution operator.

- **General Importance:** The theorem offers an alternative to the traditional maximum entropy explanation for the Boltzmann distribution. It proves that if a system is uncoupled (no interaction), the joint distribution of its states is simply the product (convolution) of the distributions of its individual subsystems.

- **Importance for AMP Analysis:** In platforms like the Orange Pi 5 or Cubie A7A, developers often assume that core islands (Big vs. Little clusters) operate independently. The Boltzmann uniqueness theorem acts as a diagnostic tool by testing if the joint energy distribution of these clusters matches the convolution of their isolated states, giving researchers the power to formally disprove cluster independence.

A.2 Boltzmann Distribution

The Boltzmann distribution gives the probability that a system in thermal equilibrium at temperature T will be found in a state with energy E. In statistical mechanics, the Boltzmann distribution for a system in thermal equilibrium is given by:

$$P(E) = \frac{1}{Z} e^{-\beta E}, \tag{A.1}$$

where $\beta = \frac{1}{k_B T}$.

Regarding this measure, the variables are as follows:

- $P(E)$ - Probability of the system having energy E
- Z - Partition function (normalization constant ensuring total probability = 1)
- β - *Thermodynamic beta* or simply *inverse temperature parameter*
- k_β - Boltzmann's constant

In terms of energy, the lower energy states are exponentially more probable. At high T (small β), the exponential approaches 1, making all states roughly equally likely. At low T (large β), only the lowest energy states dominate. These systems naturally prefer lower energy states, but thermal energy (T) causes occasional excitations to higher states with balance between competing effects captured by this distribution. To clarify this distribution, we will outline several implications as follows:

- **Convolution and Sum of Variables**: The distribution of the total energy E_{total} is the convolution of the individual density of states or probability measures of the subsystems. For independent random variables, the measure of their sum is the convolution of their individual measures:

$$(P_A * P_B)(E) = \int P_A(\epsilon) P_B(E - \epsilon) d\epsilon$$

- **Subsystem Independence**: If a system is composed of two subsystems, A and B, that are truly uncoupled, their total energy is strictly additive: $E_{total} = E_A + E_B$. In this state, the subsystems are statistically independent. The joint probability distribution $P(E_A, E_B)$ must factorize:

$$P(E_A, E_B) = P(E_A) \cdot P(E_B)$$

- **The Endomorphism Property**: The statement identifies the Boltzmann distribution as the unique convolution endomorphism (a mapping of the space of measures into itself) because the exponential form is the only functional form that preserves its structure under convolution while maintaining the additivity of the underlying variables.

- **Structure Preservation**: If $P_A \propto e^{-\beta E_A}$ and $P_B \propto e^{-\beta E_B}$, then $P_{A+B} \propto e^{-\beta(E_A+E_B)}$ is the joint distribution.

- **Uniqueness**: In the limit of large N, the Central Limit Theorem and the maximum entropy principle dictate that the exponential (Boltzmann) distribution is the unique stable distribution for energy exchange where only the mean energy is constrained.

- **Physical Implication**: If subsystems are "truly uncoupled," there is no interaction energy V_{int} such that $E_{total} = E_A + E_B + V_{int}$. If $V_{int} \neq 0$, the measures are coupled, the joint distribution no longer factorizes, and the convolution of isolated measures fails to describe the total system state accurately.

The correlation between the energy subsystem convolutions are characterized in Table A.1.

A.3 Boltzman AMP Applicability

In performance modeling and power management for heterogeneous computing (e.g., ARM big.LITTLE architecture), the use of Boltzmann-derived logic without explicit temperature or physical constants stems from isomorphism. The mathematical framework of statistical mechanics is used as an abstraction for resource allocation and stochastic task scheduling. A few relationships between the isomorphism include:

Table A.1: Measure-Theoretic Diagnostic of Architectural Entanglement in AMPs.

Scenario	Corr. (r)	p-Value	Measure-Theoretic State	Architectural Implication
Ideal Uncoupled	$r > 0.9$	$p < 0.05$	**Convolutional Commutativity:** [21] $\Phi_\beta[\mu_1 * \mu_2] = \Phi_\beta[\mu_1] * \Phi_\beta[\mu_2]$.	Validates cluster orthogonality; independent resource consumption.
Coupled (Entangled)	$r < 0$	$p < 0.05$	**Inverse Coupling:** [72] Support-preserving endomorphism failure in the semi-group.	Direct contention for shared power/thermal rails; zero-sum resource states.
Non-Linear Collapse	$r \approx 0$	$p > 0.05$	**Independence Identity Violation:** [74] Total loss of predictive power for the convolution model.	LLC cache-line contention and starvation bottlenecks; additive identity failure.

1. **The Energy-Delay Product (EDP)** In computing, "energy" is not just thermal; it is the cost of execution. The energy E in the Boltzmann factor is replaced by a cost function, often the energy-delay product.

   - **Big Cores**: High power, low latency (Low cost for heavy tasks, high cost for idle).
   - **Little Cores**: Low power, high latency (Low cost for background tasks, high cost for heavy tasks due to time-outs).

2. β **as a Sensitivity Parameter** Since there is no physical T, β is redefined as a logit scaling factor or inverse sensitivity.

   - It acts as a tuning parameter for the "greediness" of the scheduler.
   - A high β (equivalent to low temperature) forces the system into the "ground state" wherein the absolute lowest-power core is chosen regardless of performance.
   - A low β (equivalent to high temperature) allows for *thermal fluctuations* allowing tasks to migrate to high-power cores to ensure performance stability.

3. **Log-Linear Mapping** Given that both processor frequency and voltage scaling (DVFS) follow power laws is the reason that this is applicable to the Boltzmann distribution.

The power consumption P typically scales with voltage V and frequency f as $P \propto Vf^2$. When taking the logarithm of these performance states, the resulting distribution of optimal states takes the form of a Gibbs measure of which the Boltzman distribution is a physical instance of this measure.

4. **Canonical Ensemble Abstraction** The scheduler treats the workload as a Canonical Ensemble:

   - **Total Energy** (U): The power budget or Thermal Design Power (TDP).
   - **Microstates**: The possible configurations of active Big and Little cores.
   - **Partition Function** (Z): The sum of all possible core/frequency combinations.
   - The probability P_i of assigning a task to a core state i is calculated as:

$$P_i = \frac{e^{-\theta C_i}}{\sum e^{-\theta C_j}}. \tag{A.2}$$

Here, C_i is the computational cost, and θ is an arbitrary constant replacing β. This allows the system to achieve an "optimal" distribution of tasks that minimizes total power consumption across the chip without needing to measure actual Kelvins or use physical constants. It uses the form of the law to achieve the efficiency of the law. Data processing was performed by computing the discrete convolution of the isolated energy histograms:

$$h_{pred} = \text{np.convolve}(h_{big}, h_{little}). \tag{A.3}$$

We then calculate the Pearson correlation coefficient (r) and p-value between the predicted distribution h_{pred} and the observed distribution h_{joint}. Thereafter, we look at some of the technical considerations required for these power measurements as follows:

1. **Violation of Statistical Independence:** The mathematical basis of the experiments assumes that the joint probability distribution $P(A \cup B)$ of energy states is the convolution of the individual distributions $P(A) * P(B)$. This operation is valid only if the random variables (energy consumption of the clusters) are independent.

$$P(E_{total}) = \int P_{big}(E) P_{little}(E_{total} - E) dE \tag{A.4}$$

The correlation of near-zero proves that $P(E_{big}, E_{little}) \neq P(E_{big}) \cdot P(E_{little})$. The clusters are stochastically coupled; the energy state of one cluster actively modifies the energy state of the other.

2. **Boltzmann Equation and Entropy:** The Boltzmann relationship defines entropy as $S = k_B \ln W$. For two truly independent systems, the total multiplicity W_{total} is the product of individual multiplicities:

$$W_{total} = W_{big} \times W_{little} \implies S_{total} = S_{big} + S_{little} \tag{A.5}$$

A successful convolution ($r \approx 1.0$) would have confirmed that the system entropy is additive. The result $r \approx 0$ indicates that the system is non-extensive. The interaction energy (E_{int}) between the clusters is non-negligible, meaning the Hamiltonian of the system must be expressed as:

$$H_{system} = H_{big} + H_{little} + H_{interaction} \tag{A.6}$$

Because $H_{interaction} \neq 0$, the total energy is not a simple sum of independent parts, and the Boltzmann distribution of the joint system cannot be derived by convolving the isolated parts.

3. **Theoretical Justification** The use of convolution assumes that the Big and Little clusters are independent random variables. If the total energy $E_{total} = E_{big} + E_{little}$, and E_{big} and E_{little} are independent, then the PDF of the sum is the convolution of their individual PDFs:

$$P(E_{total}) = (P_{big} * P_{little})(E) = \int P_{big}(\tau) P_{little}(E - \tau) d\tau \tag{A.7}$$

If the measured h_{joint} diverges from $h_{convolved}$, it proves that E_{big} and E_{little} are coupled (likely through shared L2 cache contention, thermal throttling, or voltage droop on the shared PMIC rail).

Appendix B

Workload Construction

The arctangent function has a geometric interpretation where arctan(1) represents the angle whose tangent is 1, which is exactly 45° ($\pi/4$ radians). Thus the series converges exactly to $\pi/4$. The series is one of the earliest discovered infinite series for π, discovered independently by Gregory (1671) and Leibniz (1682). It connects π to the alternating sum of odd reciprocals through the arctangent function.

We start with an identity derived from the Taylor series expansion of the arctangent function around x = 0:

$$\arctan(x) = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n+1}}{2n+1} = x - \frac{x^3}{3} + \frac{x^5}{5} - \frac{x^7}{7} + \cdots \tag{B.1}$$

We identify the radius of convergence: $|x| \leq 1$ (with conditional convergence at $x = \pm 1$) by setting $x = 1$:

$$\arctan(1) = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} - \cdots = \frac{\pi}{4} \tag{B.2}$$

Using this series, we select the workload for our analysis utilizing the Gregory-Leibniz Series Identity [46]:

$$\pi = 4 \cdot \arctan(1) \tag{B.3}$$

This identity is the closed form expression of an infinite, slow-converging alternating series. While theoretically elegant, its $O(1/n)$ convergence makes it unsuitable for practical high-precision computation. This is a classic example of why algorithmic choice matters more than raw computational power in numerical methods making it a good choice for our workload.

Appendix C

Execution and Telemetry

Core isolation was enforced using the Linux `taskset` command to pin processes to specific hardware threads. The experiment was performed in three distinct phases:

1. **Phase 1 (Big Isolation):** The workload was spawned on Big cores ($N = 4$ for Orange Pi, $N = 2$ for Cubie). Little cores were forced idle. Telemetry was captured to generate μ_{big}.
2. **Phase 2 (Little Isolation):** The workload was spawned on Little cores ($N = 4$ for Orange Pi, $N = 6$ for Cubie). Big cores were forced idle. Telemetry captured μ_{little}.
3. **Phase 3 (Joint Operation):** Both clusters were saturated simultaneously. Telemetry captured μ_{joint}.

Telemetry was sampled at 100 ms intervals (10 Hz) via `sysfs` interfaces. On the Orange Pi, we targeted '`/sys/class/hwmon`' for PMIC power rail readings. On the Cubie, thermal proxies ('`/sys/class/thermal`') were utilized where direct rail measurement was unstable, as thermal integration serves as a reliable low-pass filter for power density.

In contrast to the board-specific `sysfs` monitoring employed on ARM platforms, telemetry acquisition for the Intel i7-12800H was executed via the Running Average Power Limit (RAPL) interface, utilizing the `perf` subsystem to sample Model-Specific Register (MSR) energy counters at the same 10 Hz resolution. Specifically, the diagnostic framework targeted the `power/energy-pkg/` PMU events, providing architectural-level granularity for the cumulative package power plane. This approach allows for the direct monitoring of silicon-level energy consumption across both performance (P) and efficiency (E) core clusters, ensuring that the measured energy probability measures (μ) reflect internal power plane dissipation rather than the board-level electrical characteristics or motherboard-level filtering latencies associated with `hwmon` or thermal proxies.

Appendix D

Platform Deep Dive

The architectures of the Cubie A7A (Allwinner A733) and Orange Pi 5 (Rockchip RK3588) represent different implementations of the ARM DynamIQ paradigm, which fundamentally alters the energy landscape compared to some other big.LITTLE designs.

Architectural Contrast: Dual-Island vs. Triple-Island Complexity

Feature	Cubie A7A (A733)	Orange Pi 5 (RK3588)
Cluster Topology	$2 \times \text{A76} + 6 \times \text{A55}$	$(2 \times 2) \times \text{A76} + 4 \times \text{A55}$
Power Island Strategy	Dual-cluster (2-B / 6-L)	Triple-island (2-B / 2-B / 4-L)
Interconnect	Single DSU	Unified DSU (Segmented)
Entanglement State	Transient (Linked)	Rigid (Entangled)

Hardware Nuances – The "DSU Ghost" and Shared Rails: The primary cause of entanglement in both chips is the DynamIQ Shared Unit (DSU). Unlike older big.LITTLE architectures where clusters had separate L2 caches and distinct power domains, DynamIQ integrates all cores into a single complex sharing an L3 cache and a Snoop Control Unit (SCU).

The RK3588 "Triple-Island" Failure: The RK3588 splits its performance cores into two clusters (Big0 and Big1). However, the empirical data shows a persistent 0.602 W floor for any inactive Big cluster. This is the "DSU ghost"—the minimum energy required to keep the shared L3 cache and snoop logic powered for the active cluster. Because Big0 and Big1 must communicate via this shared fabric, their power states are physically coupled at the silicon level. The PMU (RK806) maintains high voltage on these rails to prevent cache-coherency latency, resulting in the observed entanglement.

The Cubie A7A (A733) Transient Independence: The A733 has the ability to transitively show independence suggesting its power management firmware may occasionally implements a deeper C-state (sleep state) for the Big cores. When the Big island is truly gated, the Allwinner PMU likely drops the entire voltage domain. The transition back to LINKED indicates a failure to maintain this state, likely due to background OS tasks waking the DSU-level logic, which "re-entangles" the Little and Big domains.

Thermodynamic Implications – Why Convolution Fails: The Boltzmann uniqueness theorem requires that sub-systems be statistically independent to define a unique equilibrium. In these architectures, the hardware nuances violate this in two ways:

1. **Microstate Coupling:** Because cores share the DSU logic, the microstate of a Little core (e.g., a cache hit in L3) is dependent on whether the Big core cluster is powered. Their energy distributions are not independent variables.
2. **Energy Non-Additivity:** In the Orange Pi 5, the total power is not the sum of the islands:

   $$P_{\text{total}} \neq P_L + P_{B0} + P_{B1}$$

   because of the shared "DSU floor." The energy required to run Big0 is $P_{B0} + P_{\text{DSU_Shared}}$. If you then run Big1, you do not add another $P_{\text{DSU_Shared}}$; it is already paid. This non-linearity prevents the mathematical convolution of individual energy densities.

Finding – The 0.602 W Constant: The most critical finding is the rigid 0.602 W floor on the RK3588. This is a hardware-locked value representing the energy cost of coherency. It serves as a physical proof that in modern DynamIQ SoCs, "power islands" are a logical abstraction rather than a thermodynamic reality. The system functions as a single, entangled thermal entity despite software commands to isolate clusters.