3-2021

# Characterizing Security Monitor and Embedded System Performance across Distinct RISC-V IP-Cores

Justin C. Tullos


# CHARACTERIZING SECURITY MONITOR AND EMBEDDED SYSTEM PERFORMANCE ACROSS DISTINCT RISC-V IP-CORES

THESIS

Justin C. Tullos, Captain, USAF

AFIT-ENG-MS-21-M-087

**DEPARTMENT OF THE AIR FORCE**
**AIR UNIVERSITY**

***AIR FORCE INSTITUTE OF TECHNOLOGY***

**Wright-Patterson Air Force Base, Ohio**

AFIT-ENG-MS-21-M-087

# CHARACTERIZING SECURITY MONITOR AND EMBEDDED SYSTEM PERFORMANCE ACROSS DISTINCT RISC-V IP-CORES

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

in Partial Fulfillment of the Requirements for the

Degree of Master of Science in Electrical Engineering

Justin C. Tullos, B.S.

Captain, USAF

March 2021

AFIT-ENG-MS-21-M-087

# CHARACTERIZING SECURITY MONITOR AND EMBEDDED SYSTEM PERFORMANCE ACROSS DISTINCT RISC-V IP-CORES

THESIS

Justin C. Tullos, B.S.
Captain, USAF

Committee Membership:

Scott R. Graham, Ph.D.
Chair

Pranav R. Patel, Ph.D.
Member

Lt Col Jeremy D. Jordan, Ph.D.
Member

# Abstract

Embedded systems have seen a rapid integration into all forms of industry as they continue to shrink in size and cost. The increased demand has highlighted a need for secure systems that are robust to attacks and demonstrate reliable performance, especially if the system operation is time-critical. Efforts to characterize the performance of secure systems have been obstructed either by proprietary restrictions or ineffective analysis.

Proprietary technology limits a comprehensive validation of a system's security and the implications it might have on performance. Performance analysis that is disclosed often lacks sufficient statistical rigor needed for a complex system. An open-source processor standard, called RISC-V, may allow sufficient transparency to thoroughly model performance trade-offs.

This research shows that a security platform and embedded system performance can be characterized through non-parametric statistics methodology, and provides a substantive foundation to scrutinize system design considerations that impact performance. This work proposes a new framework, the Statistical Performance Analysis with Relevance Conclusions (SPARC), that pioneers a synthesis of difference and equivalence hypothesis testing to provide relevant conclusions. SPARC is used to characterize performance of three RISC-V embedded systems with and without a security platform, Keystone, instantiated on an field programmable gate array (FPGA).

*This work is dedicated to God, sobriety, and my loving wife.*

# Acknowledgements

I would like to thank my advisor Dr. Scott Graham for helping me grow both academically and personally throughout my time at AFIT. I could always count on his support and feedback to guide me through the various roadblocks during my research.

I would also like to thank Dr. Pranav Patel, and Lt Col Jeremy Jordan for their feedback and advice for my research. Dr. Patel dedicated countless hours to helping me succeed and he was always available for advice. Lt Col Jordan opened my eyes to statistics and my research was meaningless without him.

Finally, there are not enough pages to express my gratitude to my wife for her support through the long days, late nights, and weekends spent researching at AFIT. You are the sunshine in my life and this would not have been possible without you. Thank you.

Justin C. Tullos

# Table of Contents

Page

Page

# List of Figures

# List of Tables

# List of Acronyms

**API** application programming interface

**ASIC** application-specific integrated circuit

**CLT** Central Limit Theorem

**CUT** component under test

**EM** Entity Monitor

**FPGA** field programmable gate array

**FWER** family-wise error rate

**HDL** hardware descriptive languages

**HPT** Hierarchical Performance Testing

**ISA** instruction set architecture

**ORM** object reference monitor

**OS** operating system

**PMP** physical memory protection

**RM** Resource Monitor

**SGX** Software Guard Extensions

**SPARC** Statistical Performance Analysis with Relevance Conclusions

**SPEC** System Performance Evaluation Corporation

**SRM** subject reference monitor

**SUT** system under test

**TOST** Two One-Sided Tests

**UART** universal asynchronous receiver-transmitter

CHARACTERIZING SECURITY MONITOR AND EMBEDDED SYSTEM PERFORMANCE ACROSS DISTINCT RISC-V IP-CORES

# I. Introduction

## 1.1 Motivation

By 2025, the global military embedded systems market will reach $18.4 billion [2]. Military and civilian applications are seeing a surge of integration with embedded systems as they continue to shrink in size and cost. The increased demand has highlighted a need for secure systems that are both robust to attacks and demonstrate reliable performance, especially if the system operation is time-critical. But securing an embedded system is often a trade-off between costs for the level of security needed, complexity of the solution, and the performance impact. Performance characterization of a secure system is often obfuscated, depending on the architecture of the processor. Closed-source architectures, like Intel or ARM processors, offer security platforms without sufficient insight into implementation details, thus inhibiting system developers in understanding the interactions and implications of using it [3]. The advent of an open-source processor standard, called RISC-V, may enable sufficient transparency of the interactions between system components and a security platform to thoroughly model performance trade-offs.

## 1.2 Problem Statement

The 2011 release of RISC-V, an open-source instruction set architecture, paved the way for a new era of processor technology evolution built around the open

exchange of embedded security, validation, and implementation. An open-source software security monitor developed for RISC-V processors, Keystone provides primitives to secure embedded systems. Integrating Keystone, or any other security platform, will inevitably impact the performance of an embedded system, and that impact on performance must be properly accounted for by system developers.

To assess the impact, a baseline performance characterization of the system is necessary. This is achieved through benchmarking, where a standard set of representative programs are executed to capture performance metrics. But, the resulting metrics often lack sufficient statistical rigor needed for an extensive analysis of a complex system. A number of frameworks have been proposed, notably the Hierarchical Performance Testing (HPT) framework [1], however they have limitations with respect to hypothesis testing and are insufficient in assessing conditions of similar performance that could arise in embedded systems.

The challenge in characterizing performance of Keystone and the impacts it has on a RISC-V embedded system is the lack of methodology available that relies on fundamental statistical inference common across fields of research. This thesis addresses the complexities surrounding traditional performance analysis and suggests a new framework capable of robust analysis which provides results relevant to the system. This research establishes the baseline performance of three RISC-V embedded systems with the new framework and builds upon it with an assessment of Keystone. It also identifies a modeling technique that can determine performance overhead caused by Keystone-specific security features.

### 1.3 Research Goals

This work attempts to characterize security monitor and embedded system performance across distinct RISC-V processor designs with a focus on statistical

methodology. A new framework is developed, called SPARC, to baseline embedded system performance and evaluate a trio of RISC-V open-source processors. Additionally, a modeling framework is used to determine the performance impact that Keystone has on the three RISC-V systems. The research goals of this work are outlined below:

- Develop a statistical methodology, SPARC, for performance evaluations of computers and embedded systems.
- Assess the efficacy of SPARC in relation to a similar framework, HPT.
- Evaluate margins of equivalence in performance evaluations for improved statistical inference.
- Measure the raw application performance impact of Keystone on a RISC-V IP core.
- Model Keystone-specific security features on a RISC-V embedded system and characterize the performance.

## 1.4 Hypothesis

This research hypothesizes that security monitor and embedded system performance can be characterized through non-parametric statistics methodology, and provides a substantive foundation from which to scrutinize architectural design considerations that impact performance. In addition, it theorizes that the performance impact of Keystone's secure execution environments can be modeled to determine an average system response time for embedded systems with Keystone enabled.

## 1.5 Approach

The approach consists of establishing a baseline of benchmark performance of three RISC-V softcore processors, implemented on an FPGA, without Keystone enabled. The RISC-V systems Rocket, Ariane, and Shakti, are open-source and supported the necessary hardware configurations required for Keystone evaluation after the baseline performance characterizations. The performance baseline compares the RISC-V systems (Rocket to Ariane and Rocket to Shakti) for statistical significance, to determine if relevant differences of performance exist. A secondary analysis evaluates the per-core benchmark performance of the embedded system with Keystone enabled and disabled, to assess Keystone's performance impact. Further, margins of equivalence are implemented within a statistical methodology to assess conditions of similarly performing systems. Finally, Keystone-specific security features are adapted and functionally attributed to a simple closed queue network model [4], permitting an average system response time analysis. Other findings revealed during the process are documented and adjudicated.

## 1.6 Contributions

This thesis contributes to the field of computer modeling and embedded system performance analysis through the following:

- Proposed an improved statistical framework called SPARC. It appears to be the first computer performance analysis approach to combine difference and equivalence hypotheses tests and use the results to form four conclusions [5], [6] relevant to a computer performance evaluation under study.

- Implemented non-parametric methodology within the SPARC framework, permitting analysis under distribution-free statistics tests, and developed with a straightforward procedure for implementation. Difference tests are conducted with a Wilcoxon Signed-Rank Test for paired computer performance observations for detecting statistically significant distributions.

- Assessed equivalence within a median tolerance for distributions statistically significant but practically irrelevant.

- Developed SPARC with inspiration from HPT [1]. Minimized the false positive error rate using a multiple hypotheses error correction. SPARC provides scalability based on the number of benchmark programs executed without inflating the error rate.

- Implemented SPARC framework enhances analysis with a conditional feedback loop that discriminates between overpowered or underpowered performance evaluations.

- Evaluated the new methodology with a performance evaluation consisting of a trio of RISC-V softcore processors instantiated on a FPGA.

- Applied a theoretical modeling framework to assess the performance impact of Keystone and Keystone-specific security features. Attributed Keystone's application programming interface (API) through its use of secure enclaves and trusted/untrusted operating system configurations to the model. It allows the overhead cost of using Keystone to be determined as the average system response time.

- Identified an equation that predicts time added to Keystone's initialization code based on the binary size of an application.

## 1.7 Organization

This thesis is organized into 6 Chapters. Chapter 2 provides a brief summary on the embedded system architecture, the security monitor Keystone, introduces the motivating HPT framework, and discusses related work. Chapter 3 provides key fundamentals of statistical analysis with respect to difference and equivalence hypothesis testing. It introduces the SPARC framework, with in-depth procedures to conduct analysis. It also addresses limitations of non-parametric statistics in error correction and sample size estimation. In Chapter 4, the experimental design and methodology is discussed. Chapter 5 presents an extensive analysis characterizing the performance of a trio of RISC-V open-source processors with SPARC and the performance impacts of Keystone. Chapter 6 concludes with a summary of the work presented and future work.

# II. Background and Related Work

## 2.1 Overview

This chapter provides background information about the RISC-V ISA, the Keystone reference monitor, and the HPT framework. It begins by describing how using an open-source ISA provides primitives to secure embedded systems. Next, it illustrates how Keystone operates and provides secure isolation between components. It follows with an outline of the HPT framework and the significance of incorporating a confidence-based analysis model for processor metrics. Finally, a review of current literature explores the complexity of characterizing the performance impact on a system, specifically the impact resulting from the use of secure-execution environments.

## 2.2 RISC-V ISA

The RISC-V ISA was released as a free and open-source specification to promote innovation in processor design [7]. An ISA provides the technical specification of interfacing a RISC-V processor to low-level software [8]. It provides the language and interface requirements necessary to communicate with the combined hardware that constitutes a RISC-V processor including: the machine registers, memory access, input/output, and other functionality [8].

There are two variants of the RISC-V ISA specification: Base User-Level ISA [7], and a Privilege-Level ISA [9]. The Base User-Level ISA was the original specification which defined the RISC-V architecture minus privilege levels and did not include additional hardware security implementations. However, the Privilege-Level ISA specification was released afterwards [9] and defined an architecture through privilege hierarchy for embedded system security; security was

an original consideration of the ISA, not merely an afterthought.

As part of the Privilege-Level architecture, an optional hardware unit defined as physical memory protection (PMP) was developed to allow configurable registers that prevent unauthorized access to memory [9]. PMP is central to the function and operation of the Keystone security monitor on RISC-V embedded systems.

### 2.2.1 Keystone Security Monitor for RISC-V Architectures

Keystone is a modular software security monitor and open-source framework for RISC-V based processors [10]. It builds upon the Sanctum RISC-V processor design by MIT research in [3], but with some distinctions. The primary design improvement, a similar implementation of PMP, that was used by [3] to create a secure, isolated execution environment was incorporated into the RISC-V Privilege-Level specification [10]. With both PMP configuration and privilege levels, Keystone enables secure execution enclaves to prevent unauthorized access.

In Keystone, a secure execution environment is defined as an enclave [10]. It is an area within memory that is isolated from untrusted processes (e.g. operating system) and secured through memory encryption. Hardware isolation of the enclave occurs through the PMP configuration registers, which is configured by Keystone at enclave creation to prevent unauthorized access.

Similar to Keystone, Intel's Software Guard Extensions (SGX) [11] is an alternative implementation of enclaves released for the x86 ISA. The primary difference between Keystone and SGX, not withstanding the different computer architectures, is that Keystone is open-source software and SGX is closed-source proprietary.

#### 2.2.1.1 Security Monitor and Reference Monitor Nomenclature

The nomenclatures security monitor and reference monitor, with respect to embedded systems, are often used inconsistently within computer literature. In embedded system security there are two types of monitors: active and passive [12]. Active security monitors are software modules that monitor behaviors of executable code and can manage computer resources. Passive security monitors are akin to a security policy found in internet router firewalls, either allowing or preventing access [13]. That is to say, passive security monitors do not manage computer resources, they merely enforce usage criteria.

In [4], the authors define a reference monitor as a security mechanism to observe and perhaps control the information flows between two objects. In the context of this research, Keystone is defined as a reference monitor, which is similar to a passive security monitor based on the definitions provided. In the literature for Keystone [10] both security monitor and reference monitor are used interchangeably, but hereafter this research uses reference monitor as the preferred nomenclature.

### 2.2.2 Benchmarks

While individual benchmark programs may be specifically developed to evaluate a computer system's performance, several are often bundled together as a suite of applications. Hereinafter, these individual benchmarks are simply denoted as benchmarks. The System Performance Evaluation Corporation (SPEC) [14] is a popular example of a benchmark suite that can be compiled and executed on a variety of computer architectures. To compare two systems, one merely needs to execute a given benchmark suite on each system, after which the execution times can then be appropriately compared.

In some cases, benchmark programs may be very specialized in order to test

specific functionality of a system under test (SUT); examples include testing floating-point operations or integer multiplication. After the SUT completes a benchmark, performance metrics are reported as time-based or throughput. Often, they are developed as separate software applications rather than originating from a sole benchmark suite such as SPEC. Over time, users consolidate the applications into a suite that is suitable for their requirements. An example is the benchmark suite, RV8, compiled for the RISC-V ISA used later in the text and listed in Table 1.

**Table 1. Benchmark descriptions.**

| Benchmark | Description |
|---|---|
| AES | Encrypt, decrypt and compare 30 MiB of data with 256-bit AES encryption |
| Bigint | Compute $23^{111121}$ and count base 10 digits |
| Dhrystone | Synthetic integer workload |
| Miniz | Compress, decompress and compare 8 MiB of data |
| Norx | Encrypt, decrypt and compare 30 MiB of data |
| Primes | Calculate largest prime number below 33333333 |
| Qsort | Sort array containing 50 million items |
| SHA512 | Calculate SHA-512 hash of 64 MiB of data |

### 2.2.3 Field Programmable Gate Arrays

An FPGA is a hardware platform combination of built-in digital circuitry and reconfigurable-logic components [15]. The reconfigurable-logic components of an FPGA are what distinguish it from an application-specific integrated circuit (ASIC).

There are several advantages that FPGAs provide over ASICs, often associated with costs and reconfiguration. After an ASIC is manufactured, any design changes that alter the circuit often require it to be manufactured again [15]. This process can lead to higher manufacturing and sustainment costs over the product's release, but it also can affect development costs. Prototyping is an early development process in which a design undergoes multiple changes until a final design is established [15].

Frequently, a determination is made to use an FPGA as the prototyping hardware platform for an ASIC if the costs for manufacturing prototype ASIC designs exceed a threshold. The reconfigurable-logic components on an FPGA can be configured with software such that their behavior emulates the behavior of an ASIC design [15], reducing the time it takes to make and test design changes.

The configuration of logic and behavior on an FPGA occurs through software code defined as hardware descriptive languages (HDL). HDL is a software-compiled language that defines how to simulate the behavior of a circuit component [15]. Further information regarding FPGAs and HDL can be found in [15].

### 2.2.4 Broad Analytical Model of a Reference Monitor

A reference monitor was previously defined as a passive security mechanism, observing interactions and enforcing information flows between two objects [4]. In the proposed framework in [4], the authors use a subject-object model; objects are components acted upon by subjects. In terms of an embedded system: objects are resources like memory or processors; subjects are active processes within an operating system (OS) or other executable. Depending on the system configuration, any number of subjects can act upon any number of objects. To assure security, a reference monitor enforces a security policy to allow authorized accesses and prevent unauthorized accesses to resources. The reference monitor is also considered a subject, but one that is activated anytime a flow of information occurs between a subject and an object.

The original concept, depicted in Figure 1, defined the reference monitor with two major purposes: an object reference monitor (ORM), and an subject reference monitor (SRM) [4]. The SRM ensures that subjects (i.e. processes) instantiated are authorized as per the security policy; an external third-party trying to execute

malicious code with an elevated privilege level is one example. The second function, ORM, enforces any security policies that govern usage of resources in an embedded system. Both functions are needed based on the defined properties that encompass a reference monitor and the security policy it implements.

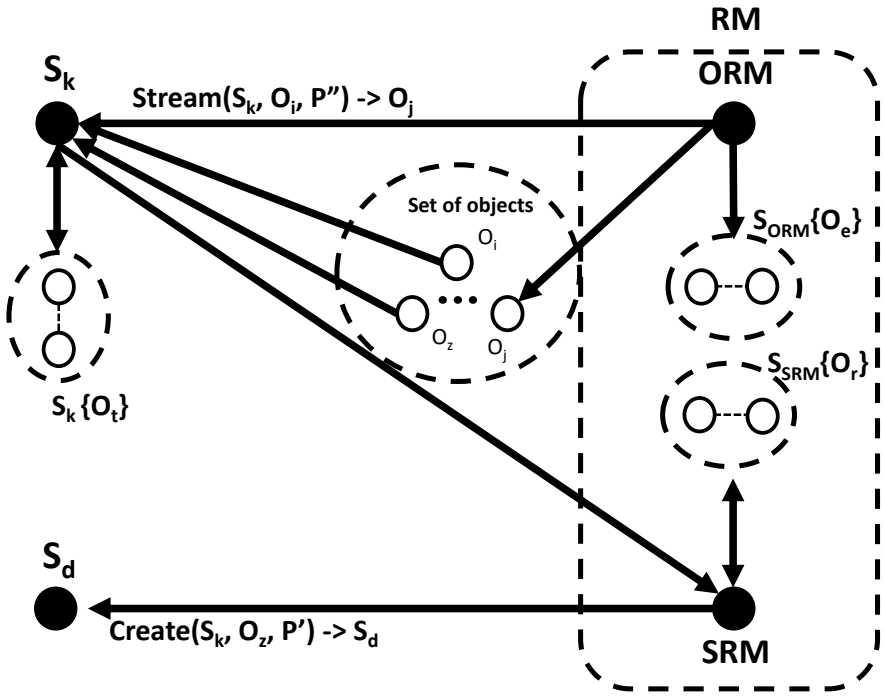



**Figure 1. Reference monitor concept model. ©2019 IEEE. Reprinted, with permission, from Gorbachov et al.**

Analyzing the concept of authorized and unauthorized accesses by a reference monitor, the authors concluded that this concept could be represented by the closed queuing network modified with system descriptions in Figure 2. Including a reference monitor in a system's design is expected to incur a performance impact on its operation. The model includes both the SRM and ORM, in addition to an access controller that operates as the entry point for reference monitor control.

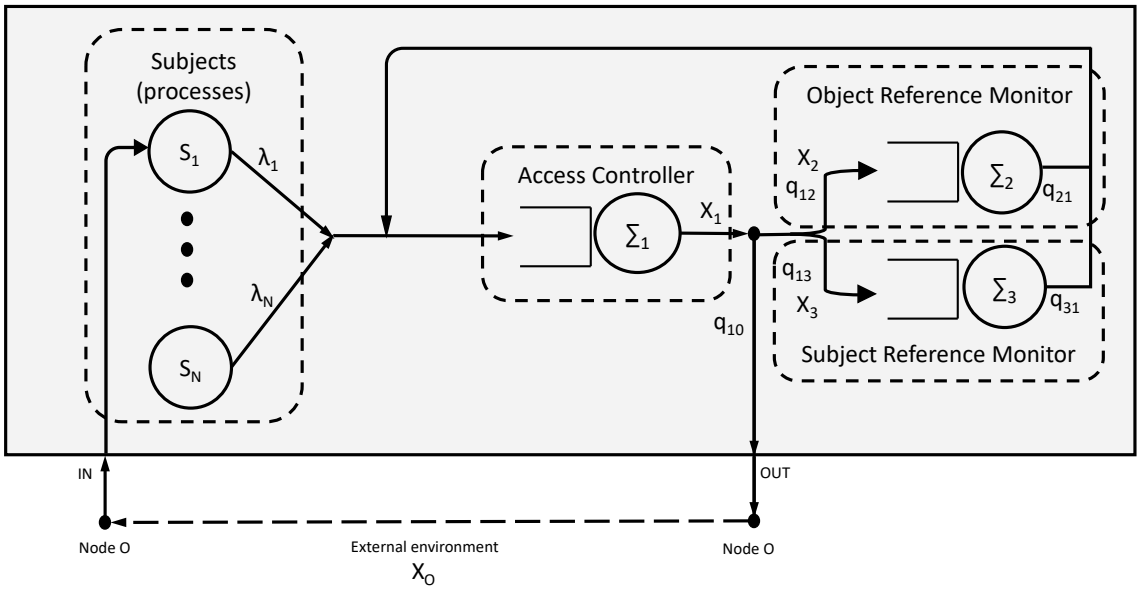



**Figure 2. Reference monitor functional model derived from Gorbachov et al.**

Tracing the information flow from Subjects $S_1, S_2, \ldots, S_N$ begins when the information request is first received by a limited-queue access controller $\Sigma_1$; based on the security policy and type of request, and either sent to the ORM $\Sigma_2$ for access to a system resource or sent to the SRM $\Sigma_3$ for a process activation. The response is then delivered back to the access controller $\Sigma_1$ for further adjudication; it could be sent back to the subject as a denial of service or processed dependent on the policy. $X_1, X_2, \ldots, X_N$ are throughputs into each queue; $q_{ij}$ is the routing frequency from one node to another; $\lambda_1, \lambda_2, \ldots, \lambda_N$ are the rates of request sent by subjects [4].

The performance impact of a reference monitor is then defined by [4], through a system of flow balance equations, leading to Equation (1):

$$R = \frac{N}{X_0} - \frac{1}{\lambda}, \tag{1}$$

where $R$ is the average response (processing) time of the reference monitor, $X_0$ is the throughput of the reference monitor to Subjects $S_1, S_2, \ldots, S_N$, $\lambda$ is the rate of requests for access, and $N$ is the number of Subjects.

### 2.2.5 HPT Framework

In [1], the authors developed the non-parametric HPT framework to promote statistically sound computer performance evaluations. The framework is a methodology using difference hypothesis tests to compare benchmark suite results between two computers to determine superiority. The authors reveal common errors made with respect to parametric and non-parametric statistics while conducting performance evaluations.

Chen *et al.* illustrates the improper use of parametric statistical tests, such as the t-test, on non-normally distributed computer performance data. If the data collected from a computer benchmark is not properly characterized prior to statistical analysis, it could be incorrectly assumed to be parametric instead of non-parametric. Without appropriate verification tests, an assumption of the underlying distribution of the data may contribute to a faulty analysis and misled conclusion of the comparison. They evaluate a SPEC benchmark suite comparison that displayed a skewed non-normal distribution using the t-test which resulted in transforming the data to normality. The t-test concluded the under performing computer was superior, demonstrating the deficiency in assuming a distribution.

The Central Limit Theorem (CLT) is often used to characterize distributions as approximately normal given a large sample size [16]. Frequently, a minimum sample size of 30 or more is referenced in statistics to employ the CLT. Although, this was disputed for computer performance distributions in [1] with an experiment consisting of $32,000$ benchmark performance scores. The analysis reveals that a sample size of

approximately 500 observations still deviated from normality, but could be sufficient to utilize the CLT. Executing a number of benchmarks within a suite, 500 times each, appears inefficient based on the inconsistency of the data.

Many sources of variability and non-determinism exist within a computer system and the complex layers of interactions they are comprised of, discussed in [1]. Further, published performance evaluations routinely omit confidence intervals (CI), which provide a measure of the randomness of a variable and accuracy estimate of observed data [17]. A $100(1-\alpha)\%$ confidence interval on the variable $\theta$ and the probability of the numeric values it can represent between $[L_1, L_2]$ is defined as:

$$P = [L_1 \leq \theta \leq L_2], \tag{2}$$

where $L_1$ and $L_2$ represents the lower and upper bound on $\theta$ values.

Performance evaluations often report a collection of mean completion times or relative speedups and declare one to be superior, with little, if any, documentation of statistical methods used in the comparison. While the mean completion time or speedup serves a purpose as a visual exploration of data, incorporating additional statistics provides insight into the origination, or population, of the sampled data. Such insight is fundamental in determining the accuracy of observations and conclusion. Excluding statistical analysis undermines the original intent behind the performance comparison.

Thus, the authors in [1] developed the non-parametric framework to promote statistically sound computer performance evaluations. The HPT framework is a methodology using hypothesis testing to compare benchmark suite results between two computers to determine superiority. The significance level, $\alpha$, is chosen prior to conducting the hypothesis test; standard suggestion is 0.05 for a one-tailed or 0.10 for a two-tailed hypothesis test.

In order to analyze the performance between two computers on a suite of benchmarks, a series of steps, which comprise the HPT framework were outlined by [1]. This research presents the following abridged procedure for reference, and builds upon it later in the text. Suppose a benchmark suite is used that contains $n$ benchmarks and each benchmark is repeated $m$ times. Matrices $C_A = [a_{i,j}]_{n \times m}$ and $C_B = [b_{i,j}]_{n \times m}$ must be constructed for both computers; rows represent the $n$th benchmark and columns represent the $m$th benchmark repeat of performance scores [1].

For each benchmark, a null ($H_0$) and alternative ($H_1$) hypothesis are tested for significance using a Wilcoxon Rank-Sum Test. If the results show statistical significance, reject $H_0$ that both computers are equivalent; else fail to reject $H_0$. After the Wilcoxon Rank-Sum Test is complete for all $n$ benchmarks, assign to a new column the score representing difference in medians on significant results for each benchmark or assign a 0 for insignificant results.

Concluding HPT is a comprehensive hypotheses test consisting of $H_0$ of general equivalent performance or $H_1$ general superior performance [1]. A Wilcoxon Signed-Rank Test is completed on the difference in median performance scores to either reject $H_0$ or fail to reject $H_0$ at the significance level.

### 2.3 Related Work

The choice between an open-source ISA such as RISC-V, or a closed-source ISA such as Intel x86, has design implications that could affect security. The software and hardware architecture of an embedded system is dependent on a chosen processor's ISA. For example, selecting a RISC-V processor requires software compiled for the RISC-V ISA and similarly, selecting an Intel x86 processor will require code to be compiled for x86 platforms. In [18], the authors, who also

developed and released the RISC-V ISA, provide attributes of closed-source ISAs that hinder processor and embedded system technology. A transparent open-source ISA promotes verification by third-parties; security mechanisms built into the ISA can be independently corroborated for correctness. Both Intel and Arm ISAs include security measures, SGX and TrustZone respectively, that lack independent verification due to their proprietary nature.

Defining the core security properties of a reference monitor and applying a model that encompasses all threats and vulnerabilities is arduous. The closed-queuing network model proposed in Section 2.2.4 was simple and theoretical; testing the operational viability of it is needed in order to refine and improve the framework over additional iterations. The foundation of the analytical modelling framework for a reference monitor defined in [4] was an extension of the mathematical proofs found in [19] regarding reference monitor obfuscation.

A performance loss estimation model viability study was discussed by [20] to characterize SGX for x86 platforms. Efforts would be focused on algorithms that balanced mapping specific platform protections to programs based on the security needs [20]. Correlating protections to performance loss is complex as there are a variety of factors that overlap between them that can affect performance; there is not a clearly defined one-to-one relationship [20]. The follow-on research has yet to be released but could provide an additional framework to test against Keystone on RISC-V platforms.

Weichbrodt, Aublin and Kapitza [21] released an enclave performance analysis tool, sgx-perf, to explore SGX-based programs. The primary impacts to enclave performance are 1), context-switching while a program is executing and 2), paging events [21]. The same two factors are prevalent in Keystone execution; context-switching is the most frequent operation during enclave execution and paging events

suffer additional overhead due to input/output memory operations [10]. The sgx-perf software was developed due to the lack of sufficient measurement and analysis tools surrounding SGX enclaves [21].

In [22], the authors established a clustering method to model distributions of computer performance metrics. Observed data from benchmark distributions were non-parametric and density plots indicated bimodal and multimodal distributions. They surmised that the non-parametric distributions are a Gaussian mixture, a combination of multiple Gaussian distributions of clustered multivariate data. The clustering method determines population estimation parameters that could be used with more powerful parametric statistical tests over non-parametric.

The HPT framework in [1], VarCatcher framework in [23], and methodology in [24] illustrate the complexity of conducting a robust analysis and the lack of statistical rigor surrounding traditional computer performance comparisons. While [1] relies on difference hypothesis testing with non-parametric statistics, [23] and [24] cite the lack of relevant information at the conclusion of hypothesis testing as motivation for their respective custom frameworks.

## 2.4 Summary

This chapter presented a brief summary of the RISC-V architectures and technologies to secure a RISC-V embedded system. In addition, it discussed benchmark programs which are used to characterize the performance of computer or embedded systems. The HPT framework methodology [1] was outlined, which provides a statistical analysis framework for comparing the performance between two computers. Further, a theoretical closed-queuing network model was introduced to evaluate the performance impact of a reference monitor on an embedded system. Finally, this chapter observed related research in characterizing the performance of

embedded systems and the complexities therein if a secure execution environment is present. There is a lack of research in characterizing the performance impact of embedded systems with and without a secure execution environment. This thesis contributes to the field of both computer and embedded system performance analysis by introducing a new statistical framework that can characterize the performance of systems with and without secure execution environments.

# III. Characterizing the Performance of Embedded Systems

## 3.1 Overview

This research introduces two frameworks for characterizing the performance of an embedded system and a reference monitor: the SPARC framework, and an applied analytical model for evaluating a RISC-V reference monitor. Through decomposition, the RISC-V embedded system of focus in this research is modeled through abstracted architecture layers into two sub-systems for easier analysis; one sub-system with the reference monitor Keystone enabled and the other with Keystone disabled. Subsequently, the SPARC framework methodology is described in order to provide a statistically rigorous analysis of both sub-systems and their benchmark performance. To analyze Keystone-specific security features and their performance impact to an embedded system, the closed-queuing network model presented in Chapter II is extended. This chapter presents the sub-system layered models, SPARC, and the closed-queuing network model used in the experimental analysis.

## 3.2 Modular Framework Design

A common approach to modeling an embedded system's architecture is through stacked layers based on functionality [25]. The layers are depicted as an abstraction of specific functions that are required for a system to operate within designed specifications. The layered model is advantageous as the functions are independent and modular, allowing for functions to be added or removed to a system design as requirements change. It also provides a deterministic structure of development; the system is developed at the bottom layer first and proceeding up to the next higher layer until the final layer is completed.

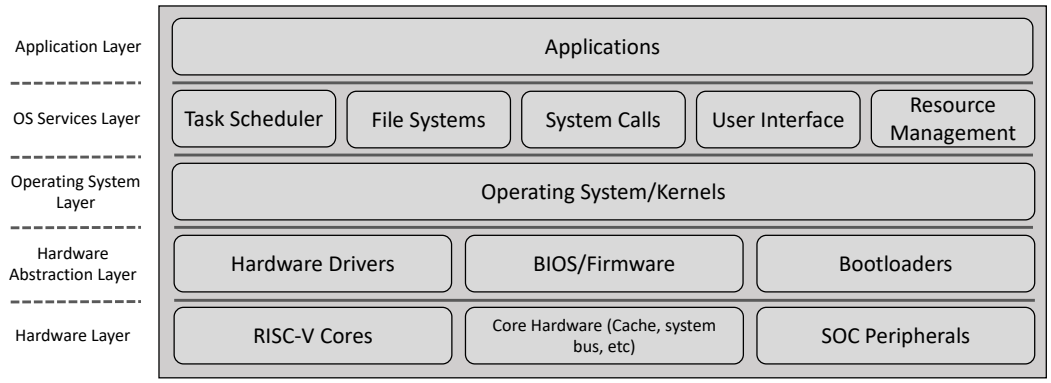



**Figure 3. Architectural layers of a RISC-V embedded system with Keystone disabled.**

There are two sub-systems modeled that contribute to the embedded system's performance: Keystone enabled and Keystone disabled. The RISC-V embedded system, with the Keystone reference monitor disabled, is modeled with layers in Figure 3. Five architectural layers representing the embedded system are described as follows:

- The Hardware Layer describes the tangible components that the embedded system is built of. This includes the RISC-V cores that are the primary functional units that process data or instructions as required by design [25]. Hardware that supports the processor (e.g. the processor cache, system bus, memory, etc.) is located here along with any peripherals used for internal or external connection.

- The Hardware Abstraction Layer provides low-level software that is necessary to interface to the hardware. Bootloaders are often, but not always, the first software executed upon power-on and contain a minimal set of unique hardware drivers and firmware needed for the embedded system. It provides a basic level of operating functionality or as a prerequisite for the OS Layer.

- The OS Layer contains the OS or kernel and is a set of software libraries with additional abstraction, serving as an interface between unique software in the Hardware Abstraction Layer and OS services [25].

- The OS Services Layer specifies software libraries that manage and automate system functions as required by the embedded system. For example, resource management is a required function often automated in software by the OS Services Layer. The additional software packaged within this layer can also provide a standardized, stable, and user-friendly interface for running applications. Software developers can build an application for a specific OS version that has service drivers commonly distributed with it. This improves application stability and maintainability, but it can prevent required automated code from being compiled with an application if it is already present in the OS version.

- The Application Layer interfaces with the OS Services Layer for any system or function calls required to execute user-level code.

Each of the architectural layers within the embedded RISC-V system can affect the performance of a running application, as they are interconnected and rely on the same resources. Within the OS Services Layer, automated system functions like the Task Scheduler could interrupt a running application to run another task unrelated to the application. This could add time to the running application, delaying its completion. Although there may be benefits to running an OS with an embedded system, such as ease of use, there may be disadvantages, such as performance impacts, including the possibility of additional execution time when the OS interrupts to perform unrelated tasks. These tasks may be necessary, such as automated system functions for periodic maintenance, perhaps to prevent errors, or even system crashes from occurring.

An embedded system often has custom user requirements, and the functions may be presented at varying levels of abstraction. Accordingly, the performance of an application will also exhibit variations between embedded systems depending on any functionality differences in the layers. This research defines the RISC-V embedded system for further performance evaluation without Keystone as depicted in Figure 3.

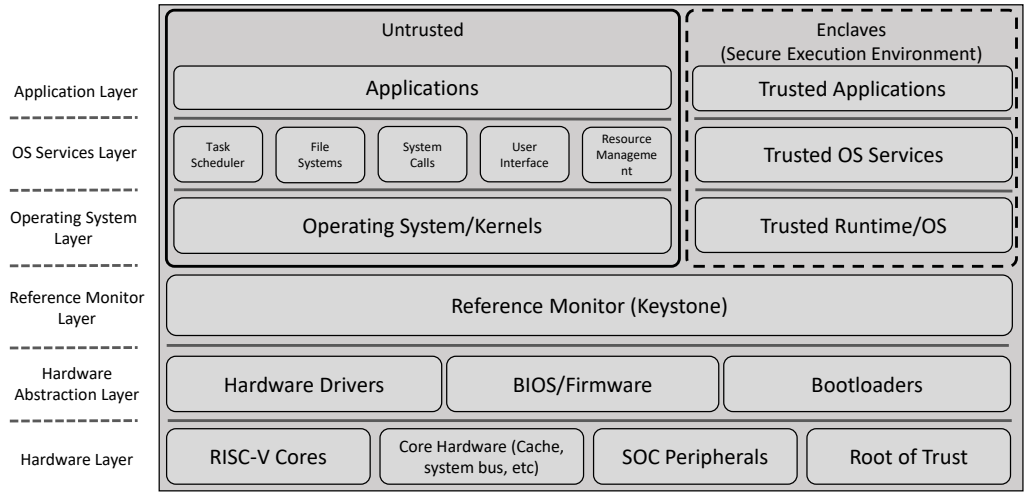



**Figure 4. Architectural layers of a RISC-V embedded system with Keystone enabled.**

Similarly, a RISC-V embedded system with Keystone enabled is modeled through architectural layers depicted in Figure 4. As discussed in Section II, Keystone provides two primary functionalities: a secure boot method and the ability to provide a secure execution environment for applications. Keystone accomplishes both through security primitives offered by the RISC-V Privileged ISA [9]. This research is primarily concerned with the performance impact to a system with respect to Keystone's secure execution environment and its effect on a running application, rather than a system's startup performance affected by secure boot.

The addition of Keystone within the model presents another layer that can affect performance. The Reference Monitor Layer contains the Keystone software that

serves as an intermediary between the OS Layer and Hardware Abstraction Layer. Temporarily ignoring the addition of a secure execution environment, adding another layer to the system could impact performance of an application. On the other hand, the primary distinction of a system with Keystone enabled is the ability of a user to purposefully run an application within an enclave (i.e. secure execution environment). As such, applications using the additional security features provided by Keystone are dependent on the trusted functions as well as untrusted functions within the system layers.

Based on the two system models, this research characterizes the performance impact of an application, first by capturing a performance baseline with the RISC-V embedded system without Keystone, as depicted in Figure 3, and then through the system with Keystone enabled in Figure 4.

## 3.3 The Statistical Performance Analysis with Relevance Conclusions Framework

In this section, a new statistical framework, SPARC, is introduced which was inspired by HPT [1]. It is a method for establishing a baseline of performance of an application on a RISC-V embedded system. Similar frameworks were lacking in the statistical rigor needed to characterize an application, hereafter defined as a benchmark, on an embedded system. SPARC incorporates equivalence tests and family-wise error correction associated with multiple hypotheses tests. It reduces Type I errors and supports various conclusions for relevant and practical results of a performance evaluation.

### 3.3.1 Statistical Significance Versus Practical Relevance

This research forces a clear distinction between two ideas that are often pooled incorrectly in hypothesis testing: statistical significance and practical relevance. Significance is the ability of a statistical test to detect an effect size [16] and is correlated to the type of test used. In difference hypothesis testing [26], failing to reject $H_0$ (i.e. lack of significance) does not imply lack of effect. Conversely, a difference hypothesis test that rejects $H_0$ (i.e. detects significance) expresses nothing about the practical relevance of the result. A statistical framework that only uses difference hypothesis testing is limited to identifying changes and does not address conditions of equivalence, or similarity between population samples within a margin.

To illustrate the limitation of difference tests, consider an exploratory data study to evaluate the performance of Keystone and its impacts to the Rocket RISC-V embedded system. Rocket was instantiated on an FPGA and performance metrics were collected. The experiment was performed with and without Keystone enabled, 30 times each, and execution times were recorded. For statistical analysis, the HPT framework was used, prior to SPARC development, on the results to assess suitability towards the extensive performance evaluation conducted later in this research. Additionally, a sample size of 30 was chosen to assess normality of the distribution with the CLT. A density plot is provided of two performance score distributions in Figure 5. Hereinafter, the decimal precision within this document is limited to three digits with rounding for display purposes only, actual experiment calculations are conducted without rounding.

The difference hypothesis test resulted in a statistical significance between the two distributions as shown in the figure. The median execution time of the program with Keystone enabled as compared to disabled is 263.287 seconds and disabled is 263.677 seconds, or a percentage difference of 0.148% between them. But, the

practical relevance of 0.1479% in the application was minuscule. The two execution times would have been concluded as approximately equivalent. Primarily conducting a difference hypothesis test excluded a condition in which both distributions would be considered equivalent.

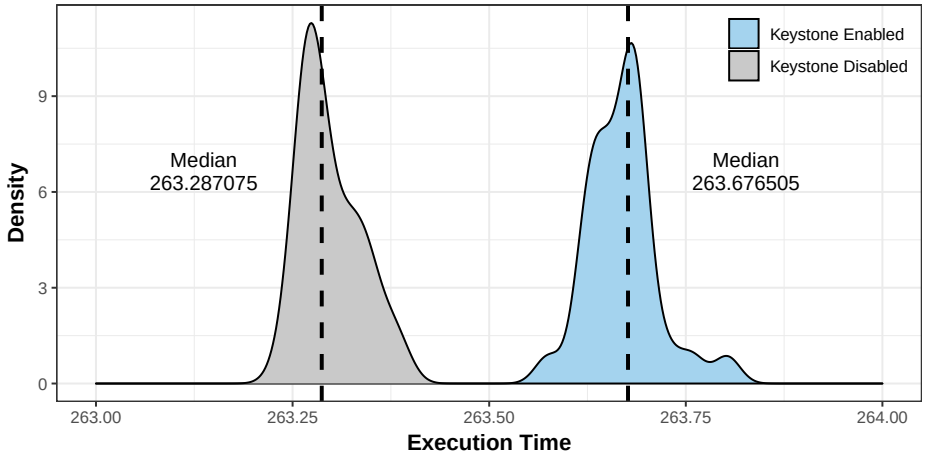



**Figure 5. Comparing two distributions of execution time with Keystone enabled and disabled.**

This analysis highlights another key limitation of difference tests: the constructed null hypothesis is illogical and a difference is always detected with sufficient samples [27], [28]. In the analysis, the null and alternate hypotheses tested either a 0 difference between the two continuous response variables, or a difference detected, respectively. The test is structured given $H_0$ being true and if the probability distribution of the test statistic is low, then $H_0$ is rejected. But, this structured argument for point or exact null is a fallacy and has been debated for decades [27], [28], [29], [30], [31], [32]. The probability that a continuous random variable assumes any specific value is zero [17]. Likewise, with sufficient population samples the test will always detect a difference [33].

In embedded system performance evaluation, inadvertent data manipulation

often occurs either due to rounding, or with an insufficient context of data output. Both can lead to incorrectly assuming that two response variables are equal or that the difference between them is 0 and affect the study. Returning to the example experiment, observe the original density plot with and without Keystone enabled in Figure 6. The graph illustrates the response variable density, execution time, which defaults to a precision of 5 decimal points. The plots are overlaid with a modified response variable density, generated by deliberately rounding data to a precision of 1 decimal point.

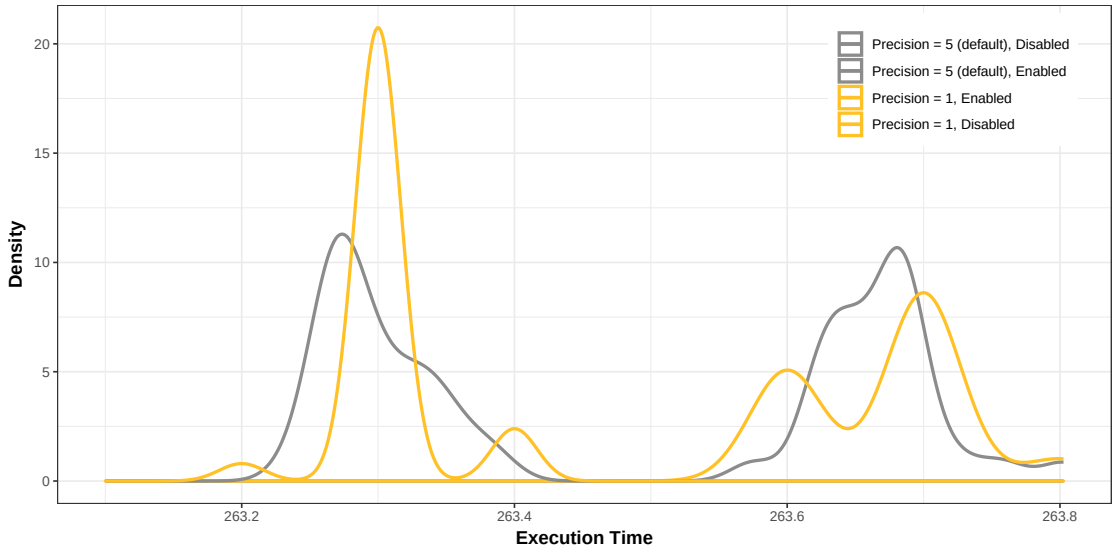



**Figure 6. Comparing two distributions of execution time with the effects of differing decimal precision.**

As shown in the figure, the characterization of the data distribution has altered significantly. Notably, the illustration fails to capture how altered data can proliferate into a difference hypothesis test. The differences to the $n$th decimal that once characterized the data are filtered out along with any insightful conclusions that could be derived. While it might seem obvious, the issue is highlighted after encountering it in computer performance analysis research within the field. The

aforementioned criticisms are not limited solely to computer performance evaluations, but to the field of null difference hypothesis testing in general.

### 3.3.2 Elements of Relevant Statistical Performance Evaluations

A key element in any statistical experiment, including benchmarking, is designing the experiment such that results provide valid statistical information required for analysis. Design of experiments [34] is a field of study dedicated to this aim. The SPARC methodology focuses primarily on non-parametric statistical tests after benchmark data has been collected and assumes the experiment uses an appropriate design. But this research addresses three essential elements for consideration prior to conducting any data collection: 1) standardized hypotheses notation; 2) family-wise error correction; and 3) sample size estimation. Error correction and sample size estimation are implicitly linked when considering multiple benchmarks for statistical analysis; the number of samples affects the significance of a statistical analysis and the significance is affected by the overall error rate for the evaluation.

#### 3.3.2.1 Standardized Hypotheses Notation

Before formally presenting the rationale behind equivalence tests, a standardized hypotheses notation is provided for use throughout this thesis. In the last section, this work discussed limitations of an analysis that uses difference hypotheses and motivated the addition of equivalence tests to SPARC.

First, the term positivist theory is introduced; it is derived from [35] to describe difference hypothesis tests. That is, the null hypothesis of a difference test, $H_0$, is often defined as the lack of an effect or no difference between effects and is tested against an alternative hypothesis $H_1$ of significant effect or difference [26]. Positivist

theory simply denotes $H_0$ and $H_1$ hypotheses of difference tests as $H_0^+$ and $H_1^+$, incorporating the + symbol to reflect testing for a significant effect. Likewise, the term negativist theory is introduced [35], to describe equivalence hypotheses that test for a lack of effect (i.e. equivalence). Negativist theory defines the equivalence hypotheses $H_0$ and $H_1$, as $H_0^-$ and $H_1^-$. Within this document, the positivist and negativist theory notations $H_0^+$, $H_1^+$, $H_0^-$, and $H_1^-$ are used to differentiate between difference and equivalence hypotheses.

#### 3.3.2.2 Multiple Hypothesis Error Correction

In the following, let $X = x_{i,1}, x_{i,2}, \ldots, x_{i,m}$ and $Y = y_{i,1}, y_{i,2}, \ldots, y_{i,m}$ for $i = 1, 2, \ldots, n$ denote independent samples of performance scores from Computer X and Computer Y on the $n$th benchmark, respectively. Each hypothesis test performed in a multiple evaluation experiment increases the probability of rejecting $H_0$ when $H_0$ is true (Type I error) defined as the family-wise error rate (FWER) [36]. In other words, in a family of comparisons that are related, the false positive error rate increases [36]. The worst-case FWER for $n$ total benchmarks tested at an $\alpha_n$ is:

$$FWER \leq 1 - (1 - \alpha_n)^{\beta+1}, \tag{3}$$

where $\beta$ is the number of benchmark tests plus an additional overall hypothesis of general performance.

Using an appropriate error correction method, the family-wise error correction can be controlled in the performance evaluation while still providing statistically significant results [36]. Each hypothesis test used to analyze a benchmark increases the FWER and requires correction. There are two methods introduced here, the Bonferroni Correction [37] and Holm-Bonferroni Correction [38]. Each $\alpha$ correction method has its advantages and disadvantages that should be considered depending

on a study's requirements. In the RISC-V performance characterizations later in the document, the Bonferroni Correction is used.

There are two benefits for using the Bonferroni Correction. First, it is a simple correction applied to every test in a performance study and, second, it allows calculating confidence intervals across benchmark comparisons [38]. It is widely used but has also been criticized as overcorrecting $\alpha$ to reduce Type I errors and subsequently reducing the probability of detecting any significance [36]. The method to calculate an error corrected $\alpha_{New}$ is as follows:

$$\alpha_{New} = \frac{\alpha_{Old}}{(n+1)}, \tag{4}$$

where $n$ is the total number of benchmarks planned plus the overall hypothesis test and $\alpha_{Old}$ is the overall requested alpha (0.05 for one-tailed, 0.10 for two-tailed tests).

After $\alpha_{New}$ is calculated, the p-value of each benchmark hypothesis test is compared with $\alpha_{New}$ to either reject $H_0$ or fail to reject $H_0$:

$$p_n \leq \alpha_{New}, \tag{5}$$

where $p_n$ represents the p-value of the $n$th benchmark.

An alternative method that does not overcorrect $\alpha$ is the Holm-Bonferroni Correction [38]. The method corrects sequentially, calculating $\alpha_{New}$ for each p-value comparison. While it provides stronger statistical power compared to Bonferroni, there is added complexity to determine confidence intervals based on a changing $\alpha_{New}$. The procedure is presented here as it pertains to the SPARC framework as an option if confidence intervals are not required. Let $p_n$ be denoted as the p-value calculated after conducting the Wilcoxon Signed-Rank Test, for the $n$th benchmark. Sort in ascending order such that $\{p_1 < p_2 < \cdots < p_i$ for $i = 1, 2, \ldots, n$. Assign

$\alpha_{New}$ based on ranks of the test until the first non-significant result is found (failed to reject $H_0$) and the correction is complete. Any further benchmark hypothesis tests are non-significant. The equation for this procedure is as follows:

$$p_n < \frac{\alpha_{new}}{i + 1 - n}, \tag{6}$$

#### 3.3.2.3 Sample Size

In computer performance evaluations, determining the proper sample size is a fine balance between under or over sampling for a proper test. The significance (p-value) of each benchmark analysis is correlated with the sample size [39]. If an insufficient number of samples are collected from a benchmark, there is risk of an underpowered test (i.e. not providing a significant result due to a low p-value). If an over abundance of samples are collected, then the risk is an overpowered study that inefficiently used resources.

There are multiple ways to calculate sample sizes for the parametric t-test statistic based on an effect size estimate, such as Cohen's D [40], if the underlying distribution is known or assumed. However, for non-parametric statistics tests this research makes no assumptions on the underlying distribution. The methods in [41], [42], however, illustrate how an estimated sample size can be determined for the Wilcoxon Signed-Rank Test if assumptions are made on the effect size and an unbiased estimator through a resampling process. Suffice it to say, there is merit in applying the techniques to a computer performance evaluation to reduce the number of benchmark repeats or increase power of the tests. At the same time, execution times are often non-deterministic which suggests resampling observations with prior data could affect the outcome or provide inaccurate sample size estimations. While there are no clear methods available for sample size estimation

suitable for the SPARC framework, two of the resulting outcomes will report if a benchmark test was underpowered or overpowered.

### 3.3.3 Equivalence Testing

Instead of testing the significance that performance scores from two computers are different, another approach is introduced, called equivalence testing [43], [44]. In difference testing, one attempts to prove the alternative hypothesis $H_1^+$ of a significant statistical difference. If the test fails to reject the null $H_0^+$ of no difference, then the only conclusion is a lack of evidence to reject $H_0^+$. A conclusion of equivalence cannot be valid because it was not tested. By adding equivalence hypotheses tests to the framework, it presents additional information to make inferences of a performance evaluation.

Equivalence testing is often found in clinical settings to assess whether the effect of two treatments or medications are within a predefined equivalence margin [43]. The burden of proof for equivalence resides in the alternative hypothesis $H_1^-$. An equivalence margin $[-\delta, \delta]$ establishes the range in which two variables contained within are considered practically equivalent at $\delta$. In context of the SPARC framework, the equivalence margin renders two statistically significant but practically irrelevant performance score distributions as equivalent if the response variable is within the predefined $[-\delta, \delta]$ interval.

One widely used method for equivalence testing is the Two One-Sided Tests (TOST) procedure in [45]. Choosing an appropriate equivalence margin $\delta$ for the TOST is paramount to a performance evaluation [43]; selecting a $\delta$ which is too stringent risks excluding practically equivalent performance scores, and selecting a $\delta$ which is too broad risks false equivalence. [43] proposed either using past studies or pilot studies to establish a $\delta$, but this research considers it unsuitable for computer

performance evaluations as preexisting data is either lacking or includes components specific to an embedded system. Instead, this work suggests to tailor it to the evaluation depending on the motivation and context of the study.

Alternatively, a performance ratio between two systems, commonly referred to as the speedup ratio, can be selected as the $\delta$. Suppose two computers, Computer X and Computer Y, execute a benchmark and the resulting times are compared. Let $S_{Y/X}$ (Speedup ratio of Y compared against X) denote the execution time relation of Computer X to Computer Y, defined in Equation 7.

$$S_{Y/X} = \frac{Execution\_Time_X}{Execution\_Time_Y} \tag{7}$$

Therefore, $S_{Y/X}$ denotes the increase or decrease of Computer X's benchmark time to Computer Y's benchmark time. In this text, performance increase (i.e. better performance) means minimizing the execution time of a benchmark. For example, an $S_{Y/X} = 1$ would indicate that the execution time of Computer X is equal to Computer Y. Whereas, an $S_{Y/X}$ greater than 1 would indicate the execution time of Computer X was higher than the execution time of Computer Y; another valid statement would be Computer Y's performance was better than Computer X's. As a note, consideration for the numerator and denominator is important. $S_{Y/X}$ is not equal to $S_{X/Y}$; they are two different relations.

In this document, a speedup ratio of 0.05 is typically used for $\delta$ resulting in the margin, $[0.95, 1.05]$. To put it in terms of $S_{Y/X}$, if the ratio of Computer X's execution time to Computer Y's execution time is within 0.05 of each other then the two times are within the margin and considered equivalent.

### 3.3.4 Combining Difference and Equivalence Hypotheses

Combining hypotheses tests for difference and equivalence leads to practical and relevant conclusions not possible individually. Hypothesis testing for difference supports conclusions for statistical significance but lacks conditions for practical irrelevance or equivalence. Conversely, equivalence testing supports conclusions on equivalent distributions but lacks conditions for substantial performance differences that are of interest. Therefore, the prevailing solution is a combination of difference and equivalence testing for practical relevance [5], [6].

The following text outlines the procedures in the SPARC framework for combining the two types of tests for a relevant performance characterization. The method changes the Mann-Whitney (Wilcoxon Rank-Sum) Test in [1] to a Wilcoxon Signed-Rank Test for paired observations. Although, with minor alterations, these procedures can still be applied to the Mann-Whitney Test. The RISC-V systems evaluated in a later chapter necessitated a paired non-parametric test.

Suppose a study is evaluating two computer's performance on a benchmark suite consisting of $n$ benchmarks, each repeated $m$-times. Let $(x_i, y_i)$ be the $ith$ pair for $i = 1, 2, \ldots, m$ for Computer X and Computer Y of $m$ observations on the $nth$ benchmark. Construct matrices $B_n = [x_{i,1}, y_{i,2}, r_{i,3}]_{m\times 3}$ for $n = 1, 2, \ldots, n$ for $n$ benchmarks. Let $r_i$ denote the pairwise ratio $x_i/y_i$ for $i = 1, 2, \ldots, m$ and $M_{X/Y}$ denote the median pairwise ratio of performance. A Wilcoxon Signed-Rank Test is used under the assumption that the ratios $r_i$ are continuous and symmetric around a common median $\theta = 1$ [1]. Difference hypotheses for the two-tailed Wilcoxon Signed-Rank Test are defined as:

- $H_0^+$: *the median performance score ratio* $M_{X/Y}$ *of Computer X, Computer Y on the nth benchmark is symmetric around* $\theta = 1$ (corresponding with no location shift from the benchmarks)

- $H_1^+$: *the median performance score ratio* $M_{X/Y}$ *of Computer X, Computer Y on the nth benchmark is not symmetric around* $\theta = 1$

Conduct a Wilcoxon Signed-Rank Test with an $\alpha$ corrected for family-wise error to either reject $H_0^+$ or fail to reject $H_0^+$. For brevity, the procedure is omitted as it is readily available online or in statistics textbooks. However, the procedure is illustrated in detail for equivalence within a margin.

The non-parametric TOST Wilcoxon Signed-Rank Test is utilized for equivalence procedure in [46] with a median ratio [47] $\delta$ chosen *a priori*. Two one-sided tests are conducted to determine if the performance score distributions within the margin $[-\delta, \delta]$ are equivalent. Since the procedures use a ratio performance, the equivalence margin becomes $[1 - \delta, 1 + \delta]$. Both tests must reject the null for equivalence to be established [6]. The upper bound equivalence $1+\delta$ and lower bound equivalence $1-\delta$ signed ranks are computed and tested separately.

The upper bound equivalence, $\delta$, null ($H_{01}^-$) and alternative ($H_{11}^-$) hypotheses are defined as follows:

- $H_{01}^-$: *the performance score ratio distribution* $x_i/y_i$ *on the nth benchmark is greater than or equal to the upper bound equivalence* $1 + \delta$

- $H_{11}^-$: *the performance score ratio distribution* $x_i/y_i$ *on the nth benchmark is less than the upper bound equivalence* $1 + \delta$

The lower bound equivalence, $1 - \delta$, null ($H_{02}^-$) and alternative ($H_{12}^-$) hypotheses are defined as follows:

- $H_{02}^-$: *the performance score ratio* $M_{X/Y}$ *on the nth benchmark is less than or equal to the lower bound equivalence* $1 - \delta$

- $H_{12}^-$: *the performance score ratio* $M_{X/Y}$ *on the nth benchmark is greater than the lower bound equivalence* $1 - \delta$

Let $f_i = (x_i/y_i) - (1+\delta)$ for $i = 1, 2, \ldots, m$ denote the pairwise ratio for the $mth$ observation minus upper bound $1+\delta$ for Computer X, Computer Y on the $nth$ benchmark. Let $\psi_i$ denote the sign indicator of $f_i$ as:

$$\psi_i = \begin{cases} 0, & f_i > 1 \\ -1, & f_i < 1, \end{cases} \tag{8}$$

Rank $R_i$ for $i = 1, 2, \ldots, m$ the absolute values $|f_1|, \ldots, |f_i|$ in ascending order. The product $R_i\phi_i$ denotes the signed rank of $f_i$. The test statistic, $W^-$, for $1+\delta$ is the sum of absolute values of negative ranks defined as:

$$W^- = \sum_{i=1}^{m} R_i\phi_i, i = 1, 2, \ldots, m, \tag{9}$$

where $m$ denotes the number of $m$ benchmark observations

Similarly, let $g_i = (x_i/y_i) - (1-\delta)$ for $i = 1, 2, \ldots, m$ denote the pairwise ratio for the $mth$ observation minus lower bound $1-\delta$ for Computer X, Computer Y on the $nth$ benchmark. Let $\psi_i$ denote the sign indicator of $g_i$ as:

$$\psi_i = \begin{cases} 1, & g_i > 1 \\ 0, & g_i < 1, \end{cases} \tag{10}$$

Rank $R_i$ for $i = 1, 2, \ldots, m$ the absolute values $|g_1|, \ldots, |g_i|$ in ascending order. The product $R_i\phi_i$ denotes the signed rank of $g_i$. The test statistic, $W^+$, for $1-\delta$ is the sum of absolute values of positive ranks defined as:

$$W^+ = \sum_{i=1}^{m} R_i\phi_i, i = 1, 2, \ldots, m \tag{11}$$

where $m$ denotes the number of $m$ benchmark observations

If $(m < 6)$, determine the exact p-value from Wilcoxon Signed-Rank Test tables

for a one-sided test with $\alpha$ separately for both $W^-$ and $W^+$.

If ($m \geq 6$), the rank distribution is approximately normal [48]. Therefore, calculate the z-score as follows:

$$z_1 = \frac{W^- - \frac{M(M+1)}{4}}{\sqrt{\frac{M(M+1)(2M+1)}{24}}}, \tag{12}$$

$$z_2 = \frac{W^+ - \frac{M(M+1)}{4}}{\sqrt{\frac{M(M+1)(2M+1)}{24}}}, \tag{13}$$

Reject $H_{01}^-$ if $z_1 \geq z_{1-\alpha}$ indicating $M_{X/Y}$ is within $\delta$. If the test fails to reject $H_{01}^-$, then a $z_2$ calculation would not occur because equivalence does not hold. Conversely, if $H_{01}^-$ is rejected, then proceed with calculating $z_2$. Finally, reject $H_{02}^-$ if $z_2 \geq z_{1-\alpha}$, indicating $M_{X/Y}$ is within $-\delta$.

The outcomes from the non-parametric TOST of equivalence test and difference test are utilized together to determine a conclusion on the performance comparison between Computer $X$ and Computer $Y$ on the $n$th benchmark. In Table 2 and below, a list of the four conclusions are defined: *trivial difference*, *indeterminant*, *relevant difference*, and *equivalence* [5], [6]:

- **Indeterminant**: fail to reject $H_0^+$ and ($H_{01}^-$ or $H_{02}^-$). Indicating additional benchmark samples are needed for the evaluation.
- **Trivial difference**: reject $H_0^+$ and ($H_{01}^-$ and $H_{02}^-$). Performance distributions were statistically significant but practically irrelevant.
- **Relevant difference**: reject $H_0^+$ but fail to reject ($H_{01}^-$ or $H_{02}^-$). Performance difference on a benchmark that was outside the equivalence margin specified.
- **Equivalence**: fail to reject $H_0^+$ but reject ($H_{01}^-$ and $H_{02}^-$). Performance scores come from the same distribution.

**Table 2. Relevance Conclusions**

| | Difference Test | Equivalence Tests | |
|---|---|---|---|
| **Conclusion** | $\boldsymbol{H_0^+}$ | $\boldsymbol{H_{01}^-}$ | $\boldsymbol{H_{02}^-}$ |
| Indeterminant | Fail to reject | Fail to reject | Fail to reject |
| Trivial Difference | Reject | Reject | Reject |
| Relevant Difference | Reject | Fail to reject | Fail to reject |
| Equivalence | Fail to reject | Reject | Reject |

After all $n$ benchmarks have one of the four relevance testing outcomes presented above, an optional Wilcoxon Signed-Rank Test for a *relevant difference* in overall performance can be conducted depending on the results. In the case that all test's outcomes are *equivalence*, *trivial difference*, or a mixture of both, then the relevance testing is completed. This research provides recommendations further in the text for publishing the results following the optional test procedure.

An optional Wilcoxon Signed-Rank Test for difference can be employed to determine an overall general performance comparison on the benchmark suite. Let $R_i = M_{X/Y}$ for $i = 1, 2, \ldots, n$ denote the median ratio on the $nth$ benchmark. For benchmarks not concluded as *relevant difference*, assign $R_i = 0$; exclude it from the tests and reduce $n$, the number of benchmarks in the sample size, to $n = n - 1$ [17]. The test is excluded because the assumption of continuous variables under the null in a Wilcoxon Signed-Rank Test does not hold. The original family-wise error corrected $\alpha$ calculated prior to the evaluation remains unchanged to account for multiple hypotheses tests. Using the same procedures in the text above, the difference hypotheses for general performance comparison for a one-tailed Wilcoxon Signed-Rank Test are defined as:

- $H_0^+$*: the benchmark suite performance score ratios of Computer X, Computer Y are symmetric around* $\theta = 1$ (corresponding with no location shift from the

benchmark suite)

- $H_1^+$: *the benchmark suite performance score ratios of Computer X, Computer Y are symmetric around theta* $\theta > 1$ *(or* $\theta < 1$*)*

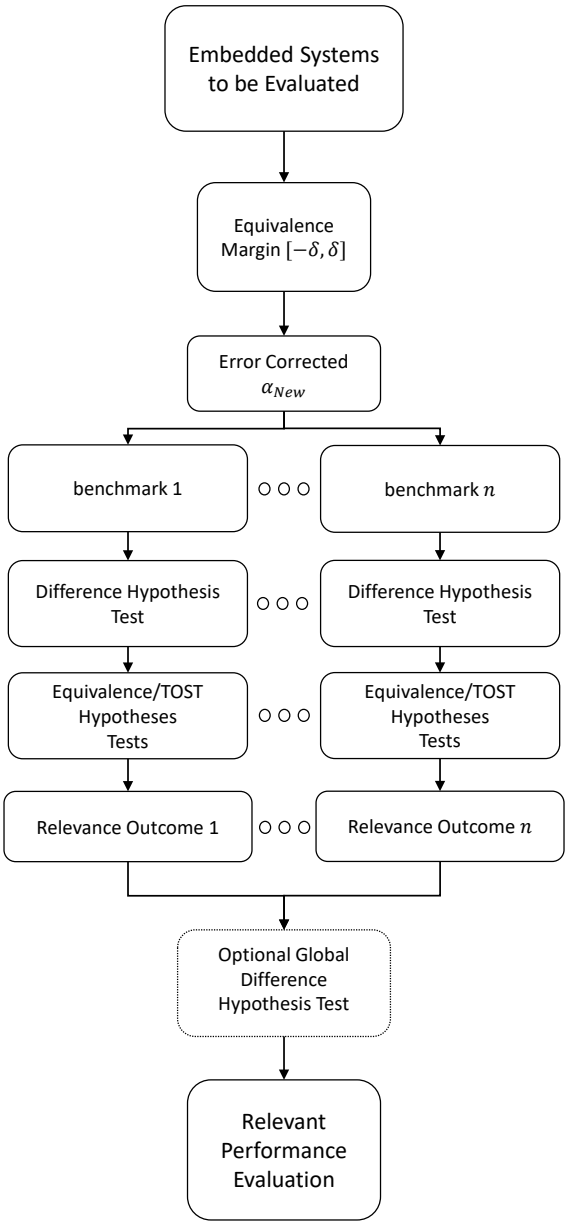



**Figure 7. Relevant statistical performance evaluation framework.**

The SPARC framework provides outcomes that are practical and relevant to the

study or performance comparison under consideration. To clarify, this research outlines the aforementioned procedures with respect to multiple benchmarks in Figure 7. As a final suggestion, it is recommended to write a conclusion that includes the number of tests, outcomes (*indeterminant*, *trivial difference*, *relevant difference*, or *equivalence*), p-values, effect size in terms of location shift, $\alpha$ or confidence level, equivalence margin $[-\delta, \delta]$, and justification for the equivalence margin for performance evaluations.

## 3.4 Modeling Keystone-specific Security Features

In this section, a model to analyze Keystone-specific security features is discussed as a technique for performance characterization. In order to provide a secure execution environment for an embedded system, Keystone has initialization code for setup and other processing requirements that have performance considerations separate from the application performance. This research extends the closed queuing network model listed in Section 2.2.4 by [4] through Keystone-specific functionality to determine the performance impact while executing a benchmark suite. The proposed model was purposely scoped as simple, so that the information gathered provides a foundation to increase complexity by adding additional Keystone-specific security features on successive iterations.

### 3.4.1 Keystone Reference Monitor Concept

A theoretical modeling framework to assess the performance impact of a reference monitor was proposed in [4] and functionally attributed to a simple closed-queue network model. The model is illustrated by referencing Keystone's API through its use of secure enclaves and trusted/untrusted OS configuration. The overhead cost of using Keystone is mapped to a network queue response time; the time it takes for

a program to begin execution while Keystone is enforcing the security policy versus the time to execute without Keystone. Additionally, processing delays commonly identified in queuing theory models are defined as delays encountered due to Keystone context switching (i.e. switch between other processes) while a program is executing.

The original reference monitor concept in Figure 1 listed two primary functions: an ORM, and a SRM [4]. This research models Keystone in a similar concept in Figure 8, but with clearer definitions based on its operation. The figure depicts an untrusted OS, Linux, that is currently activating an enclave trusted OS for a secure execution environment. Not shown in the figure are the steps to initialize Keystone, the untrusted OS, and the allocation of resources; those are executed during system bootup hence the allocated resources.

The ORMr was redefined as the Resource Monitor (RM). The RM ensures that the system resources accessed by Untrusted and Trusted Entities are secure per their privilege level and PMP configurations.

The SRM was redefined as the Entity Monitor (EM). The EM is tasked with initializing and securing the Trusted Entities that are activated within the system and monitoring any unauthorized usage of them. Activation of any Trusted or Untrusted entity is also monitored by the EM.

### 3.4.2 Proposed Keystone Closed Queuing Network Model

The analysis used in [4], and the operational approach for closed queuing models from [49], is also used here. The concept was transposed to a functionally equivalent closed queuing network in Figure 9. The primary difference in this model from [4] is the activation of the Trusted Entity $E_{RT}$; this is the enclave created by Keystone for a secure execution environment and has its own trusted OS (also called a runtime).

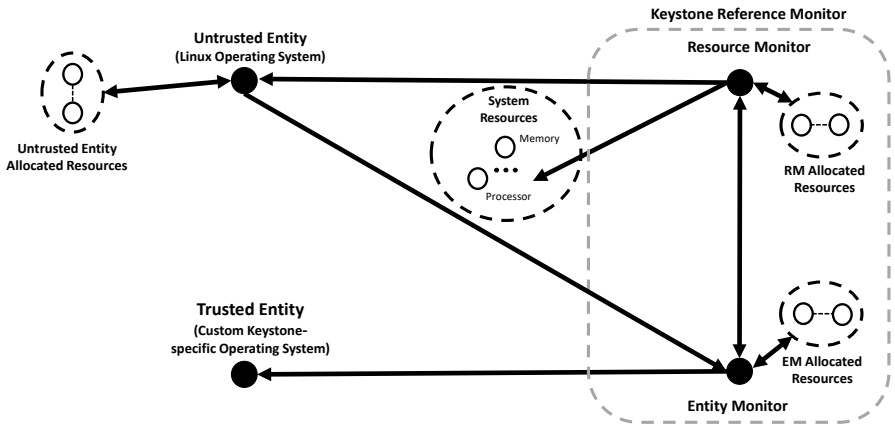



**Figure 8. Redefined concept model for Keystone reference monitor derived from Figure 1.**

### 3.4.3 Reference Monitor Job Request Generation

Access requests by $E_{OS}$ to create an enclave (and securely execute a program) generate job requests to the RM or EM. The Keystone API lists functions that are available to $E_{OS}$ and $E_{RT}$, a subset of those applicable to this framework is listed in Table 3.

In order to simplify the model analysis, a job is loosely defined in this framework as a process needed to complete a general objective. Jobs could be further broken down into specific function calls or mapped to source code, but this is left to future work.

**Table 3. Subset of Keystone API functions [1]**

| Caller | Function | Description |
|---|---|---|
| $E_{OS}$ | create enclave | Allocate and hash enclave |
| | run enclave | Start enclave and $E_{RT}$ |
| | resume enclave | Continue $E_{RT}$ operation |
| | destroy enclave | Destroy $E_{RT}$ and free enclave resources |
| $E_{RT}$ | stop enclave | Pause $E_{RT}$ |
| | exit enclave | Terminate $E_{RT}$ |

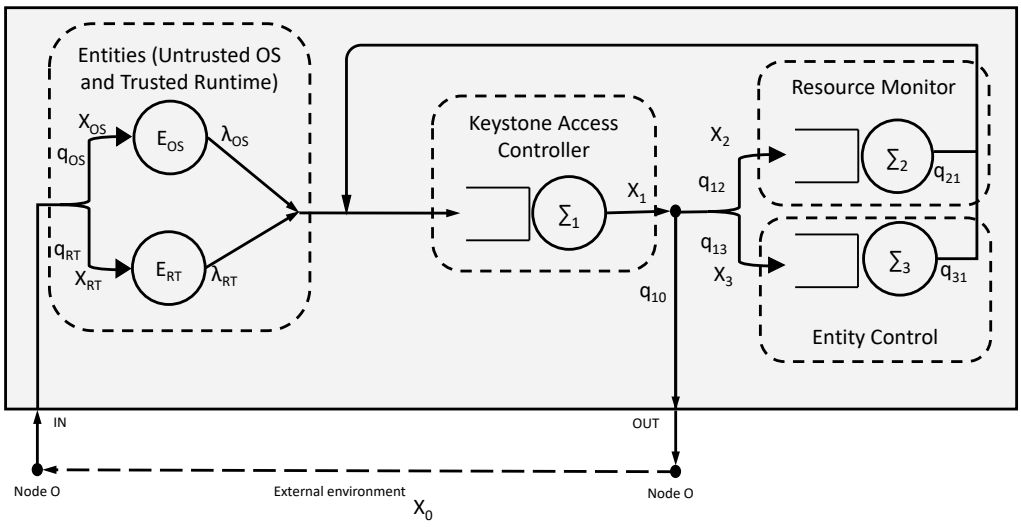



**Figure 9. Functional closed queuing network model from Gorbachov et al. (2019) modified for Keystone reference monitor.**

To start an application in an enclave, the $E_{OS}$ sends a create enclave request to the Keystone access controller, $\Sigma_1$. Assuming valid, the create enclave request generates two requests: an allocate request to $\Sigma_2$ and an initialize request to $\Sigma_3$. After the two requests complete, $\Sigma_1$ generates two additional requests: set initial PMP configuration to $\Sigma_2$ and validate/hash enclave to $\Sigma_3$. At this point the secure enclave is validated; the program and $E_{RT}$ has been loaded and initialized. $\Sigma_1$ then sends the notification to $E_{OS}$ that the enclave is ready for secure execution.

Next, $E_{OS}$ sends a run enclave job to $\Sigma_1$ which generates two requests: set PMP memory isolation to $\Sigma_2$ and transfer processor control to $E_{RT}$ to $\Sigma_3$. Upon request completion, $\Sigma_1$ notifies $E_{RT}$ that it has processor control and to start executing the program. After the program has completed, $E_{RT}$ will request to exit enclave and $E_{OS}$ will request to destroy enclave to free up resources.

## 3.5 Summary

This chapter described two models for characterizing the performance of an application on a RISC-V embedded system with Keystone enabled versus disabled. It presented architectural layer models to visualize both sub-systems and depicted multiple functions that can influence or impact performance. A new framework, SPARC, was introduced to characterize the performance of an embedded system through benchmarks. Finally, for systems with Keystone enabled, a method for modeling the Keystone-specific security functions was illustrated.

# IV. Experimental Design and Methodology

## 4.1 Objective

This research aims to characterize the performance of three RISC-V embedded systems and determine the performance impact to a system with a reference monitor, Keystone, enabled. It also seeks to establish the efficacy of the SPARC framework as a statistically rigorous performance evaluation in relation the HPT framework. The experiment design and methodology presented in this section is a demonstration of the SPARC framework procedures with an evaluation of three RISC-V processors. After a baseline performance characterization is performed consisting of all three RISC-V embedded systems with Keystone disabled, SPARC is further utilized to evaluate the performance impact of Keystone enabled on a per-core basis. In other words, on one RISC-V system the performance evaluation is between the benchmark performance with Keystone disabled versus enabled. Furthermore, the performance impact of Keystone-specific security features are evaluated through a closed-queuing network model. Specifically, the experimentation seeks to accomplish four objectives:

1. Assess the efficacy of the SPARC framework for performance evaluations of embedded systems.
2. Evaluate margins of equivalence in performance evaluations for improved statistical inference.
3. Measure the per-core performance impact of Keystone.
4. Model Keystone-specific security features on a RISC-V embedded system.

Results of the experimentation provide a performance characterization of the three RISC-V embedded systems with and without Keystone, as well as an assessment of the SPARC framework.

## 4.2 System Under Test

Figure 10 presents the SUT and component under test (CUT) diagram. The test assumptions, hardware configuration, experimental design, and performance metric measurements are discussed prior to the specifics of each evaluation.

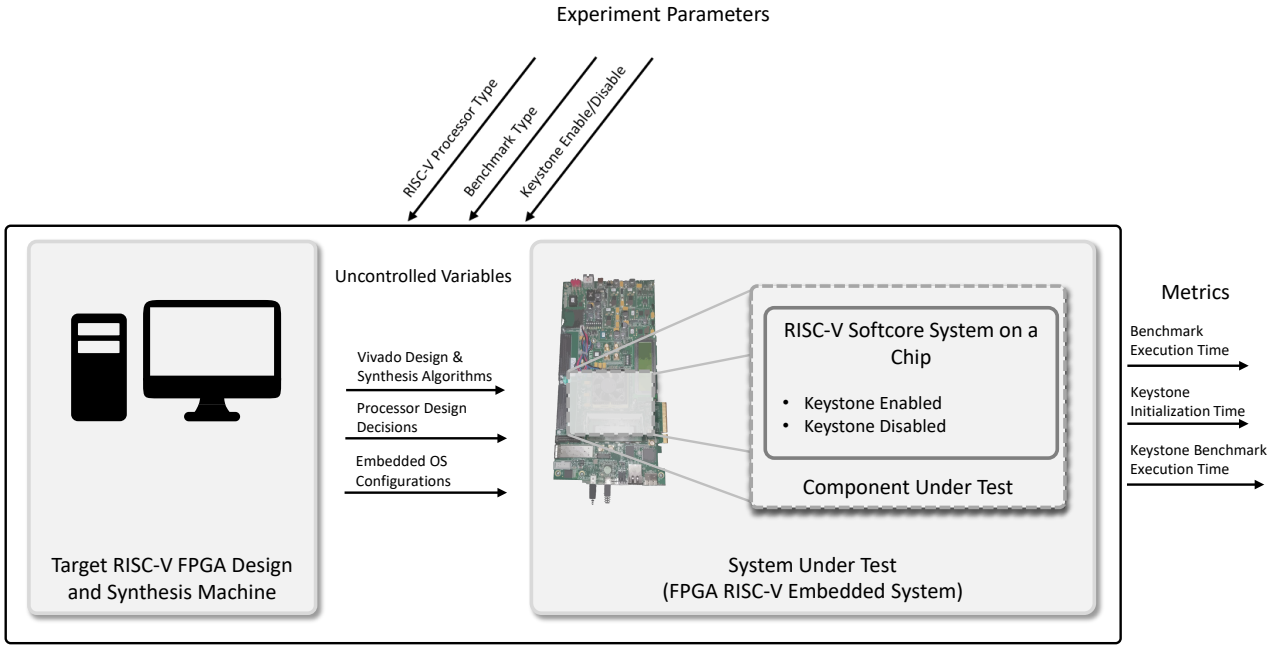



**Figure 10. System under test and component under test diagram.**

## 4.3 Assumptions

The assumptions made in the experiment are related to the instantiated embedded system on the FPGA. There are additional assumptions with respect to statistical analyses that are provided in the following chapter. The following assumptions are declared with respect to the experimental design and factors affecting it:

- The performance of the softcore RISC-V embedded systems are representative of their ASIC counterparts.

- The FPGA bitstream generation by Vivado is accomplished through a multiple stage process that uses algorithms to instantiate the components. The impacts to performance by Vivado processes are not evaluated in this research.
- Each RISC-V embedded system has a different HDL code compiler prior to bitstream generation that will not affect performance of the final system.
- The RISC-V processors, Rocket and Ariane, had existing FPGA build configurations available for use on the experiment FPGA hardware. Shakti was customized by adding peripherals present on Rocket or Ariane, but absent from the Shakti build and did not affect the datapath of the processor.
- The Linux OS configurations and hardware drivers are common across the three embedded systems except for required build-specific drivers and do not affect performance.
- Differences in how each embedded system loaded the Linux OS and benchmarks have no noticeable effect on performance. Rocket and Ariane loaded software into memory from a secure digital memory card, whereas Shakti required a debugging application to load directly into memory.
- For experiment runs with Keystone enabled, a run consists of one secure enclave created and one benchmark execution. Performance of multiple enclaves are not modeled.
- The security of Keystone or its security mechanisms are not assessed. In other words, the security of Keystone's enclave encryption is not relevant, but its performance is.
- The Keystone software code and module is built into the system bootloader and therefore persistent, but not active, across runs that test the performance

of a benchmark with Keystone disabled. Keystone-specific features are enabled and disabled by activating a software driver within the Linux OS. The software code and modules do not affect benchmark performance in experiment runs with Keystone disabled.

- All planned runs of a benchmark for a given system are executed successively and deemed independent.

## 4.4 Control Variables

The control variables are those held constant over the execution of benchmarks on the RISC-V systems. They primarily exist within the RISC-V embedded system configuration and FPGA hardware, and are listed in Table 4.

**Table 4. Control Variables**

| Variable | Value | Description |
|---|---|---|
| FPGA Hardware | Xilinx Virtex-7 XC7VX485T | FPGA model loaded with the RISC-V embedded system |
| Processor Instruction Execution | In-order | Instructions are fetched and executed in order (versus out-of-order) |
| Width (bits) | 64 | Address width of the processors |
| Clock Rate (MHz) | 50 | Clock speed of processors instantiated on the FPGA |
| L1 Inst Cache (KB) | 16 | Size of instruction cache memory |
| L1 Data Cache (KB) | 32 | Size of data cache memory |
| DDR3 Size (GB) | 1 | DDR3 synchronous dynamic random-access memory size (physical component located on FPGA development board) |

## 4.5 Independent Variables

Independent variables are specific to the two experiments: embedded system with Keystone enabled, and with Keystone disabled. These are listed and described in Table 5.

**Table 5. Independent Variables**

| Variable | Value | Description |
|---|---|---|
| Keystone | Enabled/Disable | Software switch that enables or disables Keystone functionality |
| Benchmark Type | $1, 2, \ldots, 8$ | Selection between 8 benchmarks |
| RISC-V Processor | Rocket/Ariane/Shakti | Selection of three RISC-V processor types |
| TBD | - | TBD |

## 4.6 Response Variables

The experiment response variables are listed in Table 6. These are dependent variables measured while varying the independent variables between runs. The Benchmark Execution Time, Median Benchmark Execution Time, Speedup Ratio, and Median Speedup Ratio are key attributes for analysis with the SPARC framework. The Keystone Initialization Time and Total Keystone Time provide insight into the performance impact of Keystone-specific security mechanisms.

## 4.7 Uncontrolled Variables

This research employs an FPGA to conduct the experiment with three RISC-V embedded system designs and reduces the risk of bias that could occur if ASICs were used. While the FPGA excels in this regard, there are uncontrolled variables that could impact performance, but their magnitude remains unknown.

**Table 6. Response Variables**

| Variable | Units | Description |
|---|---|---|
| Benchmark Execution Time | seconds | Time for benchmark to complete one run execution |
| Keystone Benchmark Execution Time | seconds | Time for benchmark to execute and complete within a Keystone secure enclave |
| Keystone Initialization Time | seconds | Time for Keystone-specific features to initialize prior to benchmark execution |
| Median Benchmark Execution Time | seconds | Median Benchmark Execution Time |
| Speedup Ratio | - | Ratio of two RISC-V processor's Benchmark Execution Time |
| Median Speedup Ratio | - | Median Ratio of two RISC-V processor's Benchmark Execution Time |
| Total Keystone Time | seconds | Sum of Benchmark Execution Time and Keystone Initialization Time if Keystone is enabled |

As stated in Section. 4.3, the FPGA software (i.e. Vivado) generates the bitstream file from HDL code that constitutes each RISC-V embedded system and subsequently loads the bitstream onto the FPGA for configuration. Vivado accomplishes this process through stages of algorithms, after which the embedded system behavior is verified for timing constraints. The algorithms can be customized to ensure verification is successful; they can be executed over successive iterations to improve timing, or continuously executed for extended periods of time. The Vivado settings that accompany each RISC-V embedded system, chosen by the system designers, were not altered for this experiment and therefore an uncontrolled variable.

Similarly, the decisions made by the RISC-V system designers for implementing their respective processor design are left as-is. The purpose of this research is not to analyze why a particular memory design in one processor is better than another,

although it is a possible avenue for follow-on research. Rather, it provides the foundation of performance characterization from which design analysis can occur. Nevertheless, these decisions do affect performance, either positively or negatively, but are purposefully uncontrolled in this experiment.

Finally, the Linux OS configuration is an uncontrolled variable. Each system has subtle differences in the drivers loaded through the configuration file. OS configurations and drivers are often specific to an embedded system; replicating a default configuration to use across systems can lead to errors or undesirable operation.

## 4.8 Experimental Design

There are four objectives that the experiment must satisfy, previously listed in Section 4.1. The following section documents the experimental design and suitability towards meeting research objectives.

### 4.8.1 Experimental Hardware Setup

The experiment consists of three RISC-V softcore processors, Rocket [50], Shakti [51], and Ariane [52], instantiated on an FPGA and the benchmark suite, RV8. RV8 contains eight benchmarks that are executed from which the output is measured, their descriptions were previously listed in Section 2.2.2 in Table 1. Each processor has its own system-on-a-chip design integrated within the build that includes, but is not limited to: L1 and L2 cache, DDR3 memory controller, and universal asynchronous receiver-transmitter (UART). The processors operate at 50 MHZ clock rate and the system configurations (i.e. control variables) were previously listed in Table 4.

### 4.8.2 Experiment Methodology

The procedure for experiment execution consisted of the following steps:

1. Download the Vivado-generated bitstream for one of the three RISC-V embedded systems onto the FPGA.
2. Compile the software: the Linux OS, version 5.3, and 8 benchmarks.
3. Load the software onto either a secure disk memory card (Rocket, Ariane), or prepare to load software directly to the FPGA DDR3 memory (Shakti).
4. Power-on the FPGA.
5. Software is automatically (Rocket, Ariane) or manually (Shakti) loaded into DDR3 memory and will begin the Linux OS boot sequence.
6. The Keystone driver is manually loaded, but Keystone is not enabled.
7. For each benchmark, a script within the Linux OS batch executes the benchmark 30 times with Keystone disabled to capture performance metrics (see Appendix A). Afterwards, the script enables Keystone and batch executes the benchmark an additional 30 times.
8. The procedure is complete for one RISC-V embedded system after the script has executed all 8 benchmarks and performance metrics are captured.
9. The experiment loops back to the first step and continues with the next system until all three are complete.

In the experiment, a sample size of 30 for each benchmark was chosen to examine suitability of its distribution for applying the CLT in the analysis next chapter.

### 4.8.3 Tools

The tools in Table 7 were used to synthesize FPGA designs, load the FPGA bitstream, conduct statistical tests, analyze performance metrics, and plot graphs.

**Table 7. Data Gathering and Analysis Tools.**

| Name | Version | Description |
|---|---|---|
| R | v4.0.3 | Programming language and free extensible software environment used for statistical analysis. |
| JMP | v15 | Commercial statistical analysis software. |
| Vivado | v2018.3 | Software used to generate bitstream for the FPGA design. |

After the performance metrics are captured, the statistical software R [53] was used to conduct hypothesis tests related to SPARC. R provides a software environment that can be customized by installing user-created packages for a variety of statistical tests or graphing functions. Specifically, a statistical package called "stats" contains functions to perform Wilcoxon Signed-Rank difference tests used in the analyses next chapter. The package did not have a function to perform an equivalence test with a Wilcoxon Signed-Rank Test, therefore this research modified the code from difference to equivalence (see Appendix B). Two script files were used within R: a script file to perform statistical analyses and a script file to plot graphs. The R packages used in this research and their descriptions are listed in Table 8.

## 4.9 Methodology: SPARC Framework Specifics

The data collected in the experiment are used with SPARC for two analyses: characterizing the performance through a comparison of three RISC-V systems with Keystone disabled, and characterizing the individual RISC-V system performance impact with Keystone enabled. The SPARC specifics to each analysis are discussed

**Table 8. R software packages used and their description.**

| R Package Name | Description |
|---|---|
| stats [53] | Statistics package with hypothesis tests. |
| ggplot2 [54] | Graphical plotting software for data analysis. |
| svglite [55] | Tool for producing scalable-vector graphics. |
| reshape2 [56] | Tool for reshaping data columns and rows. |
| tidyverse [57] | Collection of analysis packages. |
| dplyr [58] | Tool that standardizes data manipulation within R. |
| tidyr [59] | Tool for cleaning up and organizing data within R. |
| shades [60] | Color manipulation for plots. |
| rcolorbrewer [61] | Collection of color palettes used in plots. |
| ggthemes [62] | Collection of themes for ggplot2 package. |
| scales [63] | Adds functions to scale data for plots. |

in the following text. The methodology used to measure performance metrics specific here is discussed at the conclusion of this section.

### 4.9.1 Characterizing Three RISC-V Embedded Systems with Keystone Disabled

There are two pairwise comparisons conducted in this analysis, Rocket to Ariane and Rocket to Shakti. A third comparison of Ariane to Shakti could have been performed, but it would have required further FWER correction to $\alpha$. As discussed in Section 3.3.2.2, in a hypothesis test $\alpha$ is compared to the test's p-value to reject or fail to reject the null hypothesis. Prior to the analysis, $\alpha$ is corrected based on the FWER which depends on the number of planned hypothesis tests.

In parametric statistics, there are sample size estimates that can be calculated to ensure an error-corrected $\alpha$ test has the range to show statistically significant results. But, as mentioned in Section 3.3.2.3, sample-size estimation for non-parametric statistics is often not possible. Therefore, it is frequently not possible to verify an error-corrected $\alpha$ and the statistical strategy is to limit the number of planned hypothesis tests in analysis. Planning an abundance of

hypothesis tests without this consideration risks forfeiting the experiment and starting it over.

For the two primary RISC-V evaluations of Rocket to Ariane and Rocket to Shakti, there are a total of 34 hypotheses tests $2(8 + 8 + 1)$ conducted. For the pairwise comparison Rocket to Ariane, 8 difference hypotheses tests are conducted for location shifts plus 8 equivalence hypotheses tests and an additional test for the overall analysis. The tests are repeated for the second pairwise comparison of Rocket to Shakti. Therefore, the overall evaluation error correction is set at $\alpha = 0.05$ which translates to a $FWER \leq 0.82518$ using (3). This research uses the Bonferroni Correction method (4) to control the $FWER$ but still allow $(1 - \alpha/m)$ confidence intervals calculated. The error corrected $\alpha_{New} = 0.0014706$, which is compared to each benchmark test $p_i$, to reject $H_0$ or fail to reject $H_0$.

To show correct application of the SPARC framework equivalence tests and conclusions, this research defines three $[1 - \delta, 1 + \delta]$ equivalence margins for analysis. Specifically, the primary equivalence margin is denoted as $\delta = 0.05$ and $[0.95, 1.05]$ for the RISC-V system performance comparison. The bounds are small enough to highlight a minimal difference in speedup between two system's benchmark performance, while providing margin for equivalence due to high precision. The other two equivalence margins are hypothetical and discussed in the next section.

### 4.9.2 Evaluating Hypothetical Margins of Equivalence

To evaluate margins of equivalency with SPARC, this work also considers $\delta_{Hyp1} = 0.25$ and $\delta_{Hyp2} = 0.50$ as the hypothetical extreme evaluations with equivalence margins $[0.75, 1.25]$, $[0.50, 1.50]$ to demonstrate the framework in conditions of similarly performing systems.

For the evaluations $[0.75, 1.25]$ and $[0.50, 1.50]$, this work considers them as post-

hoc tests after the primary comparison to prevent inflating the error corrected $\alpha_{New}$. $\alpha_{New}$ is adjusted by summing the additional 16 equivalence plus 2 overall analysis tests for both pairwise comparisons. This amounts to a total of $34 + 2 \cdot (8 + 8 + 2) = 70$ hypotheses tests and a Bonferroni Corrected $\alpha_{New} = 0.0007143$ for the hypothetical evaluations.

### 4.9.3 Characterizing Individual RISC-V Embedded Systems with Keystone Enabled

Evaluating the performance impact of Keystone on individual RISC-V systems Rocket, Ariane, and Shakti, is similar in methodology to Section 4.9.1. Instead of comparing different RISC-V systems to another, this assessment compares the same system with Keystone enabled and disabled. For example, the benchmark's output of the Rocket system with Keystone enabled is evaluated against the benchmark's output with Keystone disabled.

The $\delta$ remains at 0.05 speedup with an equivalence margin of $[0.95, 1.05]$. One primary difference from the baseline characterization is the FWER and $\alpha_{New}$. In this evaluation, the comparison tests are considered in separate families of data which reduces the number of hypotheses tests affecting the error rate. There are a total of 17 hypotheses tests $(8 + 8 + 1)$ conducted, which sets the $FWER \leq 0.00294$ using Equation 3, and will determine whether to reject $H_0$ or fail to reject $H_0$.

### 4.9.4 Performance Metric Measurements

Within each benchmark, code was inserted to capture the clock cycles with inline assembly through a RISC-V specific pseudo-instruction, *rdcycle* [9]. The code executes at program start and program completion to calculate the number of clock cycles consumed. That result it then divided by the clock rate to derive the

execution time $L$ as follows:

$$L = \frac{Cycles_{End} - Cycles_{Start}}{Clock_{Rate}}, \tag{14}$$

$L$ is used to calculate the Speedup Ratio $S$ between pairwise comparisons defined as:

$$S = \frac{L_A}{L_B}, \tag{15}$$

The Speedup Ratio and Median Speedup Ratio are used to abstract out units of time and identify performance shifts that occur between the embedded systems.

### 4.10 Methodology: Keystone-specific Security Features

An additional experiment was conducted to model Keystone-specific security features on the Rocket RISC-V embedded system. The experiment used the Rocket system with the configuration listed in Table 4, except for operating at a clock rate of 100 MHz rather than 50 MHz.

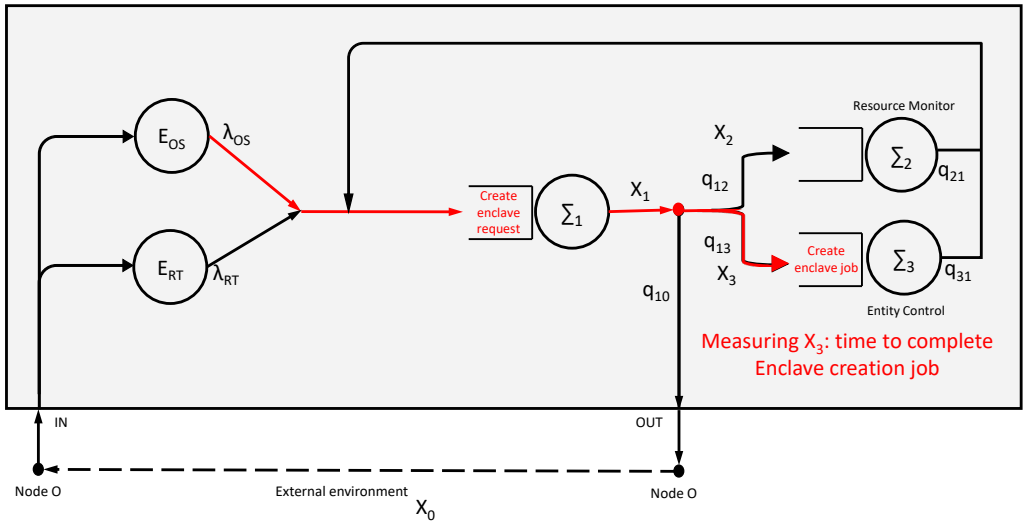



**Figure 11. Functional closed queuing network model for Keystone reference monitor.**

The proposed functional closed queuing network model is illustrated in Figure 11. This research is only concerned with calculating the average system response time for Keystone to initialize and prepare to run a benchmark in a secure enclave.

The analysis methodology is to measure the steps required to instantiate $E_{RT}$, which is defined in Table 6 as the Keystone Initialization Time. By measuring the time for $\Sigma_3$ to complete one enclave creation job, the initialization impact to the overall average service time of the system can be determined.

The operational analysis of a closed queuing network by [49] provides a simple model to measure the average system response time $R$ based on the system output rate, $X_0$. In [4], the flow balance equations for the system are:

$$X_0 = (q_{10}) \cdot (X_1)$$

$$X_1 = X_0 + X_2 + X_3$$

$$X_2 = (q_{12}) \cdot (X_1)$$

$$X_3 = (q_{13}) \cdot (X_1)$$

The throughput, $X_3$, of $\Sigma_3$ is the output rate in jobs per second which complete at the device. This alters the system of flow balance equation to find throughput of $X_1$:

$$X_1 = (q_{13}) \cdot (X_3) \tag{16}$$

The throughput $X_1$ is calculated to solve for the remaining unknown variables and determine the average system response time. The average system response time $R$ is calculated using the Interactive Response Time Formula [4]:

$$R = \frac{N}{X_0} - \frac{1}{\lambda} \tag{17}$$

The calculation above for $R$ is the performance impact to the system that occurs if Keystone is enabled, excluding the execution time for the benchmark.

### 4.10.1 Keystone Enabled Performance Metric Measurement

The metrics captured are based on the time to initialize an enclave and execute a benchmark with $E_{RT}$. From the previous section, a measurement of any $X_i$ is required to solve the system of equations.

To run the benchmarks with Keystone enabled, the batch scripted discussed in Section 4.8.2, executes a specific line of code that enables Keystone (see Appendix A). There are three binary executable inputs to the line of code required to run a benchmark (or application) within a secure enclave. The three are the host, trusted OS, and the application. The host executable initializes the enclave using parameters, such as the size of the enclave, and launches the enclave [10]. The trusted OS handles any system-level functionality that is required between the enclave benchmark and the untrusted Linux OS. Finally, the benchmark is the user-level application that the trusted OS executes inside the secure enclave.

Within the host executable file, additional code was added to collect the number of clock cycles with inline assembly similar to Section 4.9.4. The number of clock cycles were collected prior to and after a sequence of Keystone initialization code, to determine the number of clock cycles required.

In this scenario, the code collecting the clock cycles is represented by throughput

variable $X_3$ illustrated in Fig 11. Collection starts with $E_{OS}$ sending a create enclave job, which occurs within the host executable code, and arrives at $\Sigma_3$. The cycles are collected again after the job completes, in this case after the Keystone initialization sequence has completed. The Keystone Initialization Time metric is derived from the number of clock cycles elapsed and the speed of the processor using Equation 14.

## 4.11 Methodology Summary

This chapter described the experimental methodology used for measuring the performance metrics, establish a sound experimental design, and illustrate the procedures supporting the embedded system evaluations. It presented methodology specific to the SPARC framework to establish a baseline performance characterization of three RISC-V embedded systems along with the individual RISC-V system performance impact of Keystone. Further, it described the approach for modeling Keystone-specific security features in a closed-queue network and the procedure for calculating the average system response time for a system with Keystone enabled.

# V. Observations and Analysis

## 5.1 Overview

This chapter presents the observations, results, and analysis from the experimental design described in Chapter IV. Discussions are separated into the four following evaluations: (i) using SPARC to analyze three RISC-V embedded systems with Keystone disabled, (ii) assessing SPARC with hypothetical margins of equivalence, (iii) using SPARC to analyze individual RISC-V systems with Keystone enabled, and (iv) modeling Keystone-specific security features. Issues found while modeling Keystone-specific security features are discussed and the follow-on experimentation with additional analysis is described. Significant findings, performance metrics, and statistical analyses are presented; specific to each evaluation.

## 5.2 RISC-V Performance Evaluations with the SPARC Framework

The results presented within this section characterize three RISC-V embedded systems with Keystone disabled using the new SPARC methodology, beginning with evaluation of Rocket to Ariane. Figure 12 presents quantile-quantile plots for each benchmark with data points as paired-observation differences against a theoretical normal distribution line. Visually, the plots for AES, Bigint, Norx, and Primes indicate non-normal distributions not suitable for parametric tests. The sharp curved data points around the normal line on AES and Norx are due to heavy tails and the large gap in data points on Bigint and Primes suggest bimodal distributions.

Shapiro-Wilk Tests were conducted for normality to affirm the visual analysis [64]. The result of these tests are listed in Table 9. Each benchmark distribution was tested

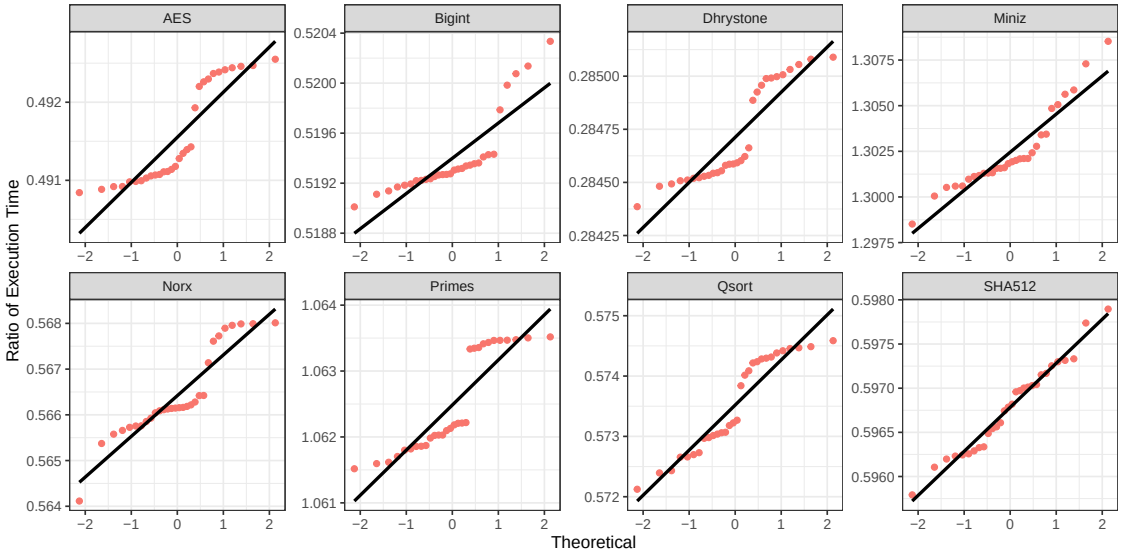



**Figure 12. Rocket with Ariane quantile-quantile plots for each benchmark. Data points are a difference, Rocket − Ariane, compared to a theoretical normal distribution line.**

against the Shapiro-Wilk $H_0$ that the distribution is normal; rejecting $H_0$ signifies the distribution is not normal. Dhrystone and SHA512 are the only two normal distributions of the Rocket to Ariane evaluation and therefore the CLT cannot be relied on, despite a larger sample size.

**Table 9. Shapiro-Wilk Tests for normality**

| Sub-benchmark | Rocket to Ariane | | | Rocket to Shakti | | |
|---|---|---|---|---|---|---|
| | ***W*** | ***p*** | **Rej $H_0^+$?** | ***W*** | ***p*** | **Rej $H_0^+$?** |
| AES | 0.84612 | 5.148e-4 | Yes | 0.90514 | 1.124e-2 | Yes |
| Bigint | 0.81796 | 1.418e-4 | Yes | 0.93125 | 5.302e-2 | No |
| Dhrystone | 0.94087 | 9.603e-2 | No | 0.26703 | 3.76e-11 | Yes |
| Miniz | 0.86087 | 1.056e-3 | Yes | 0.69106 | 1.16e-6 | Yes |
| Norx | 0.8703 | 1.701e-3 | Yes | 0.50051 | 5.425e-9 | Yes |
| Primes | 0.79085 | 4.486e-5 | Yes | 0.77818 | 2.69e-5 | Yes |
| Qsort | 0.92954 | 4.774e-2 | Yes | 0.73149 | 4.637e-6 | Yes |
| SHA512 | 0.96135 | 0.3353 | No | 0.93825 | 8.163e-2 | No |

**Table 10. SPARC framework results for difference and equivalence at $[0.95, 1.05]$ in Rocket to Ariane comparison tests**

| Benchmark | $M_X$ (sec) | $M_Y$ (sec) | $M_{X/Y}$ | $H_0^+$ | | $H_{01}^-$ | | $H_{02}^-$ | | Rej $H_0^+$ ? | Rej $H_0^-$ ? | Relevance |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $z$ | $p$ | $z_1$ | $p_1$ | $z_2$ | $p_2$ | | | |
| AES | 129.5164 | 263.6765 | 0.4912 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Bigint | 203.4705 | 391.8052 | 0.5193 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Dhrystone | 75.6602 | 265.8570 | 0.2846 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Miniz | 505.5659 | 388.3212 | 1.3019 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | -4.792 | 0.992 | Yes | No | Rel diff |
| Norx | 73.6444 | 130.0691 | 0.5662 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Primes | 257.0185 | 242.0116 | 1.0622 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | -4.792 | 0.992 | Yes | No | Rel diff |
| Qsort | 129.927 | 242.0116 | 0.5732 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| SHA512 | 81.0285 | 135.7358 | 0.5968 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |

Proceeding with the new relevance framework, difference and equivalence hypotheses tests were conducted on the speedup ratio of medians and the results are listed in Table 10. The ratio of median execution time $M_X/M_Y$ for Rocket ($M_X$) to Ariane ($M_Y$) indicates that Ariane is the faster processor if the ratio is greater than 1. Ideally, if the processors were equal in median execution time, the ratio would be exactly 1. Figure 13 presents a bar graph plotting the median execution times listed in Table 10 of Rocket and Ariane within each benchmark.

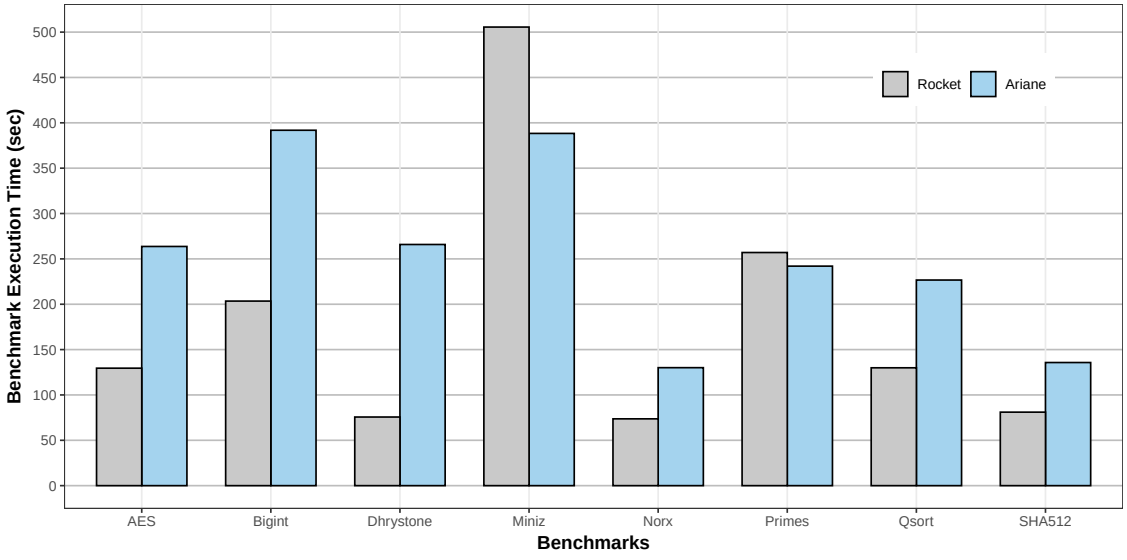



**Figure 13. Median Rocket and median Ariane bar graph for each benchmark.**

Each benchmark difference test rejected $H_0^+$, indicating that the performance score distributions are symmetric around $\theta = 1$. In other words, there was a speedup or slowdown distribution location shift of the median time. Further, the tests of equivalence at $\delta = 0.05$ also rejected $H_0^-$ in all benchmarks. The hypothesis test results, together with the four possible relevance choices from Section 3.3.4, allow us to conclude that there is a *relevant difference* in median performance between the Rocket and Ariane RISC-V processors in all 8 benchmarks. It was recommended previously in the text that the effect sizes should be listed, either in the evaluation conclusion, or as they are listed in Table 10. Rocket had a *relevant difference* of faster median execution times over Ariane in 6 of the 8 benchmarks.

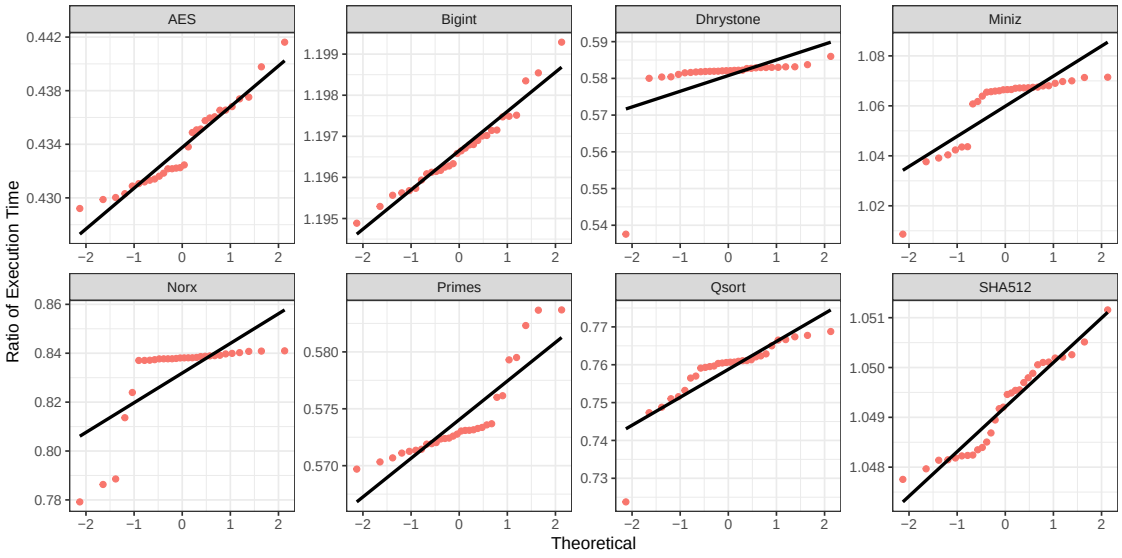



**Figure 14. Rocket with Shakti quantile-quantile plots for each benchmark. Data points are a difference, Rocket − Shakti, compared to a theoretical normal distribution line.**

In the performance evaluation of Rocket to Shakti, quantile-quantile plots are presented in Figure 14. The plots show non-normal distributions in all benchmarks except for Bigint and SHA512. In contrast to quantile-quantile plots in Figure 12, the distributions of Dhrystone, Norx, Miniz, and Qsort are highly skewed left and include

heavy tails. The heavy tail in Dhrystone is caused by an outlier data point at $-65$ seconds. Similarly, outlier data points in Norx result in a heavy tail distribution and signify parametric tests could be adversely affected. Results from Shapiro-Wilk Tests for normality listed in Table 9 confirm that all benchmarks are non-normal except for Bigint and SHA512.

Alternatively, a different statistical analysis could be employed by examining the outlying data points to determine if they can be removed and then testing for normality again. This would require altering the $\alpha$ correction again, accounting for the additional hypotheses tests, and also adding justification for outlier data point removal. But if the process was successful and produced normal distributions, then parametric statistical tests could have been performed. This research refrained from employing this technique because of the extensive time and experience required to distinguish between data points that are outliers versus data points that indicate a problem with the experimental design. Instead, SPARC was designed to test population medians under the assumption that outlier data points are not removed.

Instead of removing any outlier data points, this research presents the results from the difference and equivalence hypotheses tests performed on the median speedup ratio of execution times for Rocket ($M_X$) and Shakti ($M_Y$) in Table 11. The bar graph in Figure 15 plots median execution times for Rocket and Shakti within each benchmark comparison.

**Table 11. Rocket to Shakti comparison tests for difference and equivalence at** $[0.95, 1.05]$

| | | | | $H_0^+$ | | $H_{01}^-$ | | $H_{02}^-$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **Benchmark** | $M_X$ **(sec)** | $M_Y$ **(sec)** | $M_{X/Y}$ | $z$ | $p$ | $z_1$ | $p_1$ | $z_2$ | $p_2$ | **Rej** $H_0^+$ **?** | **Rej** $H_0^-$ **?** | **Relevance** |
| AES | 129.5164 | 299.9041 | 0.4324 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Bigint | 203.4705 | 170.0411 | 1.1966 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | -4.792 | 0.992 | Yes | No | Rel diff |
| Dhrystone | 75.6602 | 129.9888 | 0.5821 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Miniz | 505.5659 | 473.9084 | 1.0665 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | -3.682 | 0.998842 | Yes | No | Rel diff |
| Norx | 73.6444 | 87.8986 | 0.8381 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Primes | 257.0185 | 448.9452 | 0.5729 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Qsort | 129.927 | 170.825 | 0.7606 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| SHA512 | 81.0285 | 77.2075 | 1.0493 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 3.599 | 1.594e-4 | Yes | Yes | Triv diff |

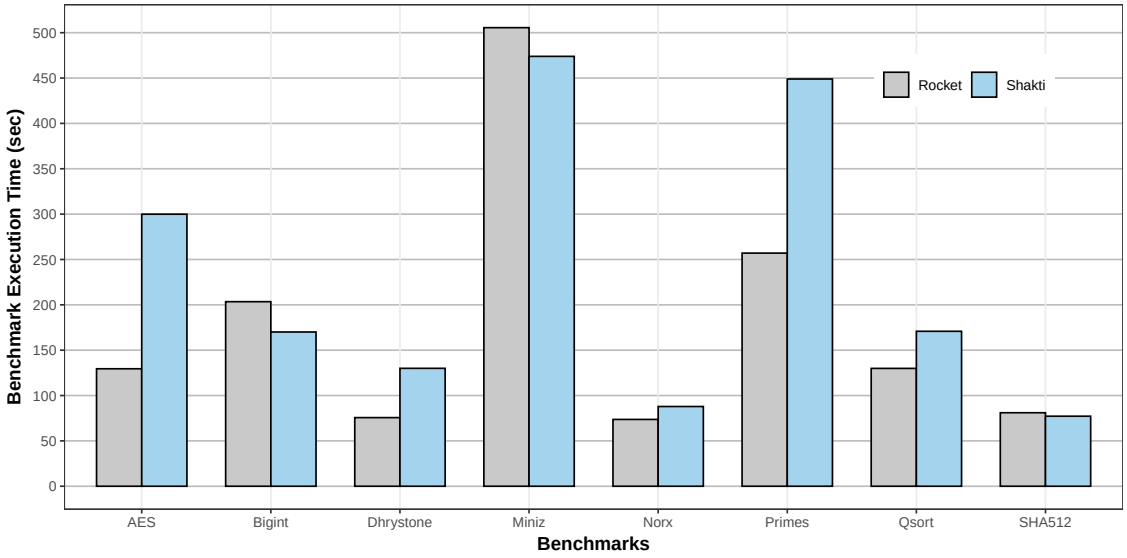



**Figure 15. Median Rocket and median Shakti bar graph for each benchmark.**

Again, the difference tests rejected $H_0^+$, indicating a distribution location shift of median execution time. Alternatively, the tests of equivalence at $\delta = 0.05$ rejected $H_0^-$ in all benchmarks except for SHA512. Here, the conclusion is a *relevant difference* in median performance between Rocket and Shakti on 7 out of 8 benchmarks and a *trivial difference* in SHA512. Another possible conclusion is from the effect sizes in Table 11 as the ratio of median performance, Rocket had a *relevant difference* of faster median execution times over Shakti in 6 of the 8 benchmarks.

In addition to the benchmark tests, a final Wilcoxon Signed-Rank Test was conducted to determine if evidence supports an overall *relevant difference* in performance between Rocket to Ariane, and Rocket to Shakti, in Table 12. Each test previously found *relevant differences* between Rocket and Ariane on all benchmarks, therefore the sample size was set at 8. In the Rocket to Ariane general performance comparison, $H_0^+$ was rejected, indicating there is a *relevant difference* in performance between the two systems. A similar test was conducted for the

**Table 12. SPARC general performance results for both comparisons.**

| | $M_{X/Y}$ | |
|---|---|---|
| **Benchmark** | **Ariane** | **Shakti** |
| AES | 0.4912 | 0.4324 |
| Bigint | 0.5193 | 1.1966 |
| Dhrystone | 0.2846 | 0.5821 |
| Miniz | 1.3019 | 1.0665 |
| Norx | 0.5662 | 0.8381 |
| Primes | 1.0622 | 0.5729 |
| Qsort | 0.5732 | 0.7606 |
| SHA512 | 0.5968 | Triv diff |
| Results $H_0^+$ | $p$ | Rej at $\alpha = 0.10$? |
| Ariane | 0.0391 | Yes |
| Shakti | 0.1094 | No |

Rocket to Shakti general performance comparison, with a different outcome. In the previous tests, one resulted in a *trivial difference* between Rocket and Shakti for SHA512. Therefore, it was removed from consideration as stated in Section 3.3.4 and the sample size reduced to 7.

Out of 7 benchmarks, Rocket outperformed Shakti on 7 of them, but not enough to reject $H_0^+$. The results are not unexpected. It is reasonable to assume that two embedded systems with varying levels of superior performance would require more than 8 benchmarks to reach a conclusion. The insight gained from the test of general performance between Rocket to Shakti is that the test failed to reject $H_0^+$ of equal performance and a follow-on experiment with additional benchmarks would be required for further determination.

### 5.2.1 Hypothetical Equivalence Margin Efficacy

Hypothetical equivalence margins were used to assess the efficacy of SPARC in scenarios where the performance of the systems with Keystone disabled are similar. The scenario considers $\delta_{Hyp1} = 0.25$ and $\delta_{Hyp2} = 0.50$ as the hypothetical extreme evaluations with equivalence margins $[0.75, 1.25]$, $[0.50, 1.50]$ to demonstrate the framework. As a note, the results from difference tests are unchanged because this scenario only affects equivalence tests but are listed again for reference.

The hypothetical equivalence results for Rocket to Ariane at $[0.75, 1.25]$ and $[0.50, 1.50]$ are listed in Table 13 and Table 14, respectively. The final Rocket to Ariane overall *relevant difference* in performance hypothesis test results for both hypothetical margins are listed in Table 15.

Remarkably, the effect of using a $[0.75, 1.25]$ margin had minimal impact to the overall *relevant difference* in performance test. From the experiment, Rocket had *relevant differences* of speedup (i.e. better performance) over Ariane on 6 of the 8 benchmarks. Primes was the only benchmark impacted in the evaluation because both $H_0^+$ and $H_0^-$ were rejected, indicating a *trivial difference*. This reduced the hypothesis test by one sample, but there was still a sufficient sample size to reject

**Table 13. SPARC framework results for difference and equivalence at $[0.75, 1.25]$ in Rocket to Ariane comparison tests**

| | | $H_0^+$ | | $H_{01}^-$ | | $H_{02}^-$ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Benchmark** | $M_{X/Y}$ | $z$ | $p$ | $z_1$ | $p_1$ | $z_2$ | $p_2$ | **Rej $H_0^+$ ?** | **Rej $H_0^-$ ?** | **Relevance** |
| AES | 0.4912 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Bigint | 0.5193 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Dhrystone | 0.2846 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Miniz | 1.3019 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | -4.792 | 0.992 | Yes | No | Rel diff |
| Norx | 0.5662 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Primes | 1.0622 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Qsort | 0.5732 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| SHA512 | 0.5968 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |

**Table 14. SPARC framework results for difference and equivalence at $[0.50, 1.50]$ in Rocket to Ariane comparison tests**

| Benchmark | $M_{X/Y}$ | $H_0^+$ | | $H_{01}^-$ | | $H_{02}^-$ | | Rej $H_0^+$ ? | Rej $H_0^-$ ? | Relevance |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $z$ | $p$ | $z_1$ | $p_1$ | $z_2$ | $p_2$ | | | |
| AES | 0.4912 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Bigint | 0.5193 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Dhrystone | 0.2846 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Miniz | 1.3019 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Norx | 0.5662 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Primes | 1.0622 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Qsort | 0.5732 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| SHA512 | 0.5968 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |

the overall performance null hypothesis, and conclude Rocket has better performance over Ariane.

Alternatively, at the $[0.50, 1.50]$ equivalence margin, the equivalence tests resulted in 6 of the 8 benchmarks as a *trivial difference*. In the overall *relevant difference* in performance hypothesis test for $[0.50, 1.50]$, $H_0^+$ was not rejected. The result is significant because it illustrates the capability of SPARC. If the equivalence margin $[0.50, 1.50]$ was a factual statement, rather than hypothetical, the results clearly demonstrate that a conclusion cannot be established given only two benchmark samples that show a *relevant difference*. In other words, there was not enough evidence to conclude that Rocket has better performance compared to Ariane.

**Table 15. Rocket to Ariane SPARC general performance results with hypothetical equivalence margins** $[0.75, 1.25]$ **and** $[0.50, 1.50]$**.**

| | **Ariane $M_{X/Y}$** | |
|---|---|---|
| **Benchmark** | **At $[0.75, 1.25]$** | **At $[0.50, 1.50]$** |
| AES | 0.4912 | 0.4912 |
| Bigint | 0.5193 | Triv Diff |
| Dhrystone | 0.2846 | 0.2846 |
| Miniz | 1.3019 | Triv Diff |
| Norx | 0.5662 | Triv Diff |
| Primes | Triv Diff | Triv Diff |
| Qsort | 0.5732 | Triv Diff |
| SHA512 | 0.5968 | Triv Diff |
| Results $H_0^+$ | $p$ | Rej at $\alpha = 0.10$? |
| At $[0.75, 1.25]$ | 0.03125 | Yes |
| At $[0.50, 1.50]$ | 0.5 | No |

The results show a similar conclusion found in the Rocket to Shakti comparison. The hypothetical equivalence results for Rocket to Shakti at $[0.75, 1.25]$ and $[0.50, 1.50]$ are listed in Table 16 and Table 17, respectively. The final Rocket to Shakti overall *relevant difference* in performance hypothesis test results for both hypothetical margins are listed in Table 18. Both hypothesis tests for overall *relevant difference* in performance failed to reject the null, which indicates there is not enough evidence to support a conclusion that Rocket has better performance than Shakti.

Suffice it to say, the SPARC framework excels in conditions of similar performance or equivalence. A valid conclusion for both system comparisons that failed to reject $H_0^+$ would indicate the need for additional benchmarks if further evaluation is pursued.

**Table 16. SPARC framework results for difference and equivalence at $[0.75, 1.25]$ in Rocket to Shakti comparison tests**

| Benchmark | $M_{X/Y}$ | $H_0^+$ | | $H_{01}^-$ | | $H_{02}^-$ | | Rej $H_0^+$ ? | Rej $H_0^-$ ? | Relevance |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $z$ | $p$ | $z_1$ | $p_1$ | $z_2$ | $p_2$ | | | |
| AES | 0.4324 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Bigint | 1.1966 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Dhrystone | 0.5821 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Miniz | 1.0665 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Norx | 0.8381 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Primes | 0.5729 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Qsort | 0.7606 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| SHA512 | 1.0493 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |

**Table 17. SPARC framework results for difference and equivalence at $[0.50, 1.50]$ in Rocket to Shakti comparison tests**

| Benchmark | $M_{X/Y}$ | $H_0^+$ | | $H_{01}^-$ | | $H_{02}^-$ | | Rej $H_0^+$ ? | Rej $H_0^-$ ? | Relevance |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $z$ | $p$ | $z_1$ | $p_1$ | $z_2$ | $p_2$ | | | |
| AES | 0.4324 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Bigint | 1.1966 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Dhrystone | 0.5821 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Miniz | 1.0665 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Norx | 0.8381 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Primes | 0.5729 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Qsort | 0.7606 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| SHA512 | 1.0493 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |

**Table 18. Rocket to Shakti SPARC general performance results with hypothetical equivalence margins $[0.75, 1.25]$ and $[0.50, 1.50]$.**

| | Shakti $M_{X/Y}$ | |
|---|---|---|
| **Benchmark** | **At $[0.75, 1.25]$** | **At $[0.50, 1.50]$** |
| AES | 0.4324 | 0.4324 |
| Bigint | Triv Diff | Triv Diff |
| Dhrystone | 0.5821 | Triv Diff |
| Miniz | Triv Diff | Triv Diff |
| Norx | Triv Diff | Triv Diff |
| Primes | 0.5729 | Triv Diff |
| Qsort | Triv Diff | Triv Diff |
| SHA512 | Triv Diff | Triv Diff |

| Results $H_0^+$ | $p$ | Rej at $\alpha = 0.10$? |
|---|---|---|
| At $[0.75, 1.25]$ | 0.25 | No |
| At $[0.50, 1.50]$ | 1.0 | No |

## 5.3 Individual RISC-V Performance Impact of Keystone with the SPARC Framework

In this section, results are presented to evaluate the performance impact of Keystone enabled on individual RISC-V embedded systems using SPARC. This section is only concerned with comparing the response variables, Benchmark Execution Time and Keystone Benchmark Execution Time, listed in Table 6. The Keystone Benchmark Execution Time is the raw benchmark performance within a secure enclave. The analysis excludes the Keystone Initialization Time response variable in the comparison, which is discussed in a separate section. Hypothetical equivalence margins are also not considered in this evaluation; their purpose was to demonstrate the SPARC framework and were presented in Section 5.2.

Figure 16 illustrates quantile-quantile plots for Rocket, with Keystone enabled versus disabled, on each benchmark with data points as paired-observation differences against a theoretical normal distribution line. The plots for AES, Bigint, Miniz, and Norx display non-normal distributions which would not be suitable for parametric

statistics tests. AES has a large gap between observations indicating a bimodal distribution. Whereas the data points in Bigint tend to flatten out towards the middle representing a heavy-tailed distribution. The left-curved data points of Miniz and Norx are characteristic of left-skewed distributions.

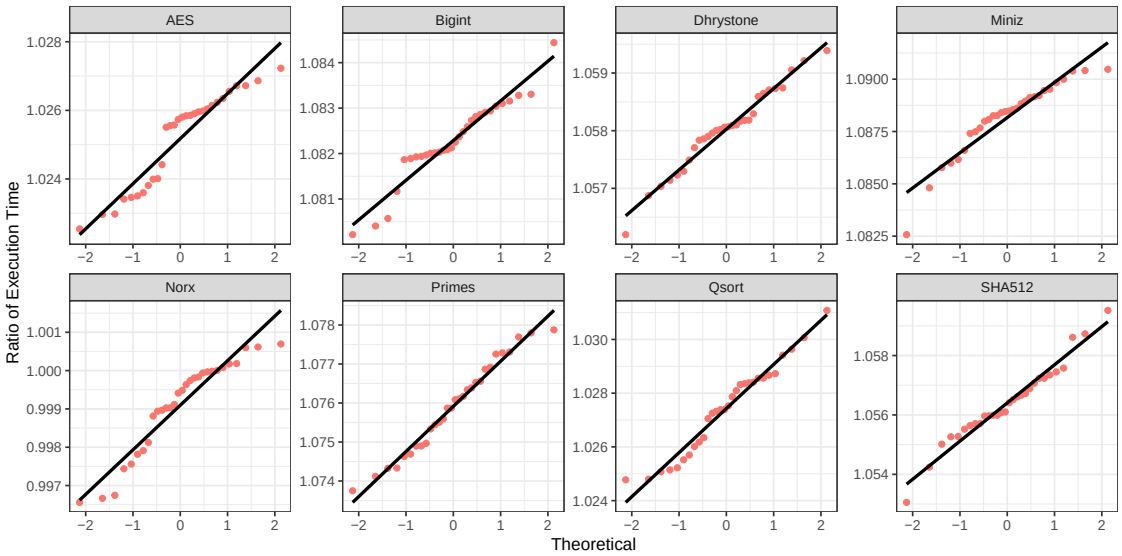



**Figure 16. Keystone performance impact on Rocket, quantile-quantile plots for each benchmark. Data points are a difference, Rocket Keystone enabled − Rocket Keystone disabled, compared to a theoretical normal distribution line.**

To assess normality of the Rocket Keystone data, Shapiro-Wilk Tests were conducted [64], with results listed in Table 19. Similar to Section 5.2, if $H_0$ is rejected then the distribution is not normal. Interestingly, Bigint failed to reject $H_0$ and is assumed normal. The likely explanation is that the heavy-tails affected the distribution, but not enough for the Shapiro-Wilk Test to reject $H_0$. As evident by the resulting p-value, 0.115, it is still above the rejection point $\alpha = 0.05$. If parametric tests were planned, the Bigint data would likely require a transformation to reduce the impact of the heavy-tail on the test. The other surprising result from the normality tests is that 5 of the 8 benchmarks failed to reject $H_0$ (i.e. assumed

**Table 19. Shapiro-Wilk Tests for normality on Keystone enabled versus disabled performance impact.**

| Sub-benchmark | Rocket | | | Ariane | | | Shakti | | |
|---|---|---|---|---|---|---|---|---|---|
| | $W$ | $p$ | Rej $H_0^+$? | $W$ | $p$ | Rej $H_0^+$? | $W$ | $p$ | Rej $H_0^+$? |
| AES | 0.9014 | 9.10e-3 | Yes | 0.8840 | 3.50e-3 | Yes | 0.9325 | 5.72e-2 | No |
| Bigint | 0.9438 | 1.15e-1 | No | 0.8971 | 7.14e-3 | Yes | 0.9564 | 2.50e-1 | No |
| Dhrystone | 0.9758 | 7.05e-1 | No | 0.9615 | 3.37e-1 | No | 0.2869 | 5.48e-11 | Yes |
| Miniz | 0.9073 | 1.27e-2 | Yes | 0.6100 | 9.68e-8 | Yes | 0.6463 | 2.82e-7 | Yes |
| Norx | 0.9050 | 1.12e-2 | Yes | 0.5669 | 2.95e-8 | Yes | 0.4797 | 3.29e-9 | Yes |
| Primes | 0.9728 | 6.17e-1 | No | 0.9412 | 9.80e-2 | No | 0.8294 | 2.36e-4 | Yes |
| Qsort | 0.9613 | 3.35e-1 | No | 0.9485 | 1.54e-1 | No | 0.7050 | 1.84e-6 | Yes |
| SHA512 | 0.9712 | 5.73e-1 | No | 0.6820 | 8.62e-7 | Yes | 0.9038 | 1.04e-2 | Yes |

normal). In terms of significance, the paired observations likely removed non-normal characteristics shared between the samples on the Rocket system. This is an expected result of dependent samples in statistics. But, there were 3 tests that nevertheless resulted in non-normality and the methodology in SPARC still applies.

Similarly, quantile-quantile plots are displayed in Figure 17 and Figure 18 for Ariane and Shakti, respectively. Results from Shapiro-Wilk Tests are also listed in Table 19. In contrast to the Rocket results, the majority of plots and tests for both Ariane and Shakti indicated the benchmark distributions are non-normal.

One possible conspicuous explanation is possible correlation between the distribution of performance data and the RISC-V system that Keystone was originally developed on, Rocket. The Keystone development team uses Rocket cores instantiated either on an ASIC or an FPGA, therefore performance inefficiencies specific to the Rocket system are noticeable and can be analyzed. Additionally, while all three systems are open-source, the Rocket core has had a longer development timeline which could have resulted in a system that is more mature with less problems (i.e. bugs). This explanation is purely speculative; the experiment in this research did not test for such evidence. It is left as an avenue for

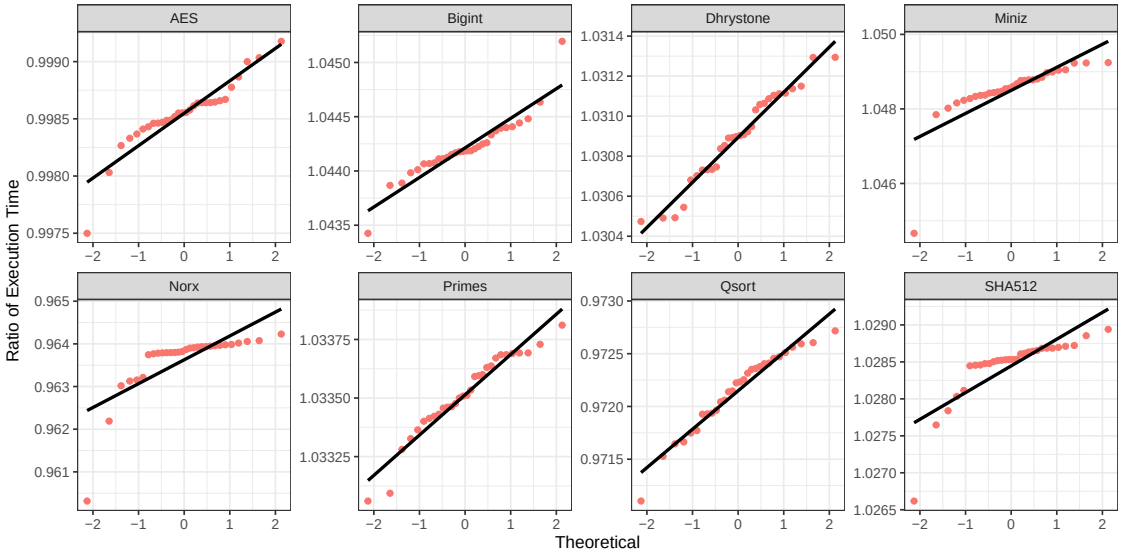



**Figure 17. Keystone performance impact on Ariane, quantile-quantile plots for each benchmark. Data points are a difference, Ariane Keystone enabled − Ariane Keystone disabled, compared to a theoretical normal distribution line.**

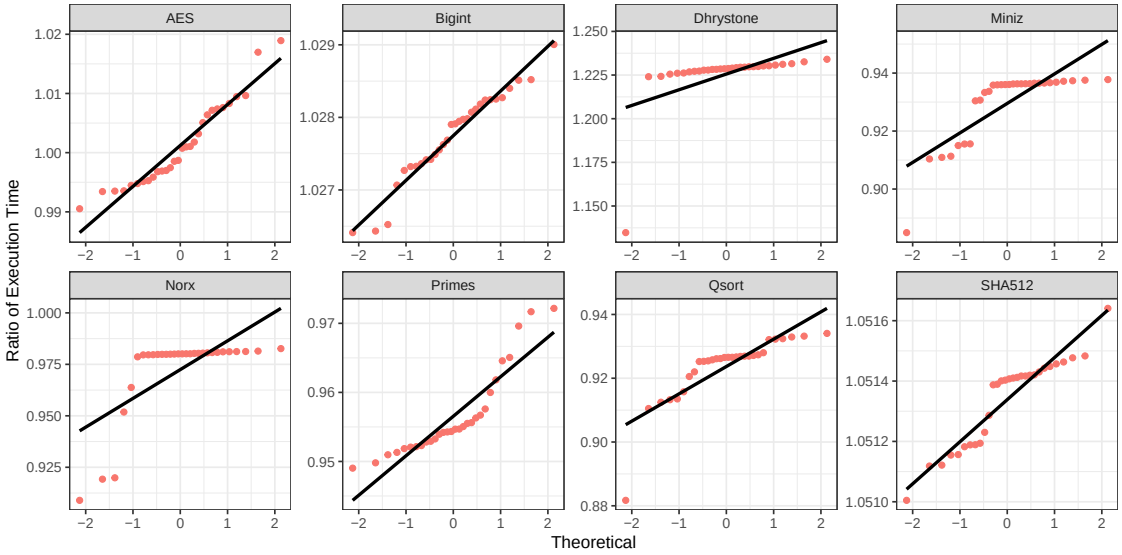



**Figure 18. Keystone performance impact on Shakti, quantile-quantile plots for each benchmark. Data points are a difference, Shakti Keystone enabled − Shakti Keystone disabled, compared to a theoretical normal distribution line.**

**Table 20. SPARC framework results for difference and equivalence at $[0.95, 1.05]$ in Rocket with Keystone disabled to Keystone enabled comparison tests**

| Benchmark | $M_{KD}$ (sec) | $M_{KE}$ (sec) | $M_{KD/KE}$ | $H_0^+$ | | $H_{01}^-$ | | $H_{02}^-$ | | Rej $H_0^+$? | Rej $H_0^-$? | Relevance |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $z$ | $p$ | $z_1$ | $p_1$ | $z_2$ | $p_2$ | | | |
| AES | 129.5164 | 132.8724 | 0.9747 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Bigint | 203.4705 | 220.2293 | 0.9239 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Dhrystone | 75.6602 | 80.0820 | 0.9448 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Miniz | 505.5659 | 550.4717 | 0.9184 | 3.414 | 6.394e-4 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Norx | 73.6444 | 73.6298 | 1.0002 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Primes | 257.0185 | 276.6503 | 0.9290 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |
| Qsort | 129.927 | 133.5821 | 0.9726 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| SHA512 | 81.0285 | 85.5810 | 0.9468 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | No | Rel diff |

future work.

The non-normal data for each RISC-V system was assessed with SPARC to characterize the performance impact of Keystone enabled to Keystone disabled. Similar to Section 5.2, difference and equivalence hypotheses tests were conducted with the ratio of medians. The results for Rocket, Ariane, and Shakti, are listed in Table 20, Table 21, and Table 22 respectively. Bar graphs for the three systems are illustrated at the end of this section in Figure 19, Figure 20, and Figure 21. The median execution time of a benchmark with Keystone disabled $M_{KD}$ and with Keystone enabled $M_{KE}$ are listed along with the median speedup ratio of execution time $M_{KD}/M_{KE}$. The median speedup ratio reference value is 1, which would indicate equal performance. Values greater than 1 indicate the performance with Keystone enabled was better (i.e. faster time to execute the benchmark), whereas values less than 1 denote a loss in performance (i.e. slower time to execute the benchmark).

Difference tests for the Rocket system rejected $H_0^+$ in all benchmarks and therefore found a statistically significant difference in performance between enabling and disabling Keystone. Tests for equivalence were conducted at $\delta = 0.05$, which resulted in rejecting $H_0^-$ for benchmarks AES, Norx, and Qsort but failing to reject for the other 5. The tests found the performance impact in benchmarks AES, Norx,

**Table 21. SPARC framework results for difference and equivalence at $[0.95, 1.05]$ in Ariane with Keystone disabled to Keystone enabled comparison tests**

| Benchmark | $M_{KD}$ (sec) | $M_{KE}$ (sec) | $M_{KD/KE}$ | $H_0^+$ | | $H_{01}^-$ | | $H_{02}^-$ | | Rej $H_0^+$ ? | Rej $H_0^-$ ? | Relevance |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $z$ | $p$ | $z_1$ | $p_1$ | $z_2$ | $p_2$ | | | |
| AES | 263.6765 | 263.2871 | 1.0015 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Bigint | 391.8052 | 409.1114 | 0.9577 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Dhrystone | 265.857 | 274.0711 | 0.9700 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Miniz | 388.3211 | 407.1603 | 0.9537 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Norx | 130.0691 | 125.369 | 1.0375 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Primes | 242.0116 | 250.1258 | 0.9676 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Qsort | 226.6757 | 220.3829 | 1.0286 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| SHA512 | 135.7358 | 139.6111 | 0.9722 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |

**Table 22. SPARC framework results for difference and equivalence at $[0.95, 1.05]$ in Shakti with Keystone disabled to Keystone enabled comparison tests**

| Benchmark | $M_{KD}$ (sec) | $M_{KE}$ (sec) | $M_{KD/KE}$ | $H_0^+$ | | $H_{01}^-$ | | $H_{02}^-$ | | Rej $H_0^+$ ? | Rej $H_0^-$ ? | Relevance |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | $z$ | $p$ | $z_1$ | $p_1$ | $z_2$ | $p_2$ | | | |
| AES | 299.9041 | 299.1829 | 1.0024 | -0.576 | 0.5647 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | No | Yes | Equivalence |
| Bigint | 170.0411 | 174.7781 | 0.9729 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |
| Dhrystone | 129.9888 | 159.7625 | 0.8136 | 4.772 | 1.825e-6 | -4.792 | 0.992 | 4.772 | 9.127e-7 | Yes | Yes | Rel diff |
| Miniz | 473.9085 | 443.6228 | 1.0683 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | -4.792 | 0.992 | Yes | No | Rel diff |
| Norx | 87.8986 | 86.1459 | 1.0203 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 2.962 | 1.529e-3 | Yes | Yes | Triv diff |
| Primes | 448.9452 | 428.4520 | 1.0478 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 3.456 | 2.747e-4 | Yes | Yes | Triv diff |
| Qsort | 170.8250 | 158.2771 | 1.0793 | -4.772 | 1.825e-6 | 4.772 | 9.127e-7 | -4.792 | 0.992 | Yes | No | Rel diff |
| SHA512 | 77.2075 | 81.1778 | 0.9511 | 4.772 | 1.825e-6 | 4.772 | 9.127e-7 | 4.772 | 9.127e-7 | Yes | Yes | Triv diff |

and Qsort was within the equivalence margin and therefore similar in performance. Thus, the relevance conclusions for 3 out of the 8 benchmarks are designated as *trivial difference*s. In other words, the impact that Keystone has on performance on the 3 benchmarks on the Rocket system was less than what this research determined relevant.

Likewise, tests for the Ariane system also rejected $H_0^+$ in all benchmarks. On the other hand, the tests for equivalence all rejected $H_0^-$ indicating the performance impact of Keystone enabled was *equivalent* to Keystone disabled. Most surprisingly, this resulted in all the benchmarks denoted as *trivial difference* for Ariane. To be specific, the performance of Keystone enabled compared to Keystone disabled was a *trivial difference* for all 8 benchmarks that were tested on Ariane. This surprising result will factor in the overall *relevant difference* in performance test conducted further in the text.

Another substantial result was found by SPARC in difference and equivalence tests on the Shakti system. In contrast to the other systems, the difference tests for Shakti rejected $H_0^+$ in all but one benchmark, AES. The speedup ratio $M_{KD}/M_{KE}$ on AES is bordering on the reference value of 1, signifying similar performance whether Keystone is enabled or disabled. If the data observations of Keystone enabled and Keystone disabled were distributed with low variance (i.e. tightly distributed in a small area) then the difference test would have rejected $H_0^+$. Likely, the variance of the data is resulting in the test failing to find enough evidence to support a difference from the reference value. The result could change if more samples were provided in additional experiments.

In the tests for equivalence of the Shakti system, 6 of the 8 benchmarks rejected $H_0^-$ finding enough evidence to conclude the performance between Keystone enabled and Keystone disabled was *equivalent*. Pertaining to the relevance conclusions, the

result of rejecting both $H_0^+$ and $H_0^-$ on AES led to a determination of equivalence for that particular benchmark. Out of the 7 benchmarks left, 4 are considered as *trivial difference* and 3 have a *relevant difference* in performance.

Finally, an overall *relevant difference* in performance test was conducted for each system and the results listed in Table 23. The test is a Wilcoxon Signed-Rank Test on the median speedup ratio $M_{KD}/M_{KE}$ for benchmarks denoted as a *relevant difference* in performance. The total sample size starts at the number of benchmarks, 8, and any *trivial difference* or equivalence benchmarks reduce the size by 1. For the Rocket system, there were 5 benchmarks of *relevant difference* with an

**Table 23. SPARC overall *relevant difference* in performance test results of Keystone enabled to Keystone disabled for all 3 RISC-V systems.**

| | $M_{KD/KE}$ | | |
|---|---|---|---|
| **Benchmark** | **Rocket** | **Ariane** | **Shakti** |
| AES | Triv diff | Triv diff | Equivalence |
| Bigint | 0.9239 | Triv diff | Triv diff |
| Dhrystone | 0.9448 | Triv diff | 0.8136 |
| Miniz | 0.9184 | Triv diff | 1.068 |
| Norx | Triv diff | Triv diff | Triv diff |
| Primes | 0.9290 | Triv diff | Triv diff |
| Qsort | Triv diff | Triv diff | 1.079 |
| SHA512 | 0.9468 | Triv diff | Triv diff |
| Results $H_0^+$ | $p$ | Rej at $\alpha = 0.10$? | Effect Size (%) |
| Rocket | 0.0625 | Yes | 6.74 |
| Ariane | - | - | - |
| Shakti | 0.5 | No | - |

effect size towards slower performance with Keystone enabled. The difference test rejected $H_0^+$ with a p-value of 0.0625. This research concludes the test found a statistically significant difference of performance occurs with Keystone enabled on the Rocket RISC-V embedded system on 5 benchmarks, with an effect magnitude

and direction of 6.74% decrease in speedup.

For Ariane, an overall *relevant difference* in performance test was not conducted considering all the benchmarks were denoted as a *trivial difference*. Therefore, this research concludes SPARC found a statistically significant, *trivial difference* of performance between Ariane with Keystone enabled to Keystone disabled on 8 benchmarks. Another valid conclusion could state SPARC found no difference of performance occurs since this research determines *trivial differences* as insignificant.

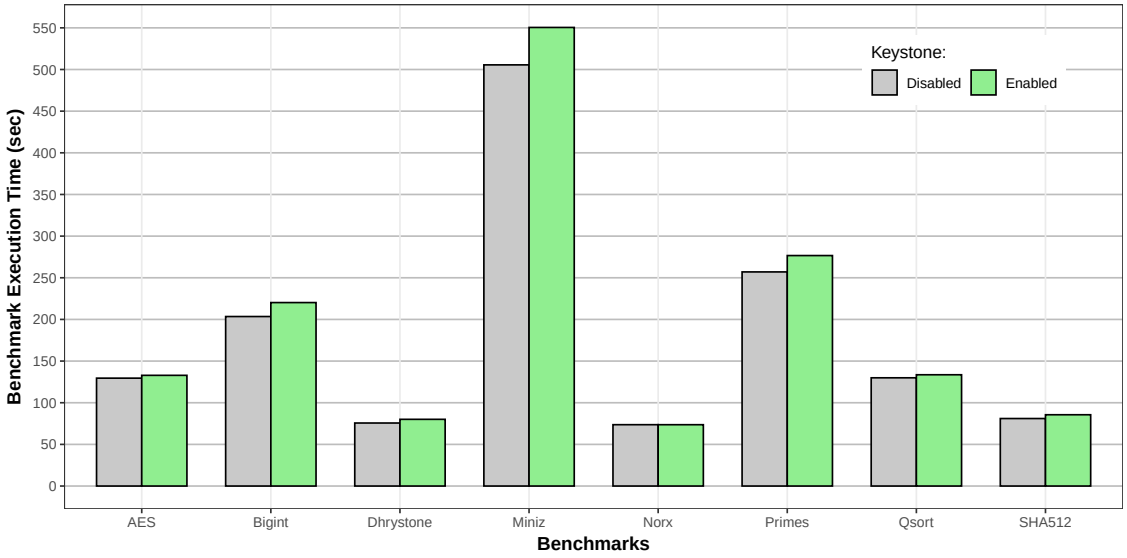



**Figure 19. Rocket median Keystone enabled and median Keystone disabled bar graph for each benchmark.**

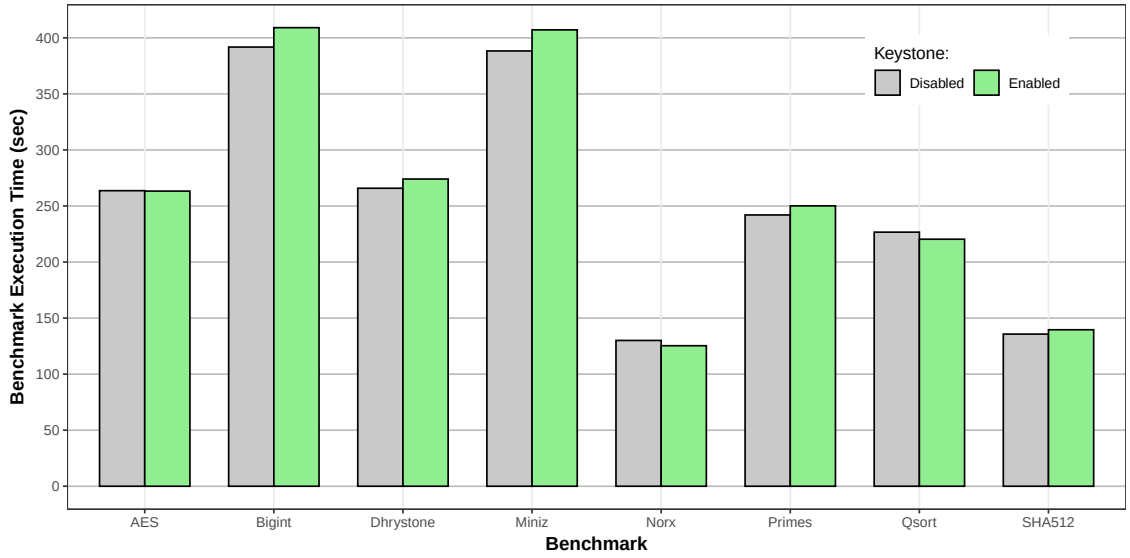



**Figure 20. Ariane median Keystone enabled and median Keystone disabled bar graph for each benchmark.**

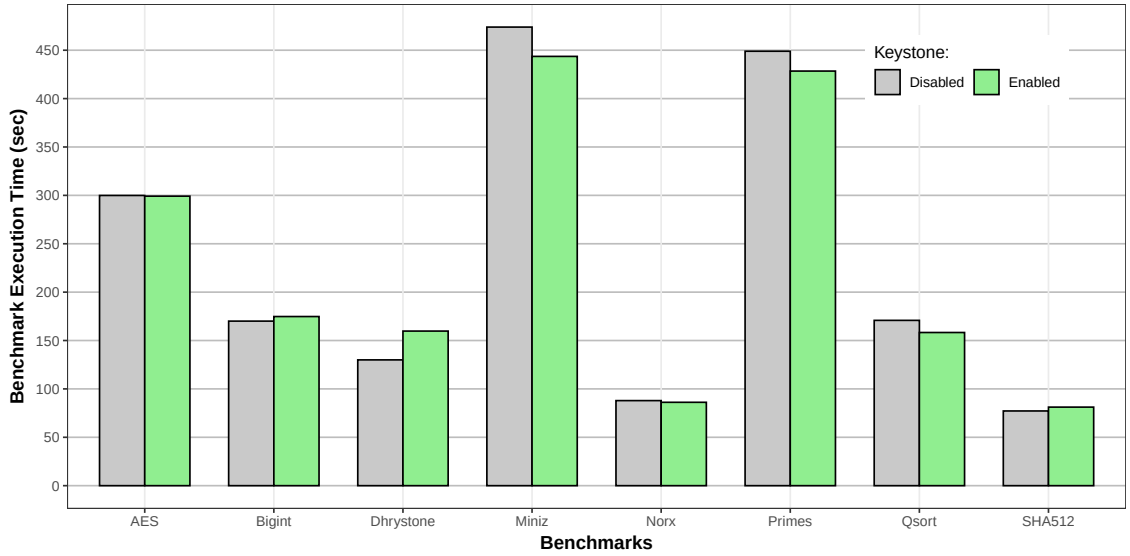



**Figure 21. Shakti median Keystone enabled and median Keystone disabled bar graph for each benchmark.**

## 5.4 Efficacy of the SPARC Framework in Comparison to HPT

In order to compare the efficacy of SPARC to the HPT [1] framework, this research considers some differences with respect to the benchmark statistics tests. As noted in Section 4.9, the observations are pairwise between processors and more appropriate for the Wilcoxon Signed-Rank Test used in SPARC. In HPT, the Wilcoxon Rank-Sum Test is usable on pairwise comparisons, but some information common to both populations is lost. Tests suitable for a difference of observations likely remove variability shared between the two observations. In contrast to the Wilcoxon Rank-Sum Test, which compares two independent observations.

The test statistic is another key difference between SPARC and HPT. In SPARC, the methodology specifically identifies the speedup ratio between processors as the test statistic, whereas HPT designates an unspecified performance score. Again, the key disparity derives from using Wilcoxon Signed-Rank Test or Wilcoxon Rank-Sum Test and how each framework classifies response variables as paired or unpaired. Therefore, the HPT tests are performed according to the procedures in [1] and designate the median execution time as the test statistic.

For HPT, two-tailed Wilcoxon Rank-Sum Tests were performed for each benchmark to determine whether Rocket or Ariane has a difference in median performance, listed in Table 24. The median execution times are unchanged from Table 10, therefore the 90% confidence intervals are listed instead. For each test, $H_0^+$ was rejected at $\alpha = 0.10$ indicating a difference in benchmark performance between the two processors. Similarly, the same tests are performed for Rocket to Shakti with results listed in Table 25. For each benchmark, $H_0^+$ was rejected indicating a difference in performance between Rocket and Shakti.

Finally, a two-tailed Wilcoxon Signed-Rank Test is performed as the HPT general performance comparison across all benchmarks. The test was conducted

**Table 24. HPT framework results for Wilcoxon Rank-Sum Test in Rocket to Ariane comparison.**

| Sub-benchmark | CI | | $H_0^+$ | Rej $H_0^+$ at $\alpha$? |
|---|---|---|---|---|
| | LB | UB | $p$ | |
| AES | -134.1845 | -134.0082 | 3.02e-11 | Yes |
| Bigint | -188.3599 | -188.3112 | 3.02e-11 | Yes |
| Dhrystone | -190.207 | -190.1609 | 3.02e-11 | Yes |
| Miniz | 117.0206 | 117.4667 | 3.02e-11 | Yes |
| Norx | -56.4443 | -56.3996 | 3.02e-11 | Yes |
| Primes | 14.9968 | 15.2686 | 3.02e-11 | Yes |
| Qsort | -96.7828 | -96.5662 | 3.02e-11 | Yes |
| SHA512 | -54.7757 | -54.6912 | 3.02e-11 | Yes |

**Table 25. HPT framework results for Wilcoxon Rank-Sum Test in Rocket to Shakti comparison.**

| Benchmark | CI | | $H_0^+$ | Rej $H_0^+$ at $\alpha$? |
|---|---|---|---|---|
| | LB | UB | $p$ | |
| AES | -170.6220 | -168.0001 | 3.02e-11 | Yes |
| Bigint | 33.3835 | 33.4687 | 3.02e-11 | Yes |
| Dhrystone | -54.3497 | -54.2725 | 3.02e-11 | Yes |
| Miniz | 30.8791 | 31.8176 | 3.02e-11 | Yes |
| Norx | -14.2729 | -14.1803 | 3.02e-11 | Yes |
| Primes | -192.1151 | -191.4319 | 3.02e-11 | Yes |
| Qsort | -41.0566 | -40.6661 | 3.02e-11 | Yes |
| SHA512 | 3.7647 | 3.8380 | 3.02e-11 | Yes |

**Table 26. HPT general performance results for both comparisons.**

| | $M_X - M_Y$ | |
|---|---|---|
| **Benchmark** | **Ariane** | **Shakti** |
| AES | -134.1601 | -170.3877 |
| Bigint | -188.3348 | 33.4293 |
| Dhrystone | -190.1969 | -54.3286 |
| Miniz | 117.2447 | 31.6575 |
| Norx | -56.4247 | -14.2542 |
| Primes | 15.0069 | -191.9267 |
| Qsort | -96.7487 | -40.8980 |
| SHA512 | -54.7073 | 3.820 |

| Results $H_0^+$ | $p$ | Rej at $\alpha = 0.10$? |
|---|---|---|
| Ariane | 0.1094 | No |
| Shakti | 0.1953 | No |

twice, on Rocket to Ariane and Rocket to Shakti, listed in Table 26. On both general performance tests, HPT failed to reject $H_0^+$.

In Section 3.3.2.2, family-wise error was discussed, in addition to possible risks to a study if $\alpha$ is not corrected. Determining if the data tested is within a family, and therefore affected by FWER, can be subjective. But, the HPT framework lacks discussion on multiple hypothesis testing, nor does it discuss methods to correct $\alpha$. This research considers the omission as accidental and purposefully discussed FWER in the SPARC procedures to remove ambiguity.

In comparison to SPARC, the Rocket to Ariane benchmark results by HPT illustrate the difference between each framework's concluding information. Initially, there is no substantial information provided by SPARC different from HPT. Both frameworks found differences that resulted in rejecting $H_0^+$. But in SPARC, a second test was conducted for equivalence and concluded there was a *relevant difference* on each benchmark.

On the other hand, the HPT Rocket to Shakti benchmark results compared to SPARC are noticeable. Specifically, in the HPT test on SHA512, $H_0^+$ was rejected compared to a *trivial difference* result in SPARC. As indicated by the follow-on equivalence test, the difference in performance was within the $[0.95, 1.05]$ margin and subsequently removed from the general performance comparison. While both SPARC and HPT failed to reject $H_0^+$ in the general performance test, this research is able to use the additional insights from SPARC to influence follow-on experimental design.

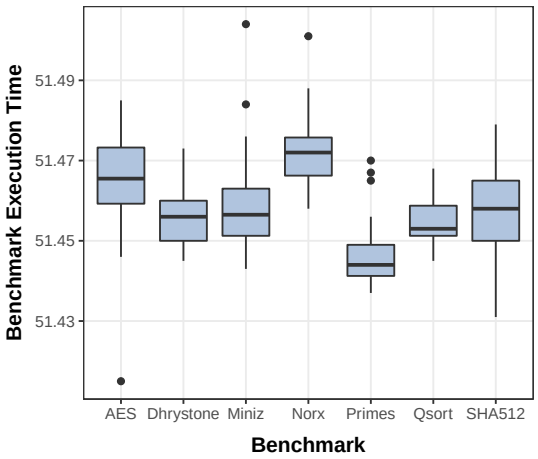
## 5.5 Performance Impact of Keystone-specific Security Features

This section presents the Keystone Initialization Time analysis for the framework model discussed in Section 4.10, which was conducted with the Rocket RISC-V embedded system. It also discusses outliers discovered by Bigint during analysis and a follow-on experiment conducted to discover the underlying cause.

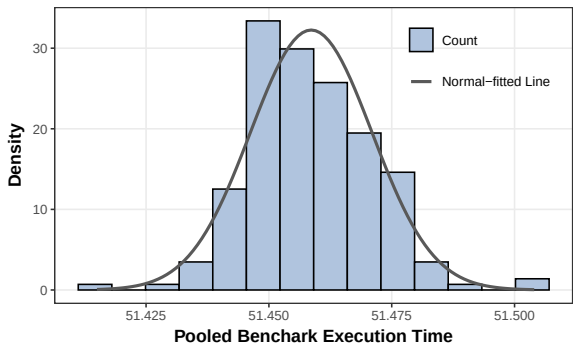
### 5.5.1 Evaluation of Keystone Closed-queuing Model

In order to evaluate the closed-queuing model, it was necessary to first determine the distribution of execution times (clock cycles consumed by Keystone) required to initialize an enclave and prepare $E_{RT}$ for all eight benchmarks.

Each benchmark was executed 30 times in order to apply the CLT for modeling the normal distribution of execution times. While graphing the distributions, data from the Bigint benchmark displayed execution times of approximately seven seconds longer compared to the other benchmarks. The findings led to an additional experiment which factored a variable for binary size and its impact on the overhead costs, which is discussed in a following section. The remainder of this analysis does not include data from the Bigint benchmark.




(a) Benchmark boxplots of mean execution time. Seven benchmark boxplots excluding Bigint.




(b) Testing normal distribution through exploratory data analysis. Distribution for pooled mean execution times for benchmarks excluding Bigint.

**Figure 22. cap**

Boxplots of the benchmark results, excluding Bigint, are provided in Figure **??**. Normal distribution fit line plots are displayed in Figure 22b.

#### 5.5.1.1 Equivalence Test of Means

Graphical exploratory data analysis of the pooled execution times in Figure 22b indicates the distribution is normal and sufficient to conduct an equivalence test with parametric statistics. The parametric equivalence test for paired samples is the TOST procedure with the t-test. The test was conducted within the statistical analysis program, JMP, to specify a range for mean execution time that includes a 50 ms margin of error for the non-deterministic properties of a processor. The equivalence margin is an arbitrary value; it can be increased or decreased based on the system design and operational time requirements.

There is a notable difference in the equivalence margin selected for this scenario compared to the equivalence margin procedures outlined by SPARC. This research chose to use execution time rather than a speedup ratio here because analysis was

completed prior to the SPARC framework development and insights gained therein. Ultimately, either can be used in the analysis but the speedup ratio had the benefit of abstracting out units of measurement. But, the model evaluated in this section considers determining the average system response time in Equation 17 and therefore suitable to use the execution time.

Results from the tests listed in Table 27 indicated there is no statistically significant difference to a mean execution time of 51.458s, which will then be used for finding the average system response time in the closed queuing network. The results in the table list each equivalence test on the null hypothesis that the mean difference is greater than or less than 50 ms.

#### 5.5.1.2 Throughput Calculation of Keystone Reference Monitor

A subset of the values used in [4] to calculate the average service time are used in this calculation. The obvious changes include the throughput calculation to $X_3$ based on the mean execution time of 51.458 seconds, observed in the experiments. The other variables are only theoretical as they were not measured in the data collection. The routing frequency, $q_{13}$, was altered from 0.4 to 0.005; in this example, only a small fraction of all jobs will require an enclave creation request.

Returning to the model, the throughput calculation of $X_3$ is as follows:

$$X_3 = \frac{1}{51.458} = 0.0194 \; accesses/second \tag{18}$$

Equation 18 alters the throughputs of $X_1$ and $X_0$ as follows:

$$X_1 = \frac{0.0194 \; accesses/second}{0.005} = 3.88 \; accesses/second \tag{19}$$

**Table 27. Results for equivalence tests on means between two benchmarks.**

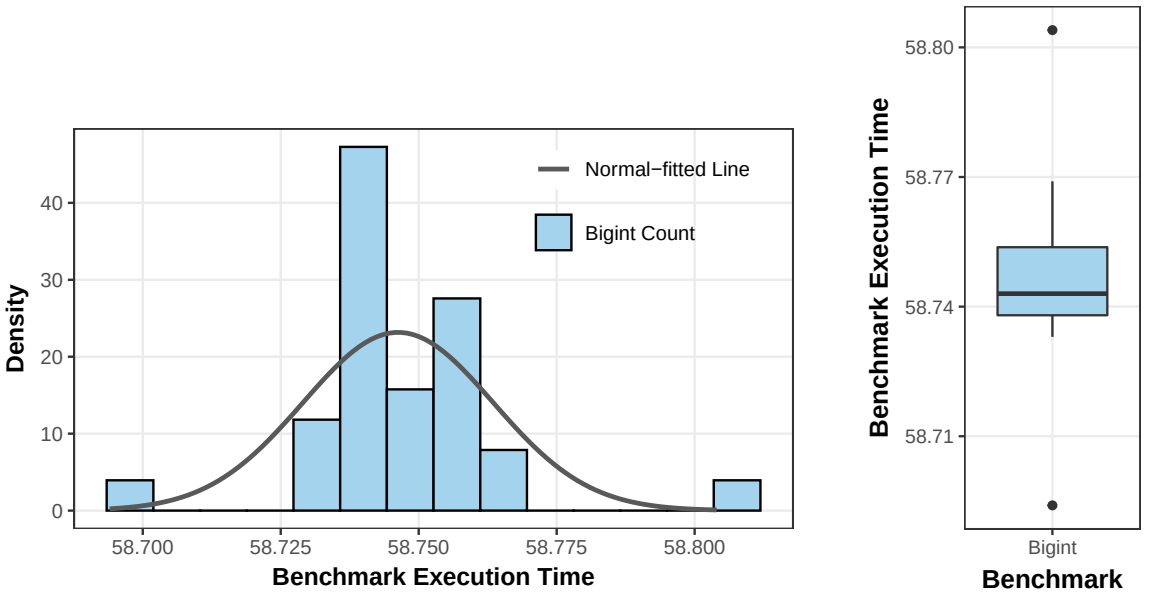
| Benchmarks Tested | t-Ratios for Mean Diff $\geq 0.05$ | t-Ratios for Mean Diff $\leq -0.05$ | Deg. of Freedom | $p$ | Rej $H_0^-$ at $\alpha = 0.10$? |
|---|---|---|---|---|---|
| Dhrystone to AES | -22.29 | 16.03 | 203 | $< .0001$ | Yes |
| Miniz to AES | -21.05 | 17.27 | 203 | $< .0001$ | Yes |
| Miniz to Dhrystone | -17.92 | 20.39 | 203 | $< .0001$ | Yes |
| Norx to AES | -16.18 | 22.13 | 203 | $< .0001$ | Yes |
| Norx to Dhrystone | -13.05 | 25.26 | 203 | $< .0001$ | Yes |
| Norx to Miniz | -14.29 | 24.02 | 203 | $< .0001$ | Yes |
| Primes to AES | -25.82 | 12.49 | 203 | $< .0001$ | Yes |
| Primes to Dhrystone | -22.69 | 15.62 | 203 | $< .0001$ | Yes |
| Primes to Miniz | -23.93 | 14.38 | 203 | $< .0001$ | Yes |
| Primes to Norx | -28.80 | 9.51 | 203 | $< .0001$ | Yes |
| Qsort to AES | -22.86 | 15.45 | 203 | $< .0001$ | Yes |
| Qsort to Dhrystone | -19.73 | 18.58 | 203 | $< .0001$ | Yes |
| Qsort to Miniz | -20.97 | 17.34 | 203 | $< .0001$ | Yes |
| Qsort to Norx | -25.84 | 12.48 | 203 | $< .0001$ | Yes |
| Qsort to Primes | -16.19 | 22.12 | 203 | $< .0001$ | Yes |
| SHA512 to AES | -21.83 | 16.49 | 203 | $< .0001$ | Yes |
| SHA512 to Dhrystone | -18.70 | 19.62 | 203 | $< .0001$ | Yes |
| SHA512 to Miniz | -19.93 | 18.38 | 203 | $< .0001$ | Yes |
| SHA512 to Norx | -24.80 | 13.51 | 203 | $< .0001$ | Yes |
| SHA512 to Primes | -15.16 | 23.15 | 203 | $< .0001$ | Yes |
| SHA512 to Qsort | -18.12 | 20.19 | 203 | $< .0001$ | Yes |

$$X_0 = (0.05) \cdot (3.88\ accesses/second) = 0.194\ accesses/second \tag{20}$$

Assuming the system has 18 subjects; $\lambda_{OS} = 0.05$ accesses/second; the resulting average system response time and impact overhead of Keystone initialization is $R =$ 72.8 seconds.

### 5.5.2 Keystone Initialization Time: Analysis of Binary Size Performance Impact

In Section 5.5.1, larger (i.e. slower) Keystone Initialization Times were discovered with respect to Bigint while conducting exploratory data analysis of the benchmark results. Additionally, the slower times were not observed in any of the remaining 7 benchmarks. A boxplot of the data collected from Bigint is displayed in Figure 23b




(a) Testing normal distribution of execution times.

(b) Boxplot of mean execution time, indicating higher execution times.

**Figure 23. Exploratory data analysis of Bigint execution times.**

and the distribution is graphed in Figure 23a.

An initial investigation was conducted and found some of the processes in Keystone's enclave initialization code are impacted by the size of the binary, which is supported and briefly mentioned in the Keystone research paper [10]. The sizes of the benchmark binaries used in Section 5.5.1 are listed in Table 28.

In the enclave initialization sequence, the untrusted Linux OS allocates physical and virtual memory for the benchmark, trusted OS, and enclave. The untrusted Linux OS passes the virtual page table (i.e. a mapping from virtual memory to physical memory) to Keystone which will verify each location is correct and then secure the enclave with encryption. The processes of location verification and enclave encryption are the security features assumed to factor in the Keystone Initialization Time discovered with Bigint. Furthermore, there is another factor assumed as a minimum Keystone Initialization Time regardless of the benchmark or application.

**Table 28. Binary sizes of benchmarks.**

| Benchmark | Binary Size (KB) |
|---|---|
| AES | 165.8 |
| Bigint | 738.4 |
| Dhrystone | 138.4 |
| Miniz | 524.1 |
| Norx | 169.2 |
| Primes | 129.4 |
| Qsort | 138.9 |
| SHA512 | 143.4 |

**Table 29. Compiled Qsort size targets for experiment.**

| Final Qsort Size (KB) |
|---|
| 250 |
| 500 |
| 750 |
| 1000 |
| 5000 |
| 10000 |
| 50000 |

To determine if supporting evidence exists, another experiment was conducted on the FPGA using the Rocket RISC-V embedded system configuration listed in Table 4. The experimental design was adding code to a benchmark's file in order to increase its size to specific targets after it was compiled. Qsort was chosen as the benchmark due to its relatively faster execution time and its 7 size targets were selected arbitrarily to cover a wide margin and listed in Table 29. The additional code

simply declared a large array of alphabetical letters in the file that was never executed by the program, thus it artificially inflated the binary size and did not impact the benchmark's execution time.

The statistical software, JMP, was used to expedite a fractional factorial experiment design. Fractional factorial experiments are commonly used with linear regression analysis [16] to explore linear relationships between factor variables and response variables. The factor variables in the experiment were the binary size and whether Keystone was enabled or disabled. The response variable was the Keystone Initialization Time. After entering the variables into JMP, it provided a fractional factorial design that listed the number of runs required to test if a linear relationship exists for the experiment along with a factor variable configuration randomly selected for each run. Although 7 binary sizes were input, only the following 3 were listed as needed in the run configurations: 250 KB, 10000 KB, and 50000 KB. The experiment design required a total of 25 experiment runs and each run alternated between Keystone enabled and Keystone disabled. A script was written to batch execute the experiment runs successively with the configurations provided by the design.

The experiment run configurations and results are provided in Table 30. Runs with Keystone disabled do not have a Keystone Initialization Time and thus display 0. The Benchmark Execution Time is the time it takes for Qsort to execute and does not include the Keystone Initialization Time. The Total Execution Time is the total time to execute Qsort plus the Keystone Initialization Time, if Keystone was enabled.

The results are consistent with the initial investigation; as the binary size increases, the Keystone Initialization Time also increases. It also confirms the assumption that only the Keystone Initialization Time is affected, as the Benchmark Execution Time stayed relatively consistent regardless of the binary size.

**Table 30. Binary size experiment benchmark data output and run configurations.**

| Run | Binary Size | Keystone Status | Keystone Init. Time (sec) | Benchmark Exec. Time (sec) | Total Exec. Time (sec) |
|---|---|---|---|---|---|
| 1 | 10000 | Disabled | 0 | 129.1859 | 129.1859 |
| 2 | 10000 | Enabled | 349.9144 | 132.9273 | 482.8418 |
| 3 | 10000 | Enabled | 326.7494 | 132.9506 | 459.6999 |
| 4 | 250 | Disabled | 0 | 128.9667 | 128.9667 |
| 5 | 250 | Enabled | 111.1440 | 132.8586 | 244.0026 |
| 6 | 10000 | Disabled | 0 | 128.7949 | 128.7949 |
| 7 | 250 | Enabled | 111.1425 | 132.8389 | 243.9814 |
| 8 | 250 | Disabled | 0 | 128.7927 | 128.7927 |
| 9 | 250 | Enabled | 111.1347 | 132.8613 | 243.9960 |
| 10 | 50000 | Disabled | 0 | 129.1704 | 129.1704 |
| 11 | 50000 | Enabled | 1341.7015 | 132.8444 | 1474.5459 |
| 12 | 50000 | Disabled | 0 | 128.9374 | 128.9374 |
| 13 | 10000 | Enabled | 326.8537 | 132.8574 | 459.7111 |
| 14 | 50000 | Disabled | 0 | 128.9938 | 128.9938 |
| 15 | 250 | Enabled | 111.1723 | 132.8454 | 244.0177 |
| 16 | 10000 | Disabled | 0 | 128.9362 | 128.9362 |
| 17 | 50000 | Enabled | 1216.5840 | 132.8419 | 1349.4259 |
| 18 | 50000 | Disabled | 0 | 129.0063 | 129.0063 |
| 19 | 50000 | Enabled | 1216.0778 | 132.8241 | 1348.9018 |
| 20 | 250 | Disabled | 0 | 128.9197 | 128.9197 |
| 21 | 10000 | Enabled | 326.8305 | 132.8568 | 459.6872 |
| 22 | 10000 | Disabled | 0 | 128.9783 | 128.9783 |
| 23 | 50000 | Enabled | 1216.4369 | 132.8558 | 1349.2927 |
| 24 | 250 | Disabled | 0 | 128.9135 | 128.9135 |
| 25 | 50000 | Enabled | 1216.3544 | 132.8535 | 1349.2079 |

The Keystone Initialization Times for runs with Keystone enabled are illustrated on three scatter plots, separated by the binary sizes, in Figure 24. Additionally, a linear method regression line was fit to the same data points, scaled to base-10 logarithmic, and displayed in Figure 25. The linear method regression model attempts to fit a linear line that best fits the data collected [16] in the form of:

$$Y = \beta_0 + (\beta_1) \cdot X, \tag{21}$$

where $\beta_0$ and $\beta_1$ are the regression coefficients that the model attempts to predict, $Y$ is the response variable Keystone Initialization Time, and $X$ represents the binary size.

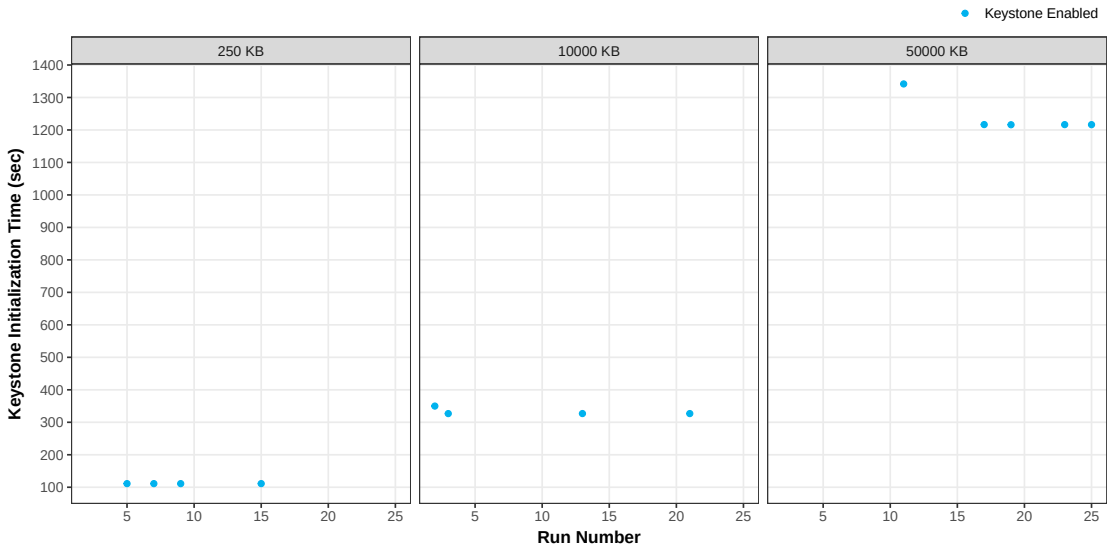



**Figure 24. Scatter plots of Keystone Initialization Time for each experiment run, separated by the benchmark binary sizes.**

In the model, the regression coefficients $\beta_0$ and $\beta_1$ are tested with a t-test to determine if they are statistically significant as predictor variables for the model. If a linear relationship did not exist between the predictor variables and response variable, then $\beta_0$ and $\beta_1$ would be 0. In other words, if changing the binary size did not affect the Keystone Initialization Time, both predictor variables would be 0. The t-test is performed separately for both predictor variables. The null hypothesis $H_0^-$ is that the predictor variables are 0 against the alternative $H_1^-$ that the predictor variables are not 0 [16]. The resulting $p - value$ is compared to the study $\alpha$ to either reject or

fail to reject $H_0^-$.

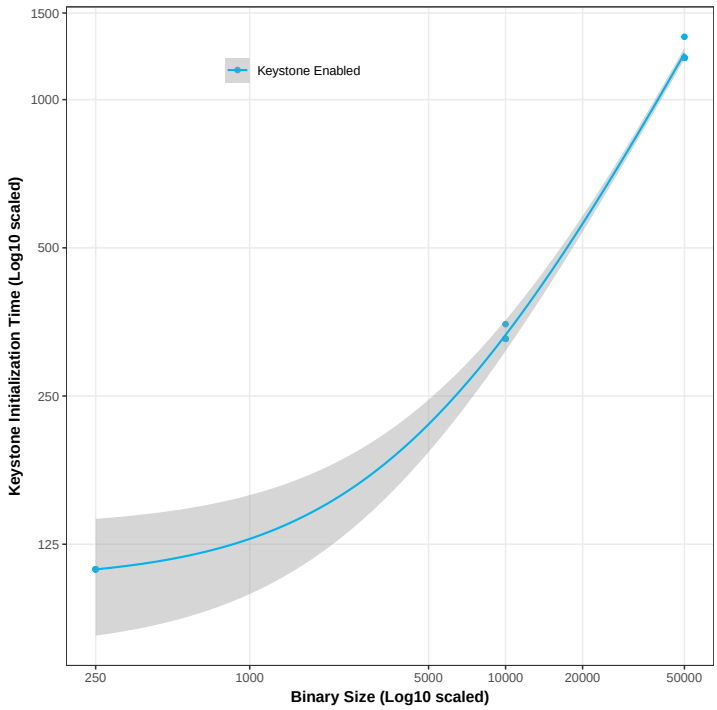



**Figure 25. Linear regression model for Keystone Initialization Time only. Data points are binary size by Keystone Initialization Time, line is linear method regression.**

In Figure 25, the shaded area around the best-fit line represents the confidence band, or list of values the variable can assume. The regression coefficient $\beta_0$ is the y-axis intercept, which in this case is the Keystone Initialization Time and $\beta_1$ is the gradient, or line slope, of the best-fit line. The results of the t-tests for each predictor variable are listed in Table 31. An f-test is also conducted to assess the overall model significance, taking into account both regression coefficients. The null hypothesis is that the fit of the linear model using only the intercept $\beta_0$ is equal to the model with both $\beta_0$ and $\beta_1$. The f-test result is listed in Table 32.

The tests reveal a statistically significant, positive correlation between the size of the benchmark binary and the Keystone Initialization time. Both t-tests on the predictor coefficients rejected $H_0^-$ with a low p-value which indicated the linear

**Table 31. Linear regression model coefficient t-test results.**

| **Predictor Coefficients** | **Estimate (sec)** | **Std. Error** | **t-Value** | ***p*** | **Rej $H_0^-$ at $\alpha = 0.05$?** |
|---|---|---|---|---|---|
| $\beta_0$ (Intercept) | 105.4329 | 13.5346 | 7.79 | 8.41e-6 | Yes |
| $\beta_1$ (Binary Size) | 0.0227 | 0.0004 | 52.88 | 1.36e-14 | Yes |

**Table 32. Linear regression model F-test results.**

| F-Test Results | F-ratio | $p$ | Rej at $\alpha = 0.05$? |
|---|---|---|---|
| Predicted Model | 2796 | 1.362e-14 | Yes |

relationship with the response variable. Additionally, the f-test of the predicted model also rejected $H_0^-$ and the predicted model fit the collected data better with both coefficients compared to a model solely using the intercept $\beta_0$. Furthermore, the tests highlighted a minimum Keystone Initialization Time of 105.4329 seconds, regardless of the binary size of the benchmark. The resulting best-fit line equation for predicting the performance impact that a binary size will have on Keystone is as follows:

$$Keystone\ Initialization\ Time = 105.4329s + (0.0227s) \cdot X, \tag{22}$$

where $X$ is the binary size of the benchmark.

Interestingly, the data observations from the experiment also displayed variation that will necessitate further investigation in future work, but briefly mentioned here. Returning to the scatter plots organized by binary sizes in Figure 24, the first experiment runs for sizes 10000 KB and 50000 KB exhibited higher times than successive runs. While purely speculative, this research believes the slower first-run times are a result of the virtual memory page table passed to Keystone requiring additional time for processing. But, Keystone requires less time on successive runs

of the same benchmark size due to a memory expansion plugin included that helps with development and testing.

### 5.6 Results Summary

This section summarizes the analysis results of all RISC-V embedded system performance characterizations, efficacy of the SPARC framework, and performance impacts of Keystone-specific security features.

- Baseline Performance Characterization Without Keystone: The Rocket to Ariane test found a *relevant difference* in general performance between the systems. Rocket performed better than Ariane on 6 of the 8 benchmarks. Whereas, the Rocket to Shakti test failed to find a *relevant difference* in general performance between the systems. There was a *trivial difference* on 1 benchmark and Rocket performed better than Shakti on 5, but the performance was not enough to reject $H_0^+$.

- SPARC Assessment with Hypothetical Equivalency Margins: The baseline performance characterization tests without Keystone were conducted again, but with hypothetical equivalency margins $[0.75, 1.25]$ and $[0.50, 1.50]$. At $[0.75, 1.25]$, the Rocket to Ariane test found a *relevant difference* in general performance, but one of the benchmarks was found as a *trivial difference*. At the same margin, the Rocket to Shakti test failed to find a *relevant difference* in general performance and a total 6 of 8 benchmarks resulted in a *trivial difference* in performance. At $[0.50, 1.50]$, both Rocket comparisons failed to find a *relevant difference* in general performance. In the comparison to Ariane, there were 6 of 8 benchmarks that resulted in *trivial differences*. Additionally, 7 of 8 benchmarks emerged as *trivial differences* in the Rocket to Shakti comparison.

- Performance Impact of Keystone on Individual RISC-V Systems: On the Rocket RISC-V system, the analysis revealed Keystone impacted system performance by 6.74%, as evidenced by the lower speedup ratio. For Ariane, the test concludes that the impact of Keystone on the system was trivial. In contrast, the Shakti test failed to find evidence that Keystone impacted performance.

- SPARC Framework Efficacy Compared to the HPT Framework: The data from the baseline performance characterization without Keystone was also analyzed with HPT and compared to SPARC. Tests from the HPT methodology failed to find a difference in general performance for both Rocket to Ariane and Rocket to Shakti comparisons. In contrast, SPARC found a difference in general performance for Rocket to Ariane but similar results for Rocket to Shakti. The HPT framework lacks a method to determine equivalency; this resulted in a different Rocket to Ariane outcome with respect to SPARC. Thus, the insights added by SPARC are more effective in statistical inference compared to HPT.

- Performance Evaluation of Keystone-specific Security Features: The Keystone Initialization Time was modeled through a closed queuing network to determine the performance impact Keystone-specific security features had on the Rocket system. Equivalence tests revealed 51.458 seconds as the average Keystone Initialization Time and is *equivalent* for all benchmarks, except Bigint. The output was used to solve the system of equations in the closed-queuing network model and resulted in an average system response time of 72.8 seconds. This is also the performance impact of the Keystone initialization sequence on the Rocket RISC-V system.

- Performance Impact of Binary Size on the Keystone Initialization Sequence: An additional experiment was performed using the benchmark Qsort with varying binary sizes. Linear regression analysis showed a statistically significant positive correlation between the binary size and the Keystone Initialization Time. The linear regression model produced a best-fit line equation for predicting the Keystone Initialization Time based on the binary size as: $105.4329s + (0.0227s)X = \ Keystone\ Initialization\ Time$, where $X$ denotes the binary size.

# VI. Conclusion

## 6.1 Overview

This chapter summarizes the research and results found during the experimental analysis. It reiterates contributions of the statistical framework SPARC and summarizes the observations and analysis of the embedded system characterizations. It closes by listing areas of future work, which include characterizing the performance of FPGAs to ASICs, expanding the SPARC framework with parametric statistics tests, and assessing the impact of all Keystone-specific security features that were not discussed in this research.

## 6.2 Research Contributions

This research was successful in characterizing the performance of a RISC-V embedded system with and without Keystone through the following contributions:

1. Developing the Statistical Performance Analysis with Relevance Conclusions (SPARC) framework.
2. Assessing SPARC's efficacy for establishing a baseline performance characterization of three RISC-V embedded systems.
3. Characterizing the performance impact of the security platform, Keystone.
4. Evaluating hypothetical equivalence margins with SPARC for improved statistical inference.
5. Modeling Keystone-specific security features with a closed-queuing network to determine the average time impact to a system.

6. Predicting the latency added to Keystone's initialization sequence based on an application's binary size.

## 6.3 Summary

The statistical framework SPARC is proposed for a scalable and distribution-free performance evaluation of computers. SPARC identifies superiority or equivalence with hypotheses tests for each benchmark that conditionally result in four relevance conclusions. Through the application of an error correction method in SPARC, Type I error inflation is reduced in multiple benchmark scenarios.

The performance characterization of three RISC-V embedded systems on an FPGA without Keystone indicated that Rocket had better performance than Ariane, but there was no evidence found to support Rocket and Shakti had different performance. Likewise, SPARC revealed Keystone impacts performance on the Rocket system at 6.75% slower speedup ratio. However, Keystone only trivially impacted performance on the Ariane system after all 8 benchmarks tested within the predefined equivalency threshold. On the Shakti embedded system, there was no evidence found that Keystone impacted performance. This does not mean that Keystone had no effect on Shakti, but that the evidence of the benchmarks tested was not enough to reject the null hypothesis.

An assessment of hypothetical equivalency margins was performed with SPARC to determine if the framework can distinguish between systems with similar performance. The analysis highlighted that the SPARC methodology excels in this area. Both of the wider equivalency margins of $[0.75, 1.25]$ and $[0.50, 1.50]$ significantly altered the final performance comparisons in 3 out of 4 tests which failed to find a difference in performance between the systems.

In comparison to the HPT framework, the additional insight from SPARC

provided by relevance conclusions enhances the study results and refines discussion for further experimentation if required. Tests from the HPT methodology failed to find a difference in performance, that was otherwise found using SPARC. The HPT framework lacks a method to determine equivalency between similar performing systems and limits statistical inference.

A high-level model was evaluated for a reference monitor closed-queuing network applied to Keystone on RISC-V architectures. The findings indicated there is an average Keystone Initialization Time overhead to an embedded system, excluding the application binary size, that can be modeled for a single RISC-V system. While this research only evaluated the initialization overhead for creating an enclave, the model provides a foundation to explore the time constraints added when secure execution environments are needed in an operational system.

The binary size of an application was determined to impact performance within the Keystone initialization sequence. Crucially, linear regression analysis identified a positive correlation between binary size and Keystone Initialization Time. The results offer a method for predicting the time added based on the binary size which would provide an additional key attribute for exploring time constraints. Embedded system developers considering to include Keystone in a code update can therefore determine the average system response time, assess if the binary size will increase it, and determine if there is a critical timing path that would be affected by the new code.

In conclusion, the results presented substantiate the original hypothesis: the performance of a RISC-V embedded system with and without a reference (security) monitor can be characterized through statistically rigorous methods. Although the research presented in this thesis relies on the use of an FPGA, the methodologies and framework presented can be extended to other architectures or to ASICs. The

performance characterization data regarding the three RISC-V systems is limited because 1) the processors were instantiated on an FPGA, and 2) the low maturity level of the RISC-V system designs. There is minimal, if any, research that exists comparing the performance of a system on an FPGA to the performance of an ASIC that is an exact replica. Additionally, all three RISC-V systems are continually maturing, with code updates, and performance optimizations. While the performance data observed in this research is months old, there could have been significant updates to either Ariane or Shakti that would result in better performance over Rocket. The performance characterizations presented in this research establish a foundation of statistical analysis to scrutinize system or architectural design changes that affect performance.

## 6.4 Future Work

There are multiple areas that could be further explored, not only in characterizing the performance of RISC-V, but in maturing both the framework and model presented. Listed below are topics of interest that would expand the scope of this research:

1. There is little research that explores a comparison between the performance of a system instantiated on an FPGA, with a replica ASIC. FPGAs have seen an increased amount of use in the past decade, but there were only a few open-source processors that could be used in an FPGA design. The release of the RISC-V ISA has resulted in numerous open-source RISC-V processor designs, knowing which performance attributes carry over to an ASIC would fill the research gap.

2. The performance impacts that Vivado or HDLs have on a system instantiated on an FPGA. An HDL could possibly be translated through a compiler to another

HDL to determine if there is a performance difference between them.

3. Expanding the SPARC framework with parametric statistics tests. The methodology presented non-parametric tests due to the non-normal distributions from the embedded systems. Adding the capability to use parametric tests with the framework would strengthen and expand its availability to a wider audience.

4. Characterizing all Keystone-specific security features on a RISC-V embedded system. Only a fraction of Keystone was tested within this research, there are a number of features that impact performance which will need to be determined for an accurate assessment.

5. Identifying the Linux OS configuration changes that affect the performance data distributions. SPARC was developed for non-parametric statistical analysis, based on the non-normal performance data distributions. While a cluster method was discussed in the related research, more investigation is needed into the factors affecting the embedded system and result in a non-normal distribution.

6. Characterizing the performance of out-of-order instruction execution RISC-V processors. This research only analyzed in-order execution processors to limit variability in performance data. Carrying the characterization to out-of-order instruction execution could identify weak areas within this research and the SPARC framework or provide avenues to explore.

7. Expanding the closed queuing framework to include context-switching between the untrusted operating system and enclave. Context-switching is the most frequent operation during enclave execution; identifying the number of operations and execution time penalty will provide more accurate picture of

the overhead impacting a system. While collecting data for the analysis in Chapter V, code was inserted to capture the operations and clock cycles for a preliminary analysis of context-switching. While the number of context-switches and clock cycles varied between the benchmarks, the impact to execution time was less than 1 ms. Exploring the operation count variability between benchmarks could provide a relationship for additional analysis.

# Appendix A. RV8 Benchmark and Experiment Start Script

runrv8bench.sh

```
#!/bin/bash

# Output logs
#Source test configuration script: TEST_CONFIG.sh
source $TEST_CONFIG

set -e
#Loop through benchmarks in folder
for tst in riscv64/*; do
    tst=$(basename $tst)

    echo "Running $tst"
    for RUN_N in $(seq $REPS); do

        #Setup log files and locations
        BASE_LOG_FILE=${TEST_LOG_DIR}/base_${tst}_${RUN_N}.log
        KEYSTONE_LOG_FILE=${TEST_LOG_DIR}/keystone_${tst}_${
           RUN_N}.log

        #Run the benchmark based on the test config
        #The system without Keystone needs to run a benchmark
        if [[ $RUN_BASELINE == 1 ]]; then
            { time ./riscv64/${tst}; } &> ${BASE_LOG_FILE}
        fi
        #The system with Keystone needs to run a benchmark
        #The three inputs to run a benchmark in a Keystone
        #enclave are below
        if [[ $RUN_KEYSTONE == 1 ]]; then
            { time ${TEST_RUNNER} ./riscv64/${tst} ${
               EYRIE_FULL_SUPPORT} --utm-size ${DEFAULT_USZ}
               --freemem-size ${XLARGE_FSZ} --time ; } &> ${
               KEYSTONE_LOG_FILE}
        fi
    done
done
```

TEST_CONFIG.sh

```
# Test directory names
TEST_FRAMEWORKS="rv8-bench"
#torch beebs iozone"

# Which things to run
RUN_KEYSTONE=1
RUN_BASELINE=1

# Where to stage our binaries/scripts
STAGING_OUTPUT_DIR=$(pwd)/staging

```

```
12 # Number of runs of each thing, short is for torch/iozone
13 REPS=30
14 SHORT_REPS=3
15
16 # Where to store logs
17 LOG_DIR_NAME=logs
18
19 # What is the name of the host bin we use?
20 TEST_RUNNER_NAME=bench-runner.riscv
21
22 # Config untrusted buffer size (K)
23 DEFAULT_USZ=4096
24
25 # Config starting freemem sizes (K)
26 DEFAULT_FSZ=32768
27 LARGE_FSZ=49152
28 XLARGE_FSZ=262144
29
30 # Various eyrie configs
31 EYRIE_FULL_SUPPORT_NAME=eyrie-rt
```

# Appendix B. R Wilcoxon Signed Rank Equivalence TOST Custom Code

```
wilcoxequivalence <- function (x, y = NULL, alternative = c("two.sided"), mu = 0, low_eqbound, up_eqbound, paired = TRUE, exact = NULL, correct = TRUE,
    conf.int = FALSE, speedup = FALSE, conf.level = 0.95, tol.root = 0.0001, digits.rank = Inf, alpha = 0.05,
    ...)
{
    alternative <- match.arg(alternative)
    if (!missing(mu) && ((length(mu) > 1L) || !is.finite(mu)))
        stop("'mu' must be a single number")
    if (conf.int) {
        if (!((length(conf.level) == 1L) && is.finite(conf.level) &&
            (conf.level > 0) && (conf.level < 1)))
            stop("'conf.level' must be a single number between 0 and 1")
    }
    if (!is.numeric(x))
        stop("'x' must be numeric")
    if (!is.null(y)) {
        if (!is.numeric(y))
            stop("'y' must be numeric")
        DNAME <- paste(deparse1(substitute(x)), "and",
            deparse1(substitute(y)))
        if (paired) {
            diffmedian <- (median(x) - median(y))
            if (length(x) != length(y))
                stop("'x' and 'y' must have the same length")
            OK <- complete.cases(x, y)
            if (speedup) {
                x <- x[OK]/y[OK]
                x <- 1 - x
                y <- NULL
                }
                else {
                x <- x[OK] - y[OK]
            y <- NULL
                }
        }
        else {
            y <- y[!is.na(y)]
        }
    }
    if (missing(low_eqbound) && missing(up_eqbound)) {
          stop("missing equivalence bounds")
    }
    else {
        #DNAME <- deparse1(substitute(x))
        #if (paired)
            #stop("'y' is missing for paired test")
    }
    x <- x[!is.na(x)]
    if (length(x) < 1L)
        stop("not enough (non-missing) 'x' observations")
```

```
50      CORRECTION <- 0
51      if (is.null(y)) {
52          METHOD <- "Wilcoxon signed rank test for Equivalence"
53            x1_up_eqbound <- x - up_eqbound
54            x2_low_eqbound <- x - low_eqbound
55
56          ZEROES <- (any(x1_up_eqbound == 0) || any(x2_low_
               eqbound == 0))
57          if (ZEROES) {
58              x1_up_eqbound <- x1_up_eqbound[x1_up_eqbound != 0]
59                  x2_low_eqbound <- x2_low_eqbound[x2_low_
                       eqbound != 0]
60                  }
61          n <- as.double(length(x1_up_eqbound))
62          if (is.null(exact))
63              exact <- (n < 50)
64          r1 <- rank(abs(if (is.finite(digits.rank)) signif(x1_
               up_eqbound, digits.rank) else x1_up_eqbound))
65            r2 <- rank(abs(if (is.finite(digits.rank)) signif(x2
                   _low_eqbound, digits.rank) else x2_low_eqbound))
66            SR1 <- setNames(sum(r1[x1_up_eqbound < 0]), "V")
67          SR2 <- setNames(sum(r2[x2_low_eqbound > 0]), "V")
68          TIES1 <- length(r1) != length(unique(r1))
69            TIES2 <- length(r2) != length(unique(r2))
70          if (exact && !TIES1 && !TIES2 && !ZEROES) {
71              METHOD <- sub("test", "exact test", METHOD,
72                  fixed = TRUE)
73              PVAL1 <- {
74                          p <- psignrank(SR1 - 1, n, lower.tail
                               = FALSE)
75                          min(2*p, 1)
76                          }
77                  PVAL2 <- {
78                    p2 <- psignrank(SR2 - 1, n, lower.tail =
                           FALSE)
79                    min(2*p2, 1)
80                    }
81              if (conf.int) {
82                  x <- x + mu
83                  alpha <- 1 - conf.level
84                  diffs <- outer(x, x, "+")
85                  diffs <- sort(diffs[!lower.tri(diffs)])/2
86                  cint <- switch(alternative, two.sided = {
87                    qu <- qsignrank(alpha/2, n)
88                    if (qu == 0) qu <- 1
89                    ql <- n * (n + 1)/2 - qu
90                    achieved.alpha <- 2 * psignrank(trunc(qu) -
91                      1, n)
92                    c(diffs[qu], diffs[ql + 1])
93                  })
94                  if (achieved.alpha - alpha > alpha/2) {
95                    warning("requested conf.level not achievable
                           ")
96                    conf.level <- 1 - signif(achieved.alpha, 2)
97                  }
98                  attr(cint, "conf.level") <- conf.level
99                  ESTIMATE <- c(`(pseudo)median` = median(diffs)
                       )
```

```
100             }
101         }
102         else {
103             NTIES1 <- table(r1)
104                 NTIES2 <- table(r2)
105             z1 <- SR1 - ((n * (n + 1))/4)
106                 z2 <- SR2 - ((n * (n + 1))/4)
107             SIGMA1 <- sqrt(((n * (n + 1) * (2 * n + 1))/24) -
                    sum(NTIES1^3 -
108                 NTIES1)/48)
109                 SIGMA2 <- sqrt(((n * (n + 1) * (2 * n + 1))/
                        24) - sum(NTIES2^3 -
110                 NTIES2)/48)
111             if (correct) {
112                 CORRECTION1 <- 0.5
113                 METHOD <- paste(METHOD, "with continuity
                        correction")
114             }
115             z1 <- (z1 - CORRECTION1)/SIGMA1
116                 z2 <- (z2 - CORRECTION1)/SIGMA2
117             PVAL1 <- pnorm(z1, lower.tail = FALSE)
118                 PVAL2 <- pnorm(z2, lower.tail = FALSE)
119             if (conf.int) {
120                 x <- x + mu
121                 alpha <- 1 - conf.level
122                 if (n > 0) {
123                   mumin <- min(x)
124                   mumax <- max(x)
125                   W <- function(d) {
126                     xd <- x - d
127                     xd <- xd[xd != 0]
128                     nx <- length(xd)
129                     dr <- rank(abs(if (is.finite(digits.rank))
                            signif(xd,
130                       digits.rank) else xd))
131                     zd <- sum(dr[xd > 0]) - nx * (nx + 1)/4
132                     NTIES.CI <- table(dr)
133                     SIGMA.CI <- sqrt(nx * (nx + 1) * (2 * nx +
134                       1)/24 - sum(NTIES.CI^3 - NTIES.CI)/48)
135                     if (SIGMA.CI == 0)
136                       warning("cannot compute confidence
                              interval when all observations are
                              zero or tied",
137                         call. = FALSE)
138                     CORRECTION.CI <- if (correct) {
139                       sign(zd) * 0.5
140                     }
141                     else 0
142                     (zd - CORRECTION.CI)/SIGMA.CI
143                   }
144                   Wmumin <- W(mumin)
145                   Wmumax <- if (!is.finite(Wmumin))
146                     NA
147                   else W(mumax)
148                 }
149                 if (n == 0 || !is.finite(Wmumax)) {
150                   cint <- structure(c(NaN, NaN),
151                     conf.level = 0)
```

```
152                     ESTIMATE <- if (n > 0)
153                       c(midrange = (mumin + mumax)/2)
154                     else NaN
155                   }
156                   else {
157                     wdiff <- function(d, zq) W(d) - zq
158                     root <- function(zq) {
159                       uniroot(wdiff, lower = mumin, upper =
                             mumax,
160                         f.lower = Wmumin - zq, f.upper = Wmumax
                               -
161                           zq, tol = tol.root, zq = zq)$root
162                     }
163                     cint <- switch(alternative, two.sided = {
164                       repeat {
165                         mindiff <- Wmumin - qnorm(alpha/2, lower
                               .tail = FALSE)
166                         maxdiff <- Wmumax - qnorm(alpha/2)
167                         if (mindiff < 0 || maxdiff > 0) alpha <-
                               alpha *
168                           2 else break
169                       }
170                       if (alpha >= 1 || 1 - conf.level < alpha *
171                         0.75) {
172                         conf.level <- 1 - pmin(1, alpha)
173                         warning("requested conf.level not
                               achievable")
174                       }
175                       if (alpha < 1) {
176                         l <- root(zq = qnorm(alpha/2, lower.tail
                               = FALSE))
177                         u <- root(zq = qnorm(alpha/2))
178                         c(l, u)
179                       } else {
180                         rep(median(x), 2)
181                       }
182                     })
183                     attr(cint, "conf.level") <- conf.level
184                     correct <- FALSE
185                     ESTIMATE <- c(`(pseudo)median` = uniroot(W,
186                       lower = mumin, upper = mumax, tol = tol.
                             root)$root)
187                   }
188                 }
189                 if (exact && TIES) {
190                   warning("cannot compute exact p-value with
                         ties")
191                   if (conf.int)
192                     warning("cannot compute exact confidence
                           interval with ties")
193                 }
194                 if (exact && ZEROES) {
195                   warning("cannot compute exact p-value with
                         zeroes")
196                   if (conf.int)
197                     warning("cannot compute exact confidence
                           interval with zeroes")
198                 }
```

```
199         }
200     }
201     else {
202         stop("Wilcoxon Rank Sum not implemented for
                equivalence")
203     }
204     names(mu) <- if (paired || !is.null(y))
205         "location shift"
206     else "location"
207     if (exact) {
208         z1 <- 0
209         z2 <- 0
210     }
211     if ((PVAL1 <= 0.05) && (PVAL2 <= 0.05)) {
212         CONCLUSION = "H01 and H02: (M1 - M2) outside (-Delta,
                Delta) (nonequivalence) is rejected!"
213       message(paste('H01 and H02: (M1 - M2) outside (-Delta,
              Delta) (nonequivalence) is rejected!'))
214       message(paste('p-value1 =', PVAL1))
215       message(paste('p-value2 =', PVAL2))
216     }
217     if (PVAL1 > 0.05) {
218         CONCLUSION = "Failed to reject H01: (M1 - M2) outside
                Delta (nonequivalence), Therefore, H02 test
                automatically fail to reject'"
219       message(paste('Failed to reject H01: (M1 - M2) outside
              Delta (nonequivalence)'))
220       message(paste('Therefore, H02 test automatically fail to
              reject'))
221       message(paste('p-value1 =', PVAL1))
222     }
223     if ((PVAL1 <= 0.05) && (PVAL2 > 0.05)) {
224         CONCLUSION = "H01: outside Delta (nonequivalence) is
                rejected but failed to reject H02: outside -Delta (
                nonequivalence)"
225       message(paste('H01: outside Delta (nonequivalence) is
              rejected but'))
226       message(paste('failed to reject H02: outside -Delta (
              nonequivalence)'))
227       message(paste('p-value1 =', PVAL1))
228       message(paste('p-value2 =', PVAL2))
229     }
230     RVAL <- list("SR1" = SR1, "SR2" = SR2, "z1" = z1, "z2" =
            z2,
231         "p-value_1" = as.numeric(PVAL1), "p-value_2" = as.
                numeric(PVAL2), "null" = mu,
232         "alternative" = alternative, "method" = METHOD,
233         "data" = DNAME, "conclusion" = CONCLUSION)
234     if (conf.int)
235         RVAL <- c(RVAL, list(conf.int = cint, estimate =
                ESTIMATE))
236     #class(RVAL) <- "htest"
237     RVAL
238 }
```

# Bibliography


1. T. Chen, Y. Chen, Q. Guo, O. Temam, Y. Wu, and W. Hu, "Statistical performance comparisons of computers," *Proceedings - International Symposium on High-Performance Computer Architecture*, pp. 399–410, 2012.

2. MarketsandMarkets, Inc., "Military embedded systems market by component, product type, platform, technology, application, architecture, services and region - global forecast to 2025," may 2020.

3. V. Costan, I. Lebedev, and S. Devadas, "Sanctum: Minimal hardware extensions for strong software isolation," in *25th {USENIX} Security Symposium ({USENIX} Security 16)*, 2016, pp. 857–874.

4. V. Gorbachov, A. K. Batiaa, O. Ponomarenko, and O. Kotkova, "Impact evaluation of embedded security mechanisms on system performance," in *2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T)*. IEEE, 2019, pp. 407–410.

5. A. Dinno, "Comment on "The effect of same-sex marriage laws on different-sex marriage: evidence from the Netherlands"." *Demography*, vol. 51, no. 6, pp. 2343–7, dec 2014.

6. W. W. Tryon and C. Lewis, "An inferential confidence interval method of establishing statistical equivalence that corrects Tryon's (2001) reduction factor," *Psychological Methods*, vol. 13, no. 3, pp. 272–277, 2008.

7. A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovi, "The risc-v instruction set manual. volume 1: User-level isa, version 2.0," CALIFORNIA UNIV BERKELEY DEPT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCES, Tech. Rep., 2014.

8. D. A. Patterson and J. L. Hennessy, "Computer organization and design risc-v edition: The hardware software interface," 2017.

9. A. Waterman and K. Asanovic, "The risc-v instruction set manual, volume ii: Privileged architecture, v1. 12," 2019.

10. D. Lee, D. Kohlbrenner, S. Shinde, K. Asanović, and D. Song, "Keystone: An open framework for architecting trusted execution environments," in *Proceedings of the Fifteenth European Conference on Computer Systems*, 2020, pp. 1–16.

11. V. Costan and S. Devadas, "Intel sgx explained." *IACR Cryptol. ePrint Arch.*, vol. 2016, no. 86, pp. 1–118, 2016.

12. M. T. Khan, D. Serpanos, and H. Shrobe, "A rigorous and efficient run-time security monitor for real-time critical embedded system applications," in *2016 IEEE 3rd World Forum on Internet of Things (WF-IoT)*. IEEE, 2016, pp. 100–105.

13. G. S. Graham and P. J. Denning, "Protection: principles and practice," in *Proceedings of the May 16-18, 1972, spring joint computer conference*, 1971, pp. 417–429.

14. J. Bucek, K.-D. Lange, and J. v. Kistowski, "SPEC CPU2017: Next-Generation Compute Benchmark," in *Companion of the 2018 ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '18. New York, NY, USA: Association for Computing Machinery, 2018, pp. 41–42.

15. R. Dubey, *Introduction to embedded system design using field programmable gate arrays*. Springer Science & Business Media, 2008.

16. A. Field, J. Miles, and Z. Field, *Discovering Statistics Using R*. Sage publications, 2012.

17. M. Hollander, D. A. Wolfe, and E. Chicken, *Nonparametric statistical methods*. John Wiley & Sons, 2013, vol. 751.

18. K. Asanović and D. A. Patterson, "Instruction sets should be free: The case for risc-v," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2014-146*, 2014.

19. V. Gorbachov, A. K. Batiaa, O. Ponomarenko, and E. Kulak, "Securing computer hardware on the base of reference monitor obfuscation," in *2018 International Scientific-Practical Conference Problems of Infocommunications. Science and Technology (PIC S&T)*. IEEE, 2018, pp. 406–410.

20. W. Zheng, S. Cao, Z. Gao, X. Wu, and Q. Ding, "The performance evaluation model of intel sgx-based data protection," in *2018 IEEE SmartWorld, Ubiquitous Intelligence & Computing, Advanced & Trusted Computing, Scalable Computing & Communications, Cloud & Big Data Computing, Internet of People and Smart City Innovation (SmartWorld/SCALCOM/UIC/ATC/CBDCom/IOP/SCI)*. IEEE, 2018, pp. 1289–1292.

21. N. Weichbrodt, P.-L. Aublin, and R. Kapitza, "sgx-perf: A performance analysis tool for intel sgx enclaves," in *Proceedings of the 19th International Middleware Conference*, 2018, pp. 201–213.

22. J. Worms and S. Touati, "Parametric and Non-Parametric Statistics for Program Performance Analysis and Comparison," INRIA Sophia Antipolis - I3S ; Université Nice Sophia Antipolis ; Université Versailles Saint Quentin en Yvelines ; Laboratoire de mathématiques de Versailles, Research Report RR-8875, Mar. 2016.

23. W. Zhang, X. Ji, B. Song, S. Yu, H. Chen, T. Li, P.-C. Yew, and W. Zhao, "Varcatcher: A framework for tackling performance variability of parallel workloads on multi-core," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 4, pp. 1215–1228, 2016.

24. T. Kalibera and R. Jones, "Rigorous benchmarking in reasonable time," in *Proceedings of the 2013 international symposium on memory management*, 2013, pp. 63–74.

25. T. Noergaard, *Embedded systems architecture: a comprehensive guide for engineers and programmers*. Newnes, 2012.

26. R. A. Fisher, *Statistical methods, experimental design, and scientific inference*. Oxford Univ. Press, 1990.

27. F. J. Boster, "On making progress in communication science," *Human Communication Research*, vol. 28, no. 4, pp. 473–490, 2002.

28. T. R. Levine, R. Weber, C. Hullett, H. S. Park, and L. L. M. Lindsey, "A critical assessment of null hypothesis significance testing in quantitative communication research," *Human Communication Research*, vol. 34, no. 2, pp. 171–187, 2008.

29. J. Cohen, "Things i have learned (so far)." in *Annual Convention of the American Psychological Association, 98th, Aug, 1990, Boston, MA, US; Presented at the aforementioned conference.* American Psychological Association, 1992.

30. ——, "The earth is round ($p < .05$)." *American psychologist*, vol. 49, no. 12, p. 997, 1994.

31. D. H. Johnson, "The insignificance of statistical significance testing," *The journal of wildlife management*, pp. 763–772, 1999.

32. P. E. Meehl, "Theoretical risks and tabular asterisks: Sir karl, sir ronald, and the slow progress of soft psychology." *Journal of consulting and clinical Psychology*, vol. 46, no. 4, p. 806, 1978.

33. P. Meehl, "14 what social scientists don't understand," *Metatheory in social science: Pluralisms and subjectivities*, p. 315, 1986.

34. R. A. Fisher, *The Design of Experiments*. Oliver & Boyd, 1960.

35. D. P. Reagle and H. Vinod, "Inference for negativist theory using numerically computed rejection regions," *Computational statistics & data analysis*, vol. 42, no. 3, pp. 491–512, 2003.

36. D. Szucs and J. Ioannidis, "When null hypothesis significance testing is unsuitable for research: a reassessment," *Frontiers in human neuroscience*, vol. 11, p. 390, 2017.

37. C. Bonferroni, "Teoria statistica delle classi e calcolo delle probabilita," *Pubblicazioni del R Istituto Superiore di Scienze Economiche e Commericiali di Firenze*, vol. 8, pp. 3–62, 1936.

38. N. J. Salkind, "Holm's sequential Bonferroni procedure," in *Encyclopedia of Research Design*. SAGE Publications, Inc., 2018, pp. 1–8.

39. S. Chakraborti, B. Hong, and M. A. Van De Wlel, "A note on sample size determination for a nonparametric test of location," *Technometrics*, vol. 48, no. 1, pp. 88–94, 2006.

40. J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*. Routledge, may 2013.

41. G. Shieh, S.-L. Jan, and R. H. Randles, "Power and sample size determinations for the wilcoxon signed-rank test," *Journal of Statistical Computation and Simulation*, vol. 77, no. 8, pp. 717–724, 2007.

42. G. E. Noether, "Sample size determination for some common nonparametric tests," *Journal of the American Statistical Association*, vol. 82, no. 398, pp. 645–647, 1987.

43. E. Walker and A. S. Nowacki, "Understanding equivalence and noninferiority testing." *Journal of general internal medicine*, vol. 26, no. 2, pp. 192–6, feb 2011.

44. S. Wellek, *Testing Statistical Hypotheses of Equivalence*. Chapman & Hall/CRC Press, 2003.

45. D. J. Schuirmann, "A comparison of the two one-sided tests procedure and the power approach for assessing the equivalence of average bioavailability," *Journal of Pharmacokinetics and Biopharmaceutics*, vol. 15, no. 6, pp. 657–680, 1987.

46. C. A. Mara and R. A. Cribbie, "Paired-samples tests of equivalence," *Communications in Statistics-Simulation and Computation*, vol. 41, no. 10, pp. 1928–1943, 2012.

47. S. Feng, Q. Liang, R. D. Kinser, K. Newland, and R. Guilbaud, "Testing equivalence between two laboratories or two methods using paired-sample analysis and interval hypothesis testing," *Analytical and bioanalytical chemistry*, vol. 385, no. 5, pp. 975–981, 2006.

48. C. A. Bellera, M. Julien, and J. A. Hanley, "Normal approximations to the distributions of the wilcoxon statistics: accurate to what n? graphical insights," *Journal of Statistics Education*, vol. 18, no. 2, 2010.

49. P. J. Denning and J. P. Buzen, "The operational analysis of queueing network models," *ACM Computing Surveys (CSUR)*, vol. 10, no. 3, pp. 225–261, 1978.

50. K. Asanovic, R. Avizienis, J. Bachrach, S. Beamer, D. Biancolin, C. Celio, H. Cook, D. Dabbelt, J. Hauser, A. Izraelevitz *et al.*, "The rocket chip generator," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2016-17*, 2016.

51. A. Menon, S. Murugan, C. Rebeiro, N. Gala, and K. Veezhinathan, "Shakti-t: A risc-v processor with light weight security extensions," in *Proceedings of the Hardware and Architectural Support for Security and Privacy*, ser. HASP '17. New York, NY, USA: Association for Computing Machinery, 2017.

52. F. Zaruba and L. Benini, "The cost of application-class processing: Energy and performance analysis of a linux-ready 1.7-ghz 64-bit risc-v core in 22-nm fdsoi technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 27, no. 11, pp. 2629–2640, Nov 2019.

53. R Core Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2020. [Online]. Available: https://www.R-project.org/

54. H. Wickham, *ggplot2: Elegant Graphics for Data Analysis*. Springer-Verlag New York, 2016. [Online]. Available: https://ggplot2.tidyverse.org

55. H. Wickham, L. Henry, T. L. Pedersen, T. J. Luciani, M. Decorde, and V. Lise, *svglite: An 'SVG' Graphics Device*, 2020, r package version 1.2.3.2. [Online]. Available: https://CRAN.R-project.org/package=svglite

56. H. Wickham, "Reshaping data with the reshape package," *Journal of Statistical Software*, vol. 21, no. 12, pp. 1–20, 2007. [Online]. Available: http://www.jstatsoft.org/v21/i12/

57. H. Wickham, M. Averick, J. Bryan, W. Chang, L. D. McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kuhn, T. L. Pedersen, E. Miller, S. M. Bache, K. Müller, J. Ooms, D. Robinson, D. P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, and H. Yutani, "Welcome to the tidyverse," *Journal of Open Source Software*, vol. 4, no. 43, p. 1686, 2019.

58. H. Wickham, R. François, L. Henry, and K. Müller, *dplyr: A Grammar of Data Manipulation*, 2020, r package version 1.0.2. [Online]. Available: https://CRAN.R-project.org/package=dplyr

59. H. Wickham, *tidyr: Tidy Messy Data*, 2020, r package version 1.1.2. [Online]. Available: https://CRAN.R-project.org/package=tidyr

60. J. Clayden, *shades: Simple Colour Manipulation*, 2019, r package version 1.4.0. [Online]. Available: https://CRAN.R-project.org/package=shades

61. E. Neuwirth, *RColorBrewer: ColorBrewer Palettes*, 2014, r package version 1.1-2. [Online]. Available: https://CRAN.R-project.org/package=RColorBrewer

62. J. B. Arnold, *ggthemes: Extra Themes, Scales and Geoms for 'ggplot2'*, 2019, r package version 4.2.0. [Online]. Available: https://CRAN.R-project.org/package=ggthemes

63. H. Wickham and D. Seidel, *scales: Scale Functions for Visualization*, 2020, r package version 1.1.1. [Online]. Available: https://CRAN.R-project.org/package=scales

64. S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3/4, pp. 591–611, 1965.

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704–0188*



| 1. REPORT DATE *(DD–MM–YYYY)* | 2. REPORT TYPE | 3. DATES COVERED *(From — To)* |
|---|---|---|
| 25–03–2021 | Master's Thesis | Sept 2019 — Mar 2021 |

**4. TITLE AND SUBTITLE**

Characterizing Security Monitor and Embedded System Performance across Distinct RISC-V IP-Cores

**5a. CONTRACT NUMBER**

**5b. GRANT NUMBER**

**5c. PROGRAM ELEMENT NUMBER**

**6. AUTHOR(S)**

Tullos, Justin C., Capt, USAF

**5d. PROJECT NUMBER**

21G171

**5e. TASK NUMBER**

**5f. WORK UNIT NUMBER**

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**

Air Force Institute of Technology
Graduate School of Engineering an Management (AFIT/EN)
2950 Hobson Way
WPAFB OH 45433-7765

**8. PERFORMING ORGANIZATION REPORT NUMBER**

AFIT-ENG-MS-21-M-087

**9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)**

Air Force Research Laboratory
2241 Avionics Circle
WPAFB OH 45433-7765
Attn: Pranav Patel
COMM 937-656-9045
Email: pranav.patel.2@us.af.mil

**10. SPONSOR/MONITOR'S ACRONYM(S)**

AFRL/RYDA

**11. SPONSOR/MONITOR'S REPORT NUMBER(S)**

**12. DISTRIBUTION / AVAILABILITY STATEMENT**

DISTRIBUTION STATEMENT A:
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

**13. SUPPLEMENTARY NOTES**

**14. ABSTRACT** Embedded systems have seen a rapid integration into all forms of industry as they continue to shrink in size and cost.The increased demand has highlighted a need for secure systems that are robust to attacks and demonstrate reliable performance, especially if the system operation is time-critical. Efforts to characterize the performance of secure systems have been obstructed either by proprietary restrictions or ineffective analysis. Proprietary technology limits a comprehensive validation of a system's security and the implications it might have on performance. Performance analysis that is disclosed often lacks sufficient statistical rigor needed for a complex system. A non-proprietary processor standard, called RISC-V, may allow sufficient transparency to thoroughly model performance trade-offs. This research shows that a security platform and embedded system performance can be characterized through non-parametric statistics methodology, and provides a substantive foundation to scrutinize system design considerations that impact performance. This work proposes a new framework, the SPARC, that pioneers a synthesis of difference and equivalence hypothesis testing to provide relevant conclusions. SPARC is used to characterize performance of three RISC-V embedded systems with and without a security platform, Keystone, instantiated on an FPGA.

**15. SUBJECT TERMS**

Statistical Performance Analysis with Relevance Conclusions, Field Programmable Gate Array, RISC-V

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT | b. ABSTRACT | c. THIS PAGE | | | Dr. Scott Graham, AFIT/ENG |
| U | U | U | UU | 135 | 19b. TELEPHONE NUMBER *(include area code)* (937) 255-6565 x4581; scott.graham@afit.edu |