CHARACTERIZING AND OPTIMIZING THE PERFORMANCE OF THE MAESTRO 49-CORE PROCESSOR

THESIS

Eric W. Mote, Captain, USAF

AFIT-ENG-14-M-55

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-14-M-55

CHARACTERIZING AND OPTIMIZING THE PERFORMANCE OF THE MAESTRO 49-CORE PROCESSOR

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

Eric W. Mote, B.S.E.E.

Captain, USAF

March 2014

AFIT-ENG-14-M-55

CHARACTERIZING AND OPTIMIZING THE PERFORMANCE OF THE MAESTRO 49-CORE PROCESSOR

Eric W. Mote, B.S.E.E.
Captain, USAF

Approved:

//signed//	11 Mar 2014
Kenneth M. Hopkinson, PhD (Chairman)	Date
//signed//	11 Mar 2014
LTC Robert J. McTasney, PhD (Member)	Date
//signed//	11 Mar 2014
Maj Kennard R. Laviers, PhD (Member)	Date

AFIT-ENG-14-M-55

Abstract

As space-based imagery-intelligence systems become increasingly complex, processing units are needed that can process the extra data these systems seek to collect. However, the space environment presents a number of threats, such as ambient or malicious radiation, that can damage and otherwise interfere with electronic systems. There is a need, then, for processors that can tolerate radiation-induced faults, and that also have sufficient computational power to handle the large flow of data they encounter.

This research investigates one potential solution: a multi-core processor that is radiation-hardened and designed to provide highly parallelized MIMD execution of applicable workloads. A variety of benchmarking programs are used to explore the capabilities of this processor. Additionally, the source code is modified in an attempt to enhance the processor speed and efficiency; the consequent improvements in performance are documented.

Acknowledgements

I would like to thank my faculty advisor, Dr. Hopkinson, for his instruction and encouragement during my research. I also appreciate the guidance offered to me by LTC McTasney and Maj Laviers of my thesis committee.

I am deeply grateful to my interns, Brent Abbey and Dakota Beckelhymer, for their initiative and expertise, as well as the many hours of tedious work they did to configure the MAESTRO board, overcome software-hardware compatibility issues, collect the data, and put the data into a usable format. Without their help, I would not have been able to complete this work.

Thanks also to my sponsor, Nicholas Keller, and his organization for providing the equipment, training, and funding necessary to conduct this research; and to Information Sciences Institute in Arlington, Virginia, for their ongoing technical support.

Eric W. Mote

Table of Contents

Page

Page

List of Figures

Figure Page

Figure Page

List of Tables

List of Acronyms

Acronym	Definition
ASIC	Application-Specific Integrated Circuit
CMOS	Complementary Metal-Oxide-Semiconductor
CUT	Component Under Test
DARPA	Defense Advanced Research Projects Agency
DFT	Discrete Fourier Transform
DSP	Digital Signal Processing
FFT	Fast Fourier Transform
FPGA	Field Programmable Gate Array
FPU	Floating Point Unit
HPL	High Performance Linpack
IMINT	Imagery Intelligence
MCET	Multi-Core Execution Time
MDB	MAESTRO Development Board
MILC	MIMD Lattice Computing
MIMD	Multiple Instruction, Multiple Data
MPI	Message Passing Interface
OPERA	On-board Processing Expandable Reconfigurable Architecture
RHBD	Radiation Hardened By Design
SCET	Single-Core Execution Time
SD-VBS	San Diego Vision Benchmarking Suite
SIGINT	Signal Intelligence
SPEC	Standard Performance Evaluation Corporation
SUT	System Under Test

CHARACTERIZING AND OPTIMIZING THE PERFORMANCE
OF THE MAESTRO 49-CORE PROCESSOR

I. Introduction

1.1 Problem Statement

As space systems become increasingly complex, processing units are needed with enough power to process the extra data that is being collected by these systems. One of the difficulties of collecting data in space is that ambient radiation poses an interference threat to electronic systems, especially digital systems. There is a need for data processing systems that can either tolerate radiation-induced faults, or recover from them, and that also have enough processing power — similar to a corresponding terrestrial system — to handle the large flow of data they encounter.

1.2 Survey of Solutions

One possible answer to this problem is a Field Programmable Gate Array (FPGA) processor. Atmel, for example, manufactures a rad-hard FPGA which is robust against single-event upsets. [1] FPGAs, however, are limited to lower-complexity applications. Another solution is the Application-Specific Integrated Circuit (ASIC), but is likewise problematic due to its processing constraints. ASICs are designed for a single kind of application and lack the versatility needed for general-use processing. A third possibility is a multi-core processor, which can handle high-complexity operations with much more efficiency than the previous options, the tradeoff being the difficulty in hardening a multi-core processor against radiation upsets. Tilera has a family of multi-core processors, one of

which is the TILEPro64 (also referred to as the Tile64). It consists of 64 cores configured in an 8×8 array. [2] These cores can operate in concert or independently.

In 2007, the DoD initiated a program called On-board Processing Expandable Reconfigurable Architecture (OPERA) whose purpose was to use the Tile64 as a starting point and build from it a radiation-hardened processor, ideal for space-based applications. This effort involved, among other things, removing the outer layer of cores, leaving only 49 cores, arranged in a 7×7 array. This multi-core, radiation-hardened system is known as the MAESTRO board. Figure 1.1 illustrates the theoretical (*i.e.,* advertised) difference between OPERA's MAESTRO and the other processing solutions mentioned above.

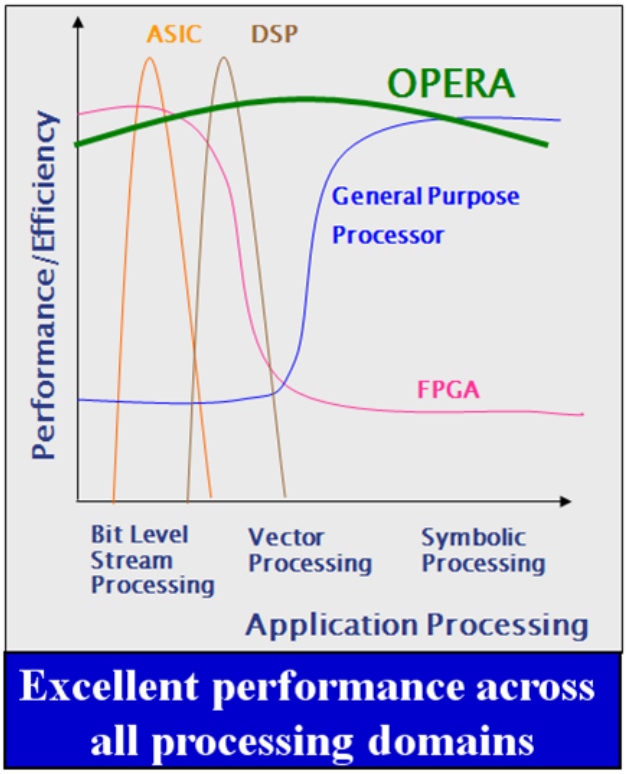


Figure 1.1: Performance comparison of architectures [3]

According to engineers at Jet Propulsion Laboratory, "The OPERA Maestro processor based on the Tilera TILE64 architecture shows potential to give high processing

performance at an error rate equivalent to current space deployed uniprocessor systems." [4] If this is an accurate assessment, then the MAESTRO warrants a thorough investigation into its performance capabilities.

1.3 Hypothesis

The 49-core MAESTRO board exhibits computational performance that is commensurate with that of the Tile64 board, with regard to certain space-based data processing applications.

1.4 Research Objectives

The overarching goal of this research is to characterize the MAESTRO platform by means of benchmarking suites that supply workloads representative of IMINT and SIGINT applications; compare it to its parent, the Tile64 platform; and, if the MAESTRO should prove to be inferior in terms of processing speed and efficiency, determine if there are modifications that can be made to improve its performance as much as possible to match that of the original Tile64 platform.

A secondary goal is to demonstrate that full advantage can be taken of the MAESTRO parallelizable architecture without the need for a significant time investment. That is, the level of required effort is rewarded by a sufficient increase in performance. If high performance is achieved but only at an undue level of effort, the outcome fails to justify the cost.

It is not within the scope of this effort to measure the fault tolerance of the MAESTRO (which has already been examined by other researchers). Rather, the focus here is solely on performance characteristics (*e.g.,* execution times and efficiency).

1.5 Expectations

The expected outcome of this study is that the MAESTRO will exhibit faster execution times and higher efficiency, per number of processors in use, compared to the Tile64,

when highly parallelizable workloads are executed. In particular, workloads involving floating-point operations should be faster on the MAESTRO, because it has a Floating Point Unit (FPU) built in to each core, whereas the Tile64 does not. However, an unparallelized program, or an integer-based workload, should execute at similar speeds on both platforms or perhaps somewhat faster on the Tile64.

1.6 Implications

If the MAESTRO satisfies the expectations described above and can be optimized toward specific functionalities with a reasonable amount of effort, it would prove to be a viable alternative to systems currently being used for space-based data processing.

1.7 Structure

Chapter 2 comprises a review of the literature relevant to this research. Chapter 3 describes the methodology of how the experiment was designed. Chapter 4 presents the results of the experiment. Chapter 5 provides analysis and interpretation of the results and describes the impact and utility of this research.

II. Literature Review

2.1 Overview

This chapter summarizes the relevant literature and is organized in a top-down thought process starting at the broad category of electronic warfare and concluding with the specific details of the processor, how it can be benchmarked, and how it has been utilized so far.

2.2 Electronic Warfare

Electronic warfare capabilities, both offensive and defensive, are increasingly important aspects of the Air Force mission. One of the key targets that offensive electronic warfare seeks to undermine is the intelligence-gathering component, most of which occurs in space by means of satellite data collection and communication. Because two of the biggest space-borne functions that apply to electronic warfare are Signal Intelligence (SIGINT) and Imagery Intelligence (IMINT), directed counter-intelligence attacks seek to disable or disrupt systems performing SIGINT and IMINT operations. [5]

In addition to the logistics of how to get a satellite into position to conduct photo-surveillance, there are complications associated with processing acquired imagery and relaying it as needed to other platforms for further analysis. In some cases, the photographs are physically transported from space back to earth. This is often true with high-quality pictures that occupy such large amounts of digital storage that it is impractical to transmit them through wireless communication channels. On the other hand, pictures that are smaller and lower definition can be relayed wirelessly. [6] If the images can be processed and analyzed by a local high-speed processor on the satellite itself, the relevant information can be sifted out and relayed, rather than forwarding the entire data set which may be quite large. In this way, the transmission cost can be mitigated and unnecessarily large amounts of data are boiled down to what is truly noteworthy.

There is little benefit in obtaining intelligence data if the digital medium on which it is stored cannot survive its intended environment. The space environment, specifically, tends to be hostile to electronic devices. In addition to environmental radiation (such as might be emitted by a solar flare, for example), the mechanics of the satellite vehicle itself can introduce radiation hazards, which are acute at the transistor level. [7, 8] Resistance to electromagnetic interference is therefore a significant consideration when designing or selecting components for use on a satellite.

2.3 Digital Signal Processing

Digital Signal Processing (DSP) is based on the idea that analog signals which are continuous can be represented in a discontinuous, or digital, way such that the essential information is still retained. The advantages of converting to the digital domain, to name just a few, are that digitized signals are simpler; they generally occupy much less electronic storage than do their analog counterparts; they are easily reproduced or copied; and they can be manipulated, filtered, and otherwise processed by means of software rather than hardware (as is necessary for analog processing). For this reason, digital data streams allow for increased flexibility in terms of making changes to the processing tools. They can be altered simply by changing the implementation code and uploading it to the processor.

Another critical element to signal processing is the transformation from a time-domain representation of a signal to a frequency-domain representation. This is important because some operations are quite complicated in one domain but trivial in the other. [9] Time and resources can be saved by pursuing a process which could be described as "transform, process, inverse-transform" to carry out such operations. In this way, three simple steps can be substituted for the single complex step of intense processing. Therefore, algorithms that facilitate these conversions (such as Fourier transforms) from one domain to the other are at the heart of DSP.

According to Oppenheim, the Discrete Fourier Transform (DFT) "plays a central role in the implementation of a variety of digital signal-processing algorithms. This is because efficient algorithms exist for the computation of the DFT." [10] One of these efficient algorithms, and perhaps the most widely used, is the Fast Fourier Transform (FFT). DSP comprises the domain in which IMINT and SIGINT problems exist, and is the primary type of application that will ultimately be running on the MAESTRO.

2.4 Multicore Processing

Both IMINT and SIGINT operations require large amounts of processing power at the chip level. In the last ten years, however, uniprocessor speeds have failed to increase at the same rate they did over the previous decade. [11] This is due to technical constraints that prevent the operating frequency of individual processors from improving in a way that is efficient in terms of power consumption and implementation cost. [11] As an alternative, net processing speeds have been improved by using a larger number of processors simultaneously. However, in order to derive the full benefit of multiple processors, they all must be able to function simultaneously and without dependencies that might cause one to wait on the others. Any amount of time spent waiting on another core's output is wasted and counteracts the desired gains in speed and efficiency. This confluence across cores requires not only physical hardware that can accommodate hierarchical, parallel execution, but also software that can be divided into multiple processing threads.

NASA has expressed an interest in using processors that are not only robust against radiation, but also utilize this multi-core architecture. In the near future, flight computers belonging to both NASA and the U.S. Air Force will rely primarily upon the capabilities of multi-core processors to execute their ever-increasing computational loads for functions like "autonomous landing and hazard avoidance." [12] Higher processing speeds depend on increased parallelization within each chip.

2.5 Parallelization Theory

To derive the benefits from parallelized hardware, the software must be designed in such a way that the workloads assigned to each processor are balanced, and so that each processor is executing independent code that employs thread-level parallelism. In terms of hardware, the microprocessor architecture known as Multiple Instruction, Multiple Data (MIMD) is a critical design structure for taking advantage of thread-level parallelism. [11]

Table 2.1 contains two code sequences that illustrate the difference between non-parallelized and parallelized algorithms.

Table 2.1: Example of non-parallel vs. parallel code

Non-parallelized code	Parallelized code
$b = a + 7$	$s = r + 5$
$c = 3b$	$u = t - 4$
$d = c^2$	$w = log(v)$

The code segment in the left column would not run faster if each line were distributed to three different cores, because the additional cores would remain idle waiting for the previous one to complete its computation. The segment in the right column, however, could be distributed three ways and executed simultaneously because there are no dependencies.

Intrinsic to the software design is the workload itself which also must be parallelizable. For example, matrix workloads that rely on algorithmic techniques such as "divide and conquer" in which the data set can be broken down into many smaller parts without creating output dependencies between the processors are natural applications for parallelized processing.

According to Amdahl's Law, the speedup factor due to parallelized execution is limited by the portion of the code that is actually parallelizable. Figure 2.1 shows the

relationship between the speedup factor and the number of cores in use. [11] Clearly, the percentage of an application that is parallelizable has a considerable effect on how much improvement is possible. If, for example, only 50 percent of the code is parallelizable, the maximum possible speedup factor will be 2, regardless of how many independent cores are employed.

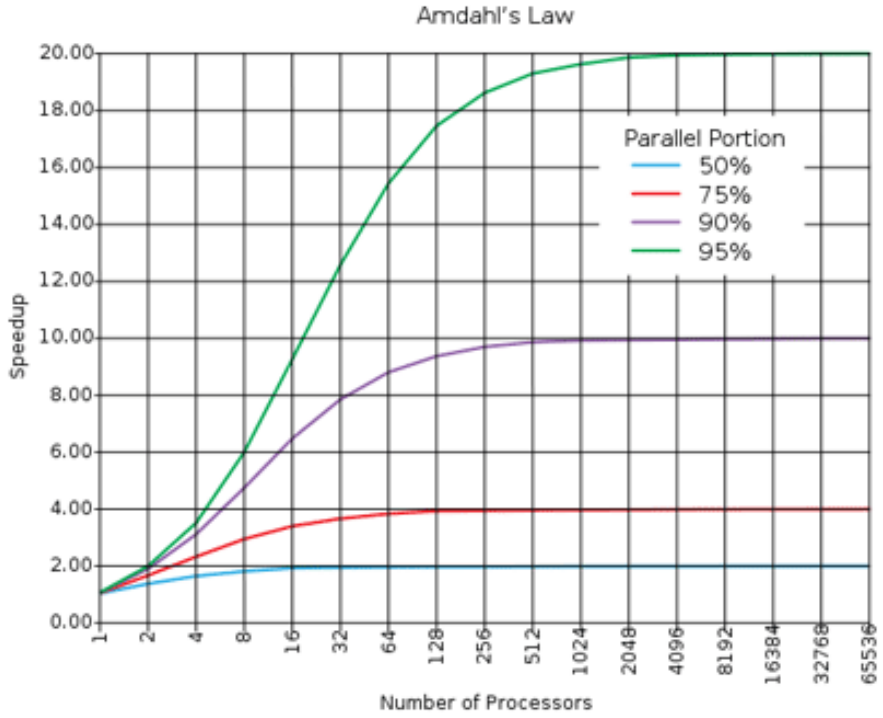


Figure 2.1: Speedup factor per processor (Amdahl's Law)

The algorithms and computational techniques that are necessary for image processing applications are linear filters which facilitate the removal of noise, Fourier transforms which allow for frequency analysis of the image, and geometric and parametric transformations which can provide skew correction and rotation. [13] Such algorithms are most efficiently executed by means of matrix algebra. [14]

2.6 Background of Tile64 & Maestro

The MAESTRO board consists of 49 processors (also referred to as tiles or cores) arranged in a two-dimensional array in a 7×7 configuration. The processors are fabricated using a 90nm Complementary Metal-Oxide-Semiconductor (CMOS) architecture. Each core has its own L1 and L2 cache, as well as a floating point processor which is IEEE 754 compliant for both single and double precision arithmetic. Figure 2.2 shows the layout of the MAESTRO architecture.

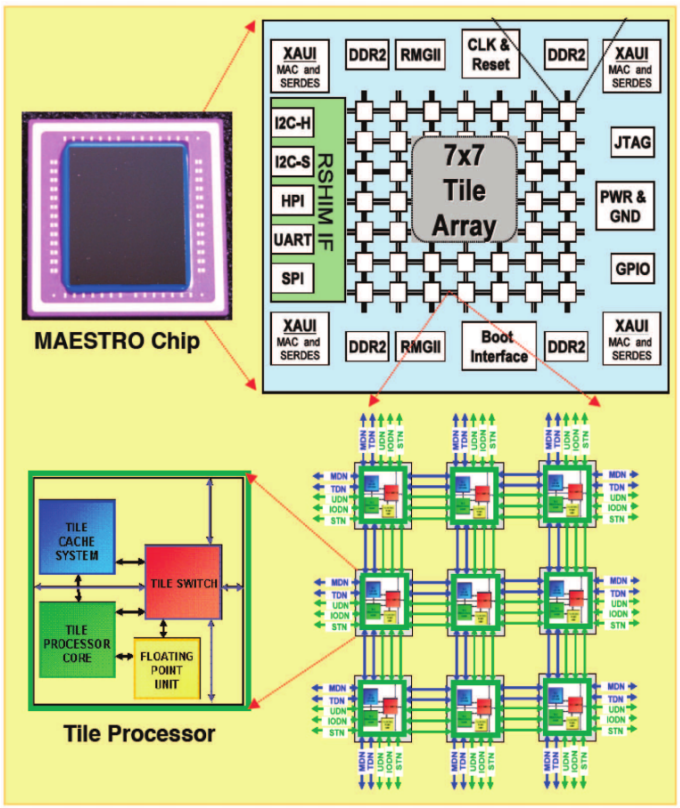


Figure 2.2: Layout of MAESTRO [3]

The MAESTRO board evolved from a commercial product called Tile64 which was created by Tilera Corporation. Defense Advanced Research Projects Agency (DARPA), through its Radiation Hardened By Design (RHBD) program, commissioned Boeing's Solid State

Electronics Development team to fabricate a single processor which would become the building block of the MAESTRO. The OPERA program was then tasked with assembling a board containing a 7×7 array of the radiation-hardened cores. (Figure 2.3 shows the chronological evolution of the technology.)

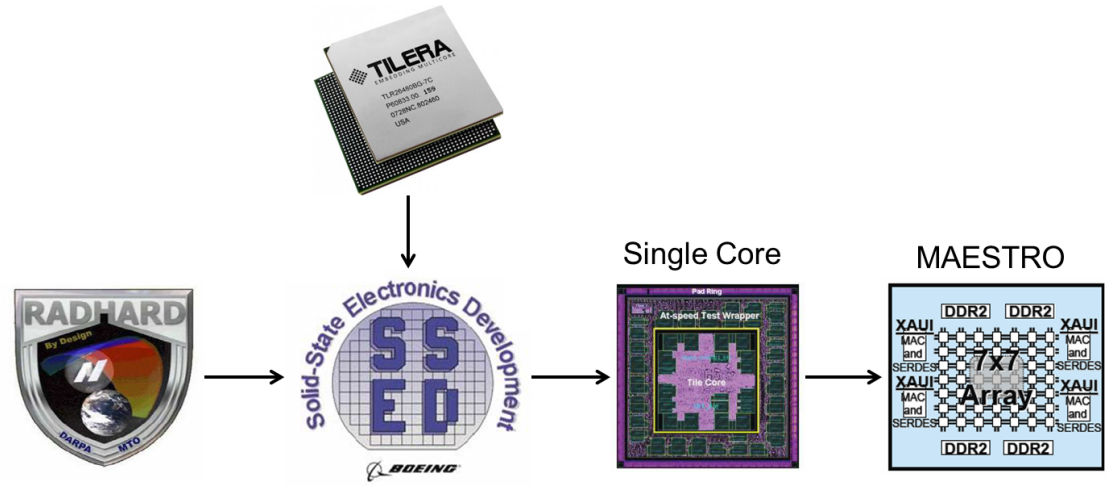


Figure 2.3: Ancestry of MAESTRO [3]

The MAESTRO board was intended for space-based applications that require high-performance processing capabilities, such as image and signal processing. The radiation hardening aids in counteracting the threats that ambient space radiation presents to electronic devices. Latch-up and Total Ionizing Dose radiation effects become less prevalent as the transistor size decreases (see Figure 2.4). However, single event effects, such as memory bit errors, become more likely with smaller transistor sizes. [15] As a result, there is a need for architectures that are immune or resistant to such effects. The MAESTRO was designed in an attempt to satisfy this need.

The most significant difference between the Tile64 and the MAESTRO is the reduction of the number of processors from 64 to 49. Other distinctions include the addition of the FPU with single and double precision capability, and minor changes to MAESTRO's L1

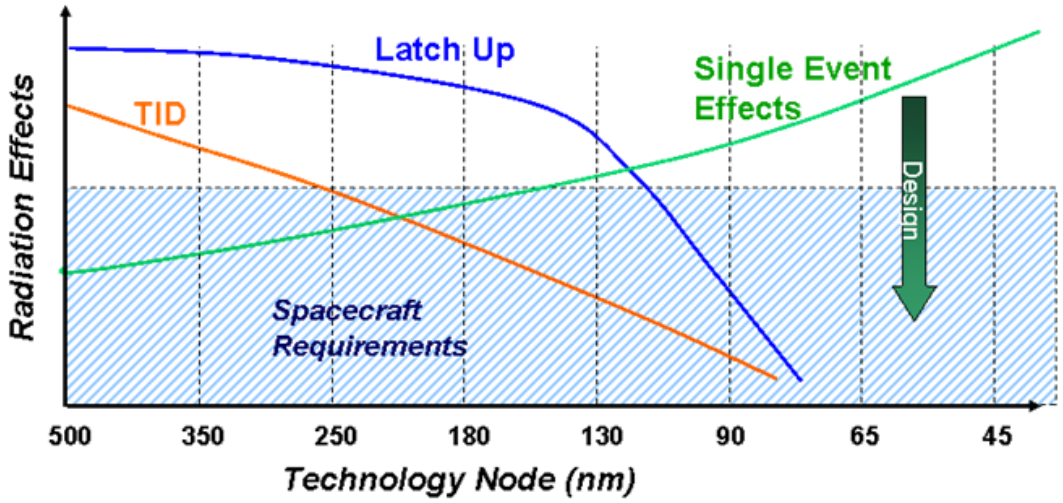


Figure 2.4: Types of radiation-induced faults [16]

instruction cache, its memory built-in self-test, and the DDR memory controllers which are now compatible with both DDR1 and DDR2 memory. [17]

In addition to creating the MAESTRO board, OPERA also produced the MAESTRO Development Board (MDB), pictured in Figure 2.5, to provide the peripherals needed for interfacing with an external computer.

MAESTRO's performance specifications are given in Table 2.2.

2.7 Benchmarking Theory

Benchmarking is an important part of determining the processing capabilities of the MAESTRO board and independently verifying the manufacturer's claims about its performance. One area of interest is how the cache miss rate is affected by various processing schemes. For example, by spreading code execution across several tiles, there may be a notable payoff in performance in that the individual caches of each tile will be less likely to overflow than if only one or two tiles were executing. The unused portion

Figure 2.5: MAESTRO Development Board [16]

of the cache can be gauged by storing and retrieving different amounts of data to memory until the memory access time suddenly increases, indicating a high cache miss rate.

Another item of interest is the performance of specific computational operations that are used heavily in image and digital signal processing algorithms — adaptive filters, finite-impulse response filters, Fourier transforms, and so forth. [18, 19] These can be executed first on a single core to provide a basis measurement. Then the computations can be threaded into a larger number of cores. This information can be presented graphically in terms of "execution time" versus "number of cores in use" for a given input. Theoretically, there is an ideal number of cores that yields a minimized ratio of processing time per number of cores in use. There may also be a case where using too many cores results in

Table 2.2: MAESTRO Specifications [16]

Parameter	**Performance**
Clock Speed (nominal 80C, 1.0V+-5%)	342 MHz
GOPS (Peak) — 32 bit word	50
GFLOPS (Peak) — single or double precision	25
XAUI Interfaces (4) (errata: half duplex)	10 GBPS (each — half duplex)
DDR2 rate (4)	266 MHZ (each)
Power Dissipation (300 MHz, 80C, 1.0V)	21W
Processors (Tiles)	49
Temp. Range	$-40°$C to $125°$C
Area	30.6×25.6 mm
Reliability	100,000 POH
Yield (preliminary)	14%

inordinate overhead costs which exceed the corresponding benefit of the additional cores. However, based on the observations of Dr. Ken Mighell, of the National Optical Astronomy Observatory, of how MAESTRO executed the algorithms he provided, it is likely that the results would be similar for other algorithms. That is, each additional tile (up to at least 45) should provide improved performance that would outweigh the additional overhead cost. [17]

2.8 Previous Work

2.8.1 CRBLASTER.

The MAESTRO board was used in the implementation of a program called CRBLASTER which was designed to import image-analysis algorithms (specifically, Laplacian edge detection) into a parallel-processing environment [20].

Dr. Mighell notes that, while experimenting with CRBLASTER on the MAESTRO board, he experienced problems with MAESTRO during certain long executions of code. The board would crash and have to be rebooted before further operation was possible. It turned out that the board was overheating due to overnight loss of air conditioning in the room where it was running. The heat in the middle of the chip was so intense that it was randomly changing the data stored in memory. Nevertheless, this accumulation of heat did not seem to cause any lasting damage to the board. [20] The MAESTRO uses more than 20 Watts of power and sources a significant amount of heat which needs to be monitored and dissipated for proper functionality. MAESTRO's degraded performance due to excessive heat must be accounted for when considering this platform for use in a space environment, as many cooling systems rely on airflow which is obviously not available in space. [21]

Overall, Dr. Mighell observed a maximum increase of 14 times the single-tile speed when running his algorithm across 45 tiles. In other words, although each core executes at only 342 MHz, he was able to use the MAESTRO at an effective operating frequency of roughly 4.8 GHz.

He further demonstrated a significant improvement in MAESTRO performance by allocating memory through all four of MAESTRO's memory controllers, as opposed to just one. Figure 2.6 shows his results, where the light blue diamonds represent the baseline (unoptimized) algorithm, and the dark blue squares indicate the performance of his user-defined memory allocation scheme.

According to Dr. Mighell, "The Maestro processor definitely has the potential to be an enabling technology for the next generation of U.S. Government satellites and NASA astrophysical missions." [17]

2.8.2 Fault Tolerance.

MAESTRO has also been independently benchmarked for its fault-tolerance and fault-rejection capabilities in a radiation environment. Such research is not immediately

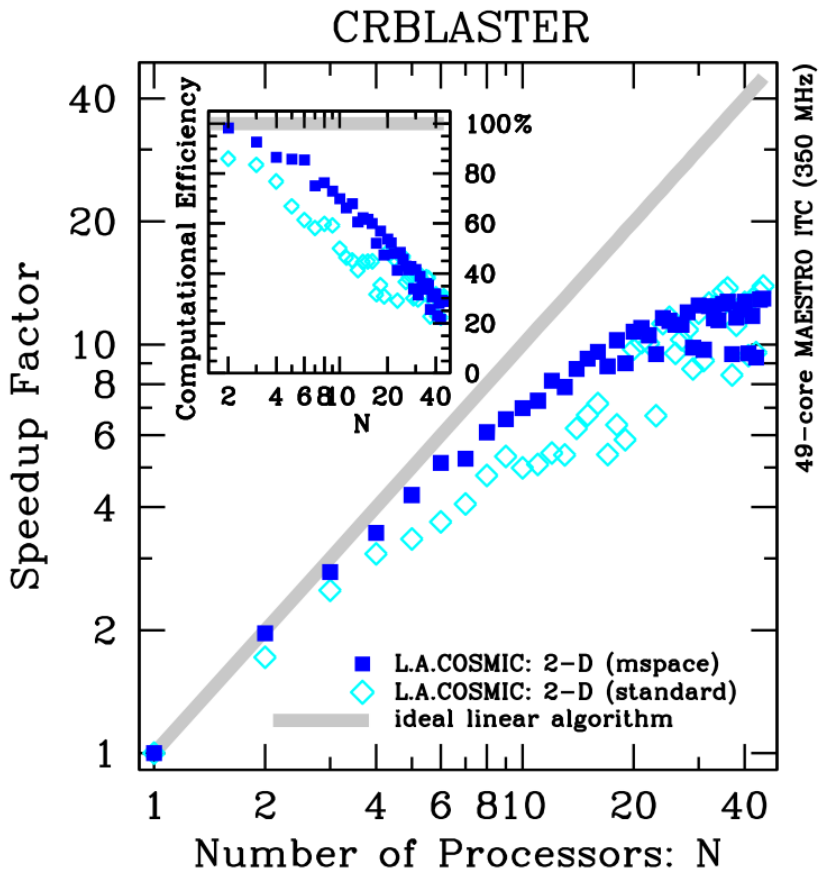


Figure 2.6: CRBLASTER performance optimization [17]

relevant to the present topic, but it has been noted both for sake of completeness and for traceability if any subsequent work seeks to overlay fault-tolerance and performance benchmarking. [22]

2.9 Summary

The literature considered here provides the necessary background information for determining what work has already been done, exactly what remains to be tested, and how to design the experiment so that useful results can be acquired from it.

III. Methodology

3.1 Approach

There are a number of software programs available for benchmarking the performance of a given system. Several of these programs have been selected for use in this research, based on how well they can test the appropriate aspects of the MAESTRO board. Because these benchmarks are software programs, they can be modified and tuned so as to conduct a more thorough experiment. They will be implemented as equitably as possible on both the MAESTRO and the Tile64. Not every suite discussed here is fully implementable on both architectures. Despite this fact, they are listed in full to show that they were considered and attempted, if not actually used.

3.2 System Boundaries

The System Under Test (SUT) is called the MAESTRO Development Board and consists of the MAESTRO multi-core processor and a number of peripherals including power supply, input/output interfacing, and on-board memory. Figure 3.1 shows the boundaries of the SUT.

The Component Under Test (CUT) is the MAESTRO chip, consisting of the 49 processors, the on-board memory, the interface mechanism, and the two-dimensional mesh network that connects the processors. Each processor has a core, a floating point unit, an L1 and L2 cache system, and a tile switch.

3.3 Workload

A variety of benchmarking suites are utilized to examine different aspects of the processing capabilities of each architecture.

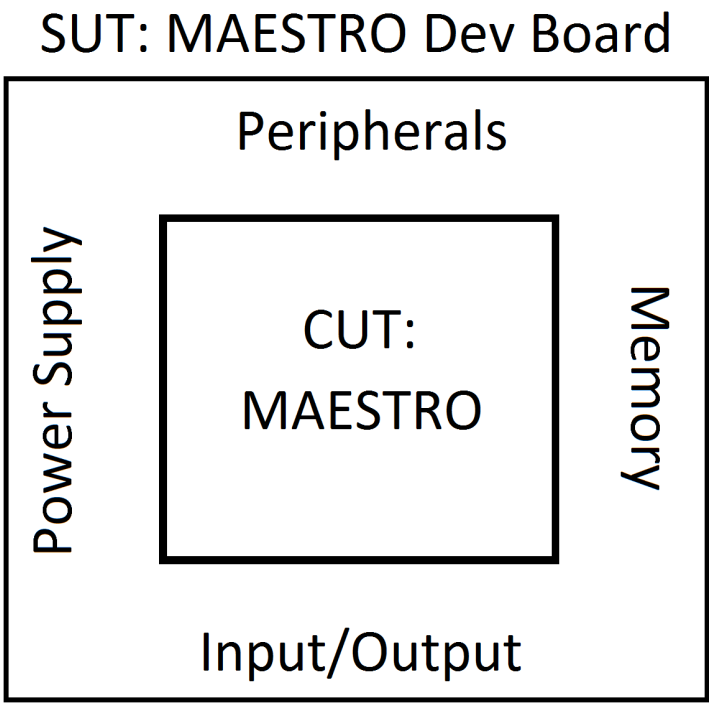


Figure 3.1: System Under Test

3.3.1 MPI Matrix Multiply.

MPI Matrix Multiply (referred to simply as MPI) is a benchmark that measures how much execution time is required for the processor to perform a given matrix multiplication operation. [23] The parameters from this suite that are varied are the input size of the matrix, and the number and configuration of cores that are employed to process the workload.

3.3.2 HPL.

The High Performance Linpack (HPL) suite runs 64-bit, double-precision arithmetic to generate solutions for dense linear systems. HPL measures the amount of time it takes for the architecture to arrive at the correct solution, and computes the accuracy of the calculations. The only active parameter used from this suite is the input size.

3.3.3 SD-VBS.

The San Diego Vision Benchmarking Suite (SD-VBS) tests the single-core capabilities of the host architecture in several different modes that represent vision-oriented applications. (See Table 3.1.) [24]

Table 3.1: Contents of SD-VBS package

Disparity Map	Motion, Tracking and Stereo Vision
Feature Tracking	Motion, Tracking and Stereo Vision
Image Segmentation	Image Analysis
Scale Invariant Feature Transform (SIFT)	Image Analysis
Maximally Stable Regions (MSER)	Image Analysis
Robot Localization	Image Understanding
Support Vector Machines (SVM)	Image Understanding
Image Stitch	Image Processing and Formation
Texture Synthesis	Image Processing and Formation

3.3.4 SPEC MPI2007.

The Standard Performance Evaluation Corporation (SPEC) Message Passing Interface (MPI)2007 suite was designed specifically to exercise the performance of a parallel architecture, as opposed to other benchmarking suites which recycle software that was written for serial execution. According to the SPEC MPI2007 website, the suite tests the following aspects of a system:

- the type of computer processor (CPU),
- the number of computer processors,
- the MPI Library,

- the communication interconnect,
- the memory architecture,
- the compilers, and
- the shared file system. [25]

3.3.5 MILC.

The MIMD Lattice Computing (MILC) benchmarking software was designed for the purpose of studying quantum chromodynamics on parallel processors that are running in an MIMD configuration. This suite is quite complex in terms of both what it measures and how it performs the measurements. Minimal documentation is available to describe the plethora of input parameters or to interpret the metrics on the output. Consequently, MILC was unusable.

3.3.6 OProfile.

OProfile is a Linux profiler that tests L1 and L2 cache miss rates and can be used to identify the specific type of miss rate. This suite does execute on both chips, but it tends to stall unpredictably when running on MAESTRO, so OProfile did not produce meaningful data for this experiment.

3.3.7 BenchFFT.

BenchFFT runs an assortment of FFT implementations in Fortran as well as C. It uses both real and complex transforms with as many as three dimensions. This suite represents typical processes that MAESTRO would execute in an operational scenario. However, MAESTRO compiler problems are to blame for lack of results acquired from this suite.

As specified above, these benchmarking programs contain many parameters that allow for adjustment of the workload submitted to the system, including number of cores to use, adjacency of cores, size of input matrix, not to mention a whole host of compiler options, library flags, and other customizable factors (depending on which suite is in use).

One aspect of the workload that is not intentionally varied as an independent parameter but still important is the level of effort (*i.e.,* number of man-hours) expended to create an ideal workload that is compatible with the MAESTRO compiler. This is important when viewed in conjunction with the output metrics. If the MAESTRO can be optimized to take full advantage of the 49 cores (49x speed-up) but it takes an inordinate amount of time to write the software, the outcome is not useful. The fact that the MAESTRO might generate a 49x improvement in execution speed is noteworthy, but it should be recorded that the corresponding level of effort is enormous.

3.4 Performance Metrics

The system performance is measured by means of the software itself, as opposed to an external entity. When a particular program runs, it takes its own measurements of the internal functionality of the system and reports those to the user upon completion of the task. Not all performance metrics listed are applicable to all the suites.

3.4.1 Execution Time.

Because the goal is to determine the performance capabilities of the MAESTRO board, execution time is an essential metric, as it directly correlates to the throughput capability of the chip. If the processing speed is too slow, the other features of the system cannot fully be taken advantage of and are wasted. Execution time is defined as the time between the beginning of the first clock cycle of the first instruction and the end of the last clock cycle of the last instruction of a given program. [26]

3.4.2 Efficiency.

To determine how well each processor is utilized, an efficiency calculation is made in terms of the Single-Core Execution Time (SCET), Multi-Core Execution Time (MCET), and number of tiles. Suppose a single processor can complete a task in x seconds. If two processors can complete the same task in $0.5x$ seconds, then they are 100 percent efficient.

However, if it takes them $0.6x$, their efficiency would be 83 percent. Equation (3.1) describes this relationship.

$$\text{Efficiency} = \frac{SCET}{MCET * Tiles} \tag{3.1}$$

The efficiency calculation indicates the percentage of the available computing power of each processor that is being taken advantage of for a given task.

3.4.3 Accuracy.

The program can throw specific exceptions during processing to indicate a problem with part of the matrix calculation. If this happens, the program will report that the accuracy of the calculation is compromised. Accuracy is defined as number of correct (verified) calculations divided by total calculations (see Equation (3.2)).

$$\text{Accuracy} = \frac{\text{Correct Calculations}}{\text{Total Calculations}} \tag{3.2}$$

3.4.4 Cache Miss Rate.

Since the concept of cache miss rate is not completely independent of execution time (the former influences the latter), it might seem redundant to take individual measurements of the cache miss rates of the cores after having already measured overall execution time. However, there is still interest in the cache by itself for two reasons: 1) because of uncertainty as to how significantly it influences the execution time; and 2) because, for a specific application, the cache might be likely to encounter high demands from the processor, in which case it is important to know under exactly what circumstances the miss rate starts to climb rapidly. This metric is defined as the number of times the cache does not currently hold the requested data, divided by the total requests to the cache over a given period of time.

3.5 System Parameters

1. Electromagnetic interference (ambient). There are several active cell phones and a microwave oven in the room where MAESTRO is tested. The interference from them is assumedly negligible with respect to the performance metrics of this research.

2. Board temperature. MAESTRO uses up to 25 Watts of power and produces a significant amount of heat. Unless dissipated by the on-board fans, the heat does cause the processors to fail. [20] MAESTRO is intended for use in space where cooling by air convection is not possible, so some other means of cooling is required when the system is deployed.

3. Other active system processes competing for resources. During this experiment, the number of extraneous processes running on MAESTRO is not varied. However, in an operational environment where there is competition for resources, this parameter affects both execution time and cache miss rate since some of the resources may be allocated to other programs.

4. Number of cores in use (workload parameter). If the code being executed is parallelizable, then the number of cores assigned to execute the program is correlated to the execution time. The system performance is highly sensitive to this parameter.

5. Adjacency of cores in use (workload parameter). MAESTRO executes processes faster if the active cores are physically near each other because the communication between the cores takes less time.

6. Size of input matrix (workload parameter). The volume of data supplied to the system affects how much memory is available at run time.

7. Compiler optimization level (workload parameter). The compiler has three optimization levels (0, 1, 2). In this experiment, the compiler uses level 2 exclusively,

which corresponds to the highest optimization level (as opposed to level 0 which is the lowest).

8. Compiler static library flag (workload parameter). This flag tells MAESTRO whether to look in cache or in RAM for the necessary libraries. When this flag is set to 1, more cache storage is used so as to improve execution time. When this flag is set to 0, execution time is slower but cache space is conserved. For this research, the flag is always 1.

3.6 Factors

1. Number of cores in use. The number of cores is varied to test how much benefit is achieved by adding more cores to the execution framework. This is the most important factor because it exercises the critical aspect of the MAESTRO architecture: parallelizability. Software designed to utilize only a single core executes much slower than does software that is correctly designed to utilize many cores. If code is written to take advantage of parallelization, two cores are faster than one, and three cores are faster than two, given inherent parallelism is available in the task. Also varied within this factor is the shape of the core conglomeration. For example, 6×7: 40 yields a different shape of cores than does 7×7: 40. (See Table 3.2 for an itemized list of configurations to be used.)

2. Size of input matrix. Since the purpose of a multi-core processor in a functional scenario is to process large volumes of data, it is necessary during testing to vary the dimensions of the inbound data matrix to determine what effect this has on the metrics of interest.

3.7 Evaluation Technique

This research utilizes a direct measurement evaluation technique on two different systems: the Tile64, and the MAESTRO.

Table 3.2: MPI Core Configurations

1×1: 1	5×5: 25
2×2: 2	5×6: 30
2×2: 4	6×6: 30
3×3: 8	6×6: 36
3×3: 9	6×7: 40
4×4: 10	7×7: 40
4×4: 16	6×7: 42
4×5: 20	7×7: 45
5×5: 20	

The MAESTRO board (version 1.0) has a native Linux kernel, 1GB of memory, and is connected via Ethernet (100 Mb/s) to a computer running the RedHat Enterprise Linux operating system (OS), version 5.9. From the external Linux system, a particular benchmarking suite is loaded onto the MAESTRO which then compiles the code locally on the SUT.

The parallelizability of the MAESTRO is examined by running a given program on a single core, then spreading that same program across two cores, then four cores, and so forth. As more cores are added, the execution time improves (decreases) until the point at which the overhead of using additional cores outweighs or counterbalances the benefit of their use. There should be a knee in the curve representing the diminishing marginal utility of employing additional cores.

The various benchmarking suites utilized in this research can take anywhere from a few seconds to several hours to finish a single run. Upon completion of the program, the benchmark software outputs the resulting measurements to the user. These are recorded for analysis.

The output metrics on the MAESTRO and the Tile64 are validated by comparing them to each other and observing that the difference between them is within reason, especially under circumstances in which the two systems are expected to generate similar results. Further validation is performed by comparing the measurements to theoretical expectations of a best-case scenario. For example, given a known number of operations to be executed and the specific clock speed of the processor, the execution time of the program can be roughly predicted.

The collected data is then analyzed using the R statistical software language, from which boxplots, barplots, and other graphs are generated to depict the findings.

3.8 Experimental Design

A full factorial design is not feasible due to lack of pre-existing benchmarking suites that allows all factors to be varied independently. Therefore the experiment consists of a partial factorial design.

For the MPI suite, there are 17 variations on how the cores will be configured. The three arguments that describe the configuration are number of rows, number of columns, and number of cores. For example, 5×5:22 implies 5 rows, 5 columns, and 22 cores selected (the maximum available in this case would be 25).

There are 17 different configurations and 30 repetitions of each, each boxplot on the graph represents 510 executions. Table 3.3 specifies a sample data collection matrix for the MPI suite. Similar matrices are used for the other suites as applicable. An example of how the cores are assigned is shown in Figure 3.2.

These levels are chosen so as to generate a meaningful data trend across the full spectrum of cores while keeping the complexity of the experiment as low as possible. The confidence interval is set at 95 percent, as this is a generally accepted confidence interval. [26]

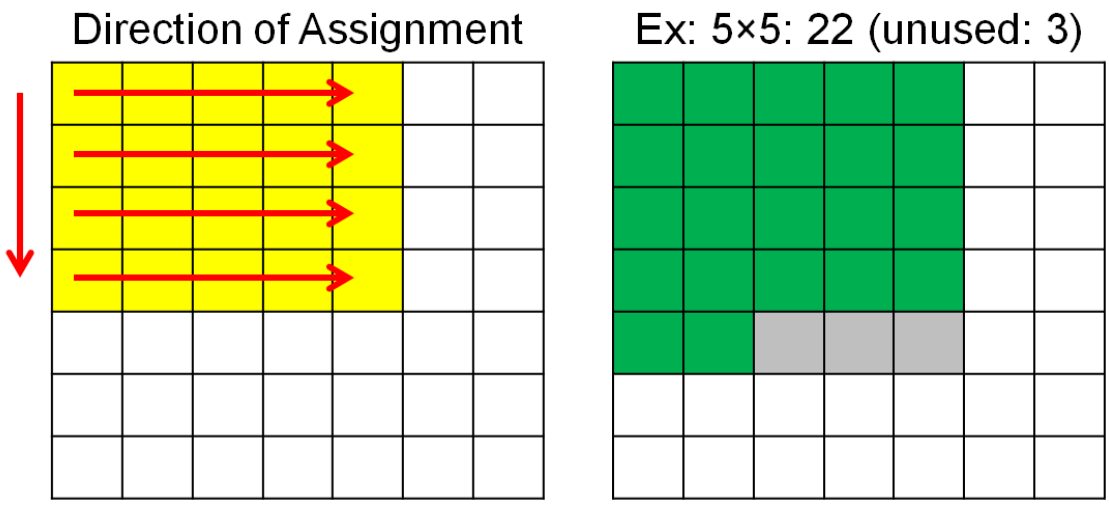


Figure 3.2: Sequential core assignment

3.9 Methodology Summary

The expected outcome of this study is that the MAESTRO will exhibit superior performance, per number of processors in use, compared to the Tile64 when highly parallelizable workloads are executed. In particular, workloads involving floating-point operations should be faster on the MAESTRO, because it has a FPU, whereas the Tile64 does not. However, an unparallelized program, or an integer-based workload, should execute at similar speeds on both platforms or perhaps somewhat faster on the Tile64. Several benchmarking suites are used to test the performance of the parallelized architecture in terms of program execution time, core efficiency, and calculation accuracy. The workload submitted is varied by changing the number of cores MAESTRO uses to run a program and the size of the input matrices that the cores operate on. Output values are validated by comparing them to theoretical expectations and observing similar experimental results, within reason, between the two architectures. A partial factorial design yields sufficient data to analyze MAESTRO's performance.

Table 3.3: Data collection matrix for MPI

		Input Size						
# Cores	Metrics	60	125	250	375	500	625	750
1	Execution Time							
	Efficiency							
2	Execution Time							
	Efficiency							
4	Execution Time							
	Efficiency							
8	Execution Time							
	Efficiency							
⋮								
45	Execution Time							
	Efficiency							

IV. Experimental Results

4.1 MPI

4.1.1 Baseline Characterization (Version 1).

Figure 4.1 shows the MCET versus the input size of the matrix for all configurations simultaneously, with the Tile64 results displayed at the top and the MAESTRO results at the bottom. Because there are 17 different configurations and 30 repetitions of each, each boxplot on the graph represents 510 executions. As the input size grows from a 60×60 matrix to a 750×750 matrix, the execution time also increases. This is due to the fact that there is more data to process, and the processors require more time to complete the computations.

Figure 4.2 illustrates how the execution time changes with respect to the number of processors that are used during the computation, and across all input sizes simultaneously. The general trend of the graph demonstrates the expected behavior: that adding more processors improves the execution time according to an exponential decay function.

Figure 4.3 shows how the efficiency is affected by input size. In general, it increases with the size of the input and levels off at some upper boundary.

Figure 4.4 shows that processor efficiency diminishes as more and more processors are added. This is the expected behavior, because the work cannot be divided into an infinite number of pieces that execute simultaneously. At some point, there is no added benefit to having more processors, namely, when the additional overhead needed to include them in the computation equals or exceeds the time they save on the overall computation.

Figure 4.5 is a plot of the 98 p-values representing the Tile64 vs. MAESTRO MCET comparison for every combination of input size and number of processors. The red horizontal line marks the standard 0.05 significance level. Any value below that line indicates high confidence that the data sets under comparison are statistically different. Of

the 98 data sets, there is only one whose p-value rose above the 0.05 level. When the input matrix size was 60×60 and 16 processors were used, the Tile64 and MAESTRO did not exhibit statistically different behavior. In all other test configurations, however, the two boards were clearly different.

Figure 4.6 shows the relative boxplots of the cumulative MCET data for all input sizes and configurations of each board. The p-value for the t-test between them is $1.18 \cdot 10^{-54}$ indicating that the two boards produced statistically different results.

4.1.2 Code Optimizations.

Several optimizations were made to the code in order to derive the full benefit of the MAESTRO capabilities. A summary of the changes that constitute the various versions of the code from Version 1 to Version 5 are given below.

Version 1:

- Added memory controller functionality. (Figure 4.7 illustrates how memory-tile quadrants are allocated to individual memory controllers.)
- Matrices are now stored via mallocs.
- Makefile has more power.

Version 2:

- Renamed the program and moved it to the source directory.
- Split the program into multiple .c files.
- The master CPU no longer broadcasts unneeded data.
- All memory controllers are being used based on how many processes are being run and the location.

Version 3:

- For situations in which the matrix size per number of tiles is a non-integer, the remainder portion is now handled by the first slave CPU, rather than by the master CPU.
- Added more functions to the program to improve functionality.
- Added OProfile targets to the makefile.

Version 4:

- Added timer functionality so that each CPU can report timed results of individual portions of the program.
- Updated the makefile.

Version 5:

- The calculate function no longer calls getters and setters.
- Each CPU now looks for data in its own cache before querying memory (notable performance improvement).

4.1.3 Optimized Characterization (Version 5).

One feature of Version 5 that was not available in Version 1 is the ability to see the functionality of the individual tiles, and to observe the amount of time they spend on particular tasks, namely, allocating memory, sending and receiving data, and calculating data. Figure 4.8 through Figure 4.13 illustrate this capability. The overall program does not finish until all cores have completed their tasks, so the slowest core determines the summary execution time as reported by the benchmark. These graphs demonstrate that MAESTRO is indeed faster for input sizes 125 and 250 (because its slowest core completes faster than Tile64's slowest core), but slower for the remaining (larger) sizes.

4.1.4 Analysis.

Clearly some execution speed and efficiency was lost in the initial creation of MAESTRO from the Tile64. However, MAESTRO can be made superior to Tile64, at least for small input sizes. Figure 4.15 displays a heat-map comparison of the median execution times for the two boards for each variation in input. This map is based on the median of the difference in MCET. Orange/red squares denote MAESTRO superiority. White squares indicate equality between the boards. Blue squares imply Tile64 superiority.

The fact that the Version 5 heat map displays more red squares than that of the Version 1 demonstrates that MAESTRO was, in fact, improved for the smaller input sizes without sacrificing its relative performance in the larger input sizes.

Equation (4.1) describes the formula used to calculate the data shown in Figure 4.14. A smaller MCET value implies faster execution, so as the data on Figure 4.14 trends lower, MAESTRO is improving. As data trends higher, MAESTRO performance is diminishing relative to Tile64. The red horizontal line denotes zero. When the data is below that line (*i.e.,* negative), MAESTROis exceeding Tile64. When data crosses above zero, the Tile64 is excelling instead.

$$\frac{\text{median(MAESTRO MCET)} - \text{median(Tile64 MCET)}}{\text{median(Tile64 MCET)}} \tag{4.1}$$

4.2 HPL

Due to a limitation on the number of graphs allowed in a single chapter by the program being used to publish this document, and because the HPL results are not directly analyzed in this chapter, the HPL graphs have been moved to Appendix A. Figure A.1 and Figure A.2 depict the GFLOPS as a function of input size and number of cores, respectively. The Tile64 outperforms the MAESTRO, which is unexpected behavior. Because the

MAESTRO has a FPU and the Tile64 does not, the former should be able to execute many more FLOPS than the latter. No optimizations were performed on the HPL suite because it is not easily modified. Potentially, this lack of optimization cripples the MAESTRO's functionality by making poor use of its memory architecture and therefore introducing a bottleneck that inhibits its actual computational performance.

Figure A.3 and Figure A.4 describe the accuracy of calculations as compared to input size and number of cores. As input size increases, the accuracy rate decreases. The same is true as the number of cores increases, but to a lesser extent.

Figure A.5 and Figure A.6 portray the execution times. The results appear similar to the baseline, unoptimized execution times of the MPI suite, in that Tile64 exhibits faster performance.

4.3 SD-VBS

No optimizations were attempted on the SD-VBS benchmark, as it exercises only a single core at a time and therefore does not provide much insight into the multi-core functionality of MAESTRO. The characterization graphs are given in Appendix B (Figure B.1 through Figure B.16).

4.4 Incompatible Suites

The MILC suite is quite complex in terms of what it measures and how it performs the measurements. Minimal documentation is available to describe the plethora of input parameters or to interpret the metrics on the output. Consequently, MILC was unusable.

OProfile does execute on both chips, but it tends to stall unpredictably when running on MAESTRO, so this suite did not produce a complete set of data for this experiment.

MAESTRO compiler problems are to blame for lack of results acquired from the BenchFFT suite. Perhaps this issue will be surmountable in future research.

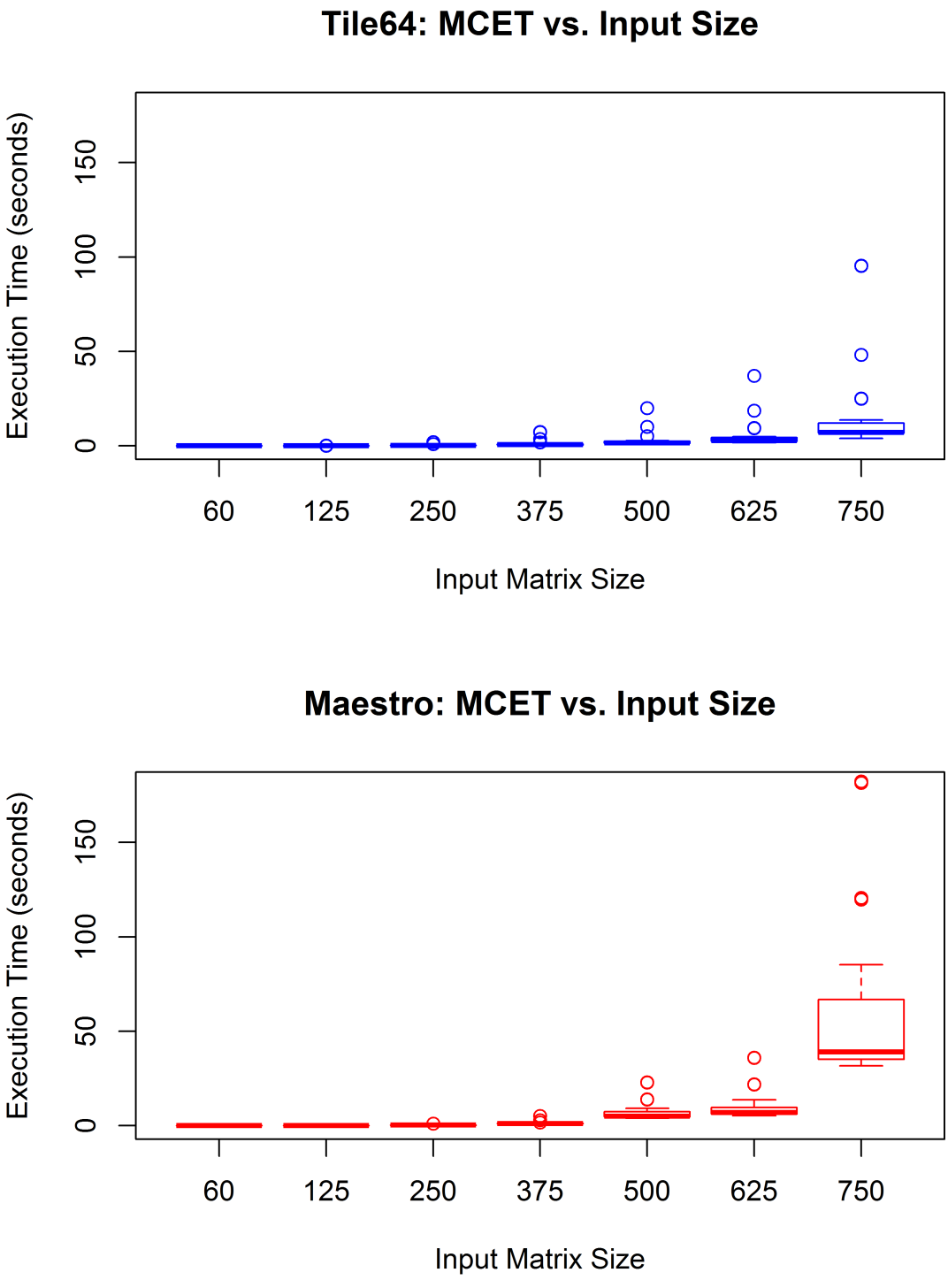


Figure 4.1: Version 1: Execution Time per Input Size

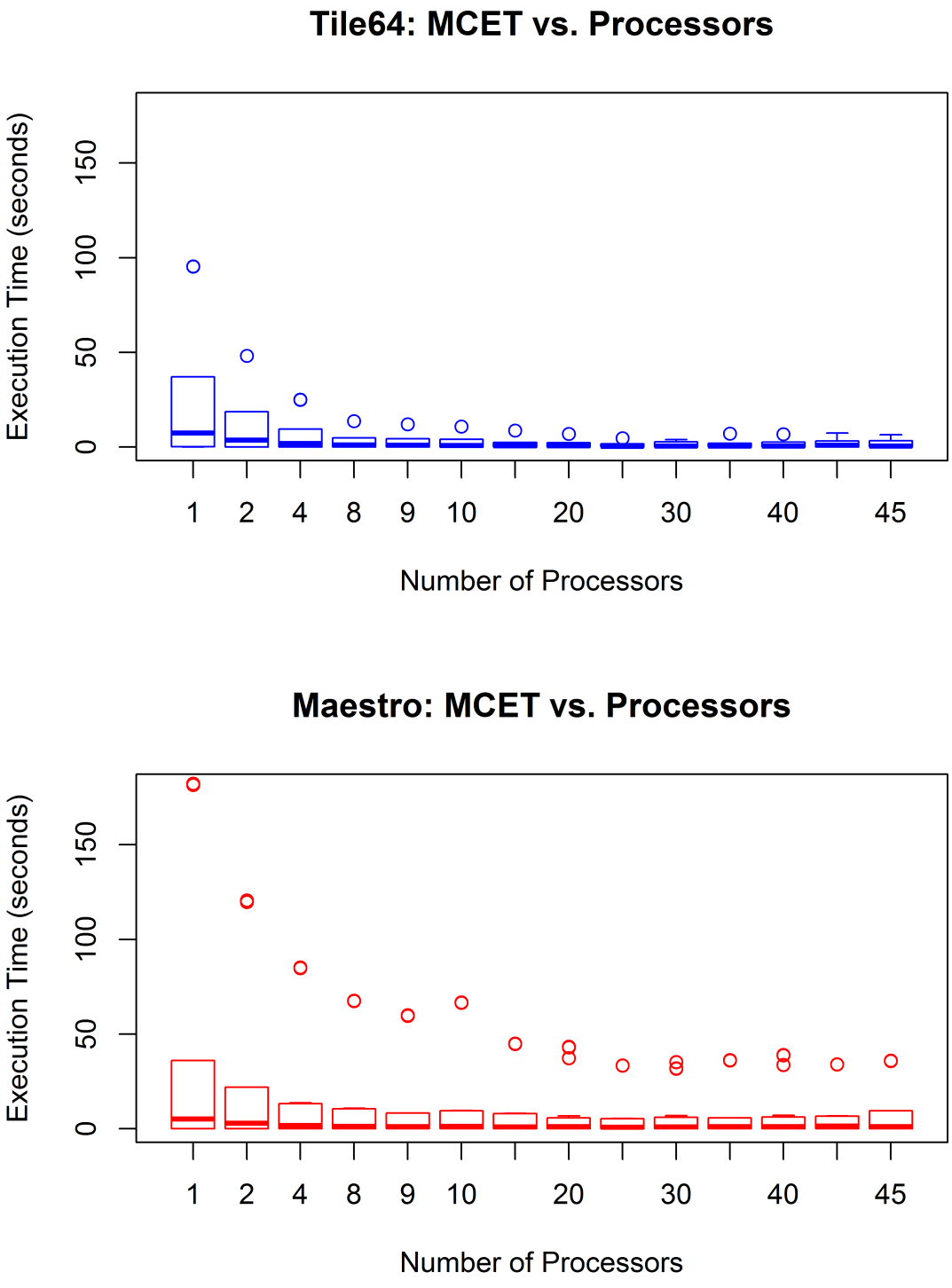


Figure 4.2: Version 1: Execution Time per Processors in Use

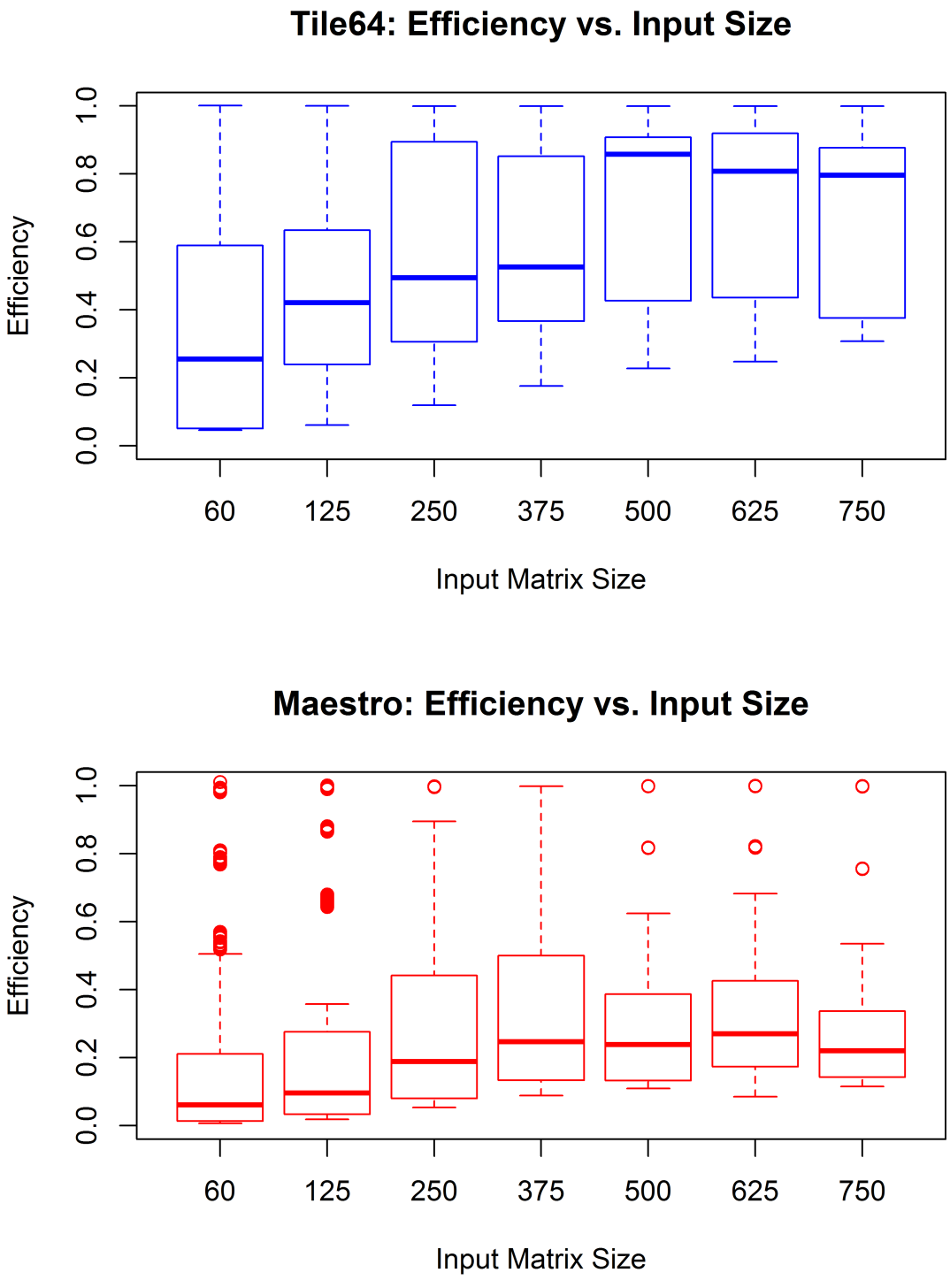


Figure 4.3: Version 1: Efficiency per Input Size

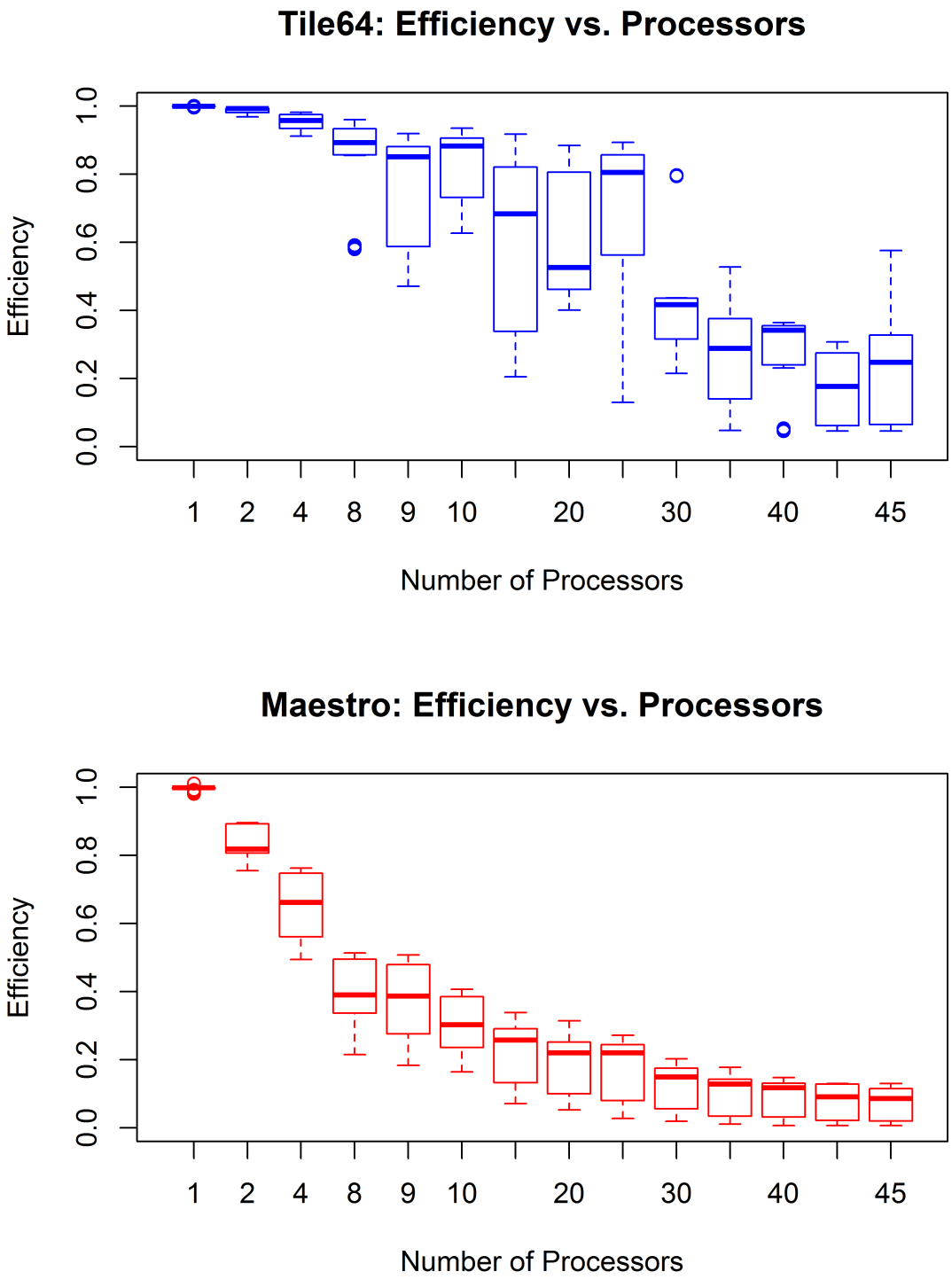


Figure 4.4: Version 1: Efficiency per Processors in Use

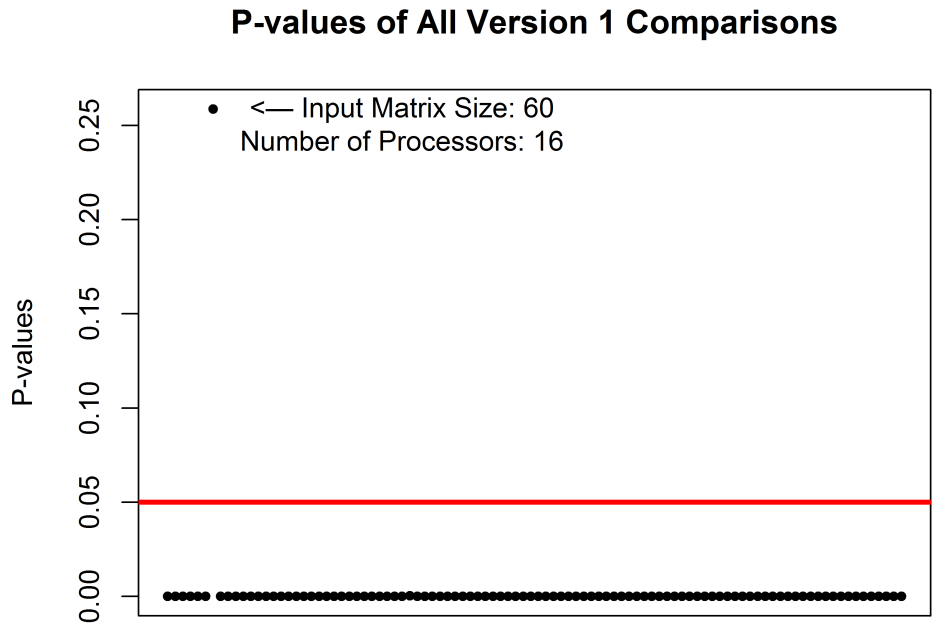


Figure 4.5: Version 1: Individual p-values for Tile64 vs. MAESTRO comparisons

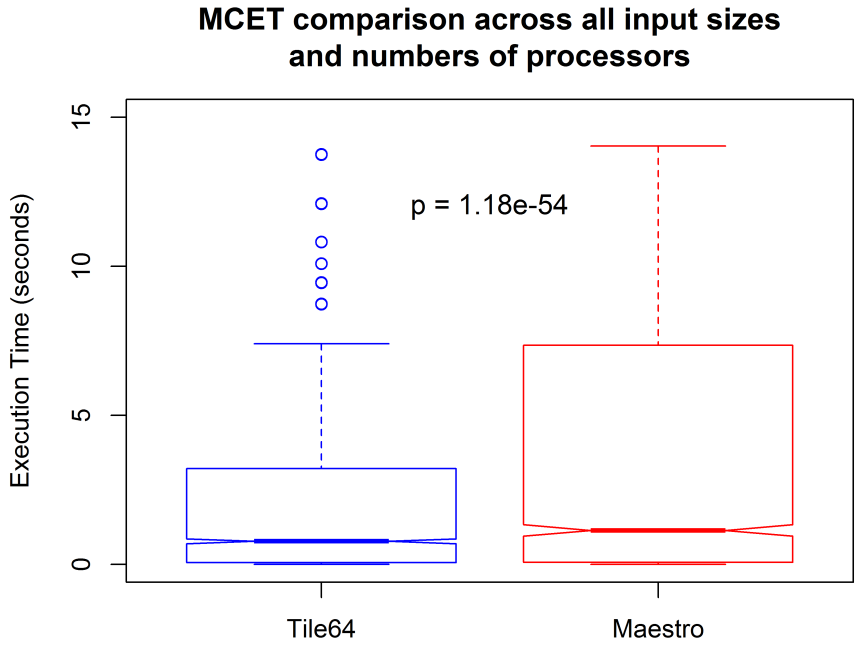


Figure 4.6: Version 1: Conglomerate p-value comparing Tile64 vs. MAESTRO

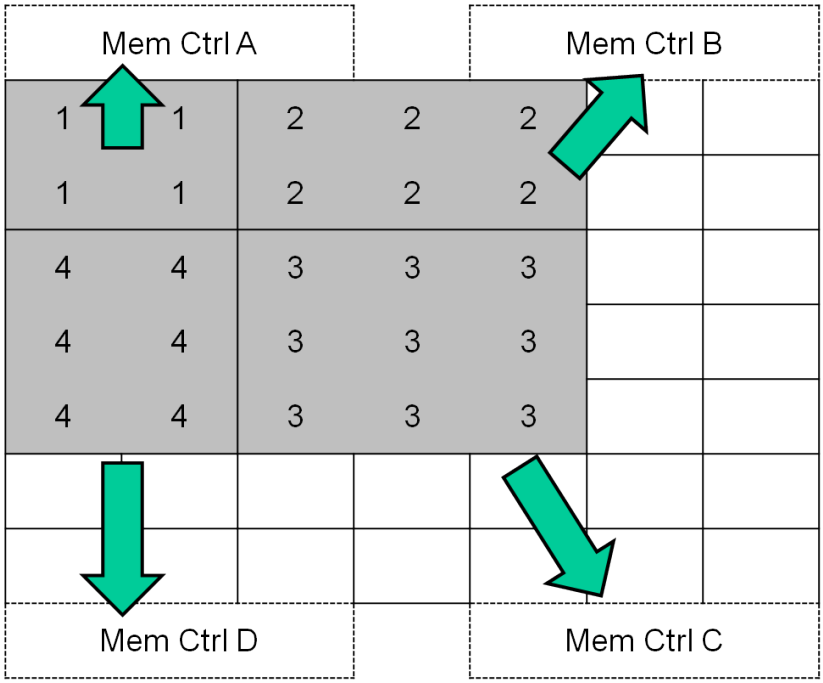


Figure 4.7: MAESTRO memory controller assignment example

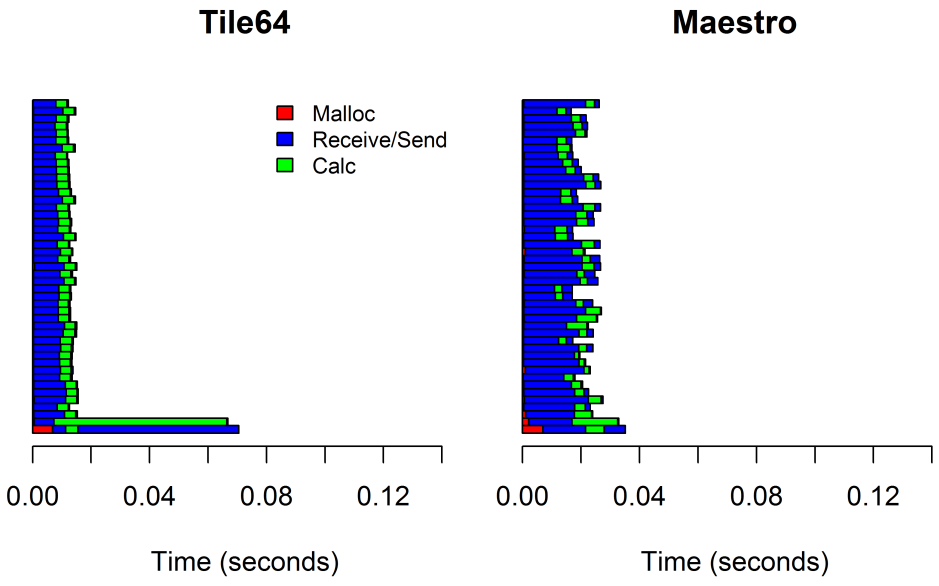


Figure 4.8: Version 5: tile execution times; input size = 125

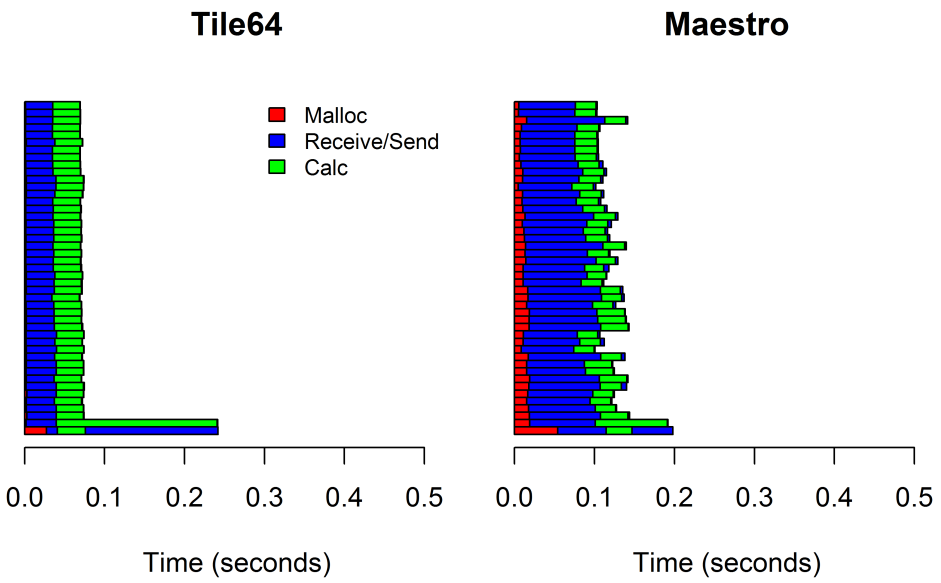


Figure 4.9: Version 5: tile execution times; input size = 250

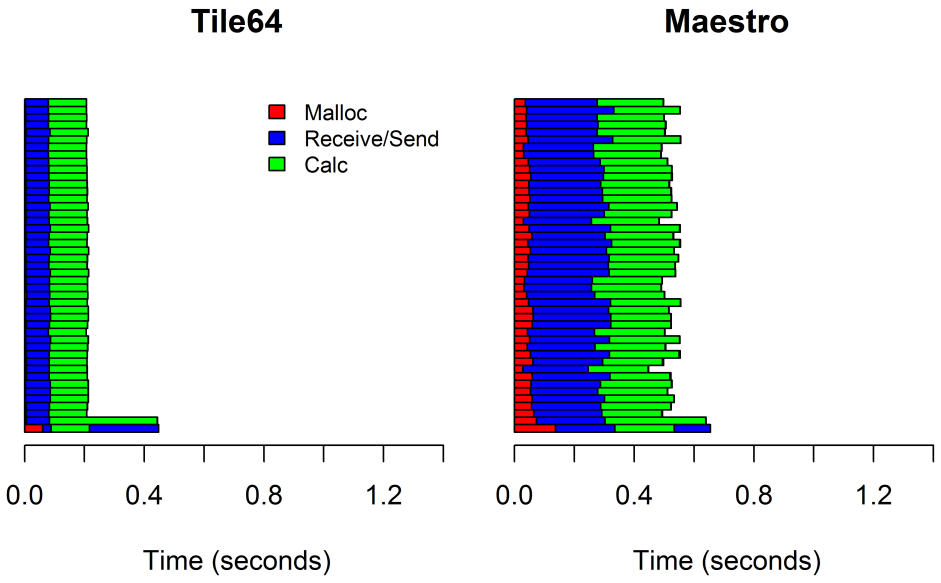


Figure 4.10: Version 5: tile execution times; input size = 375

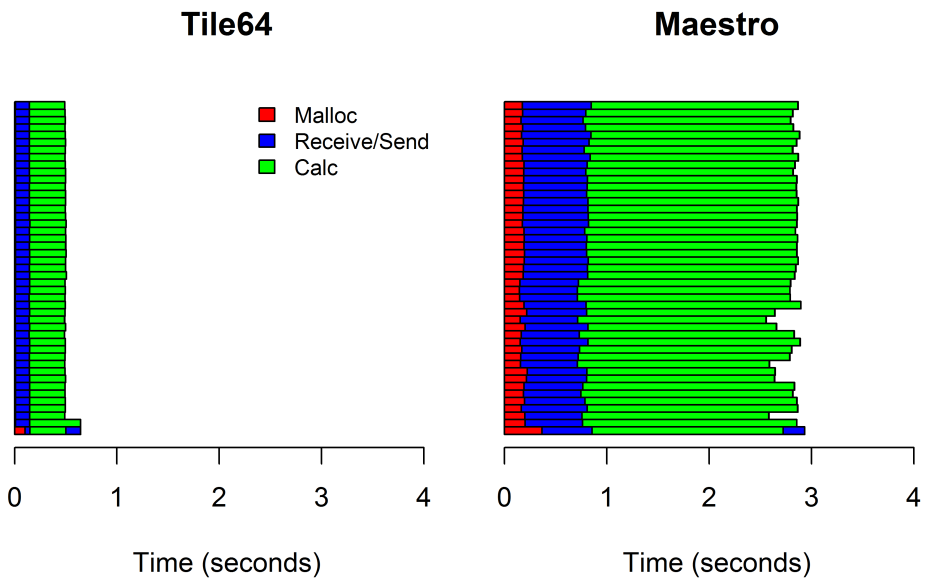


Figure 4.11: Version 5: tile execution times; input size = 500

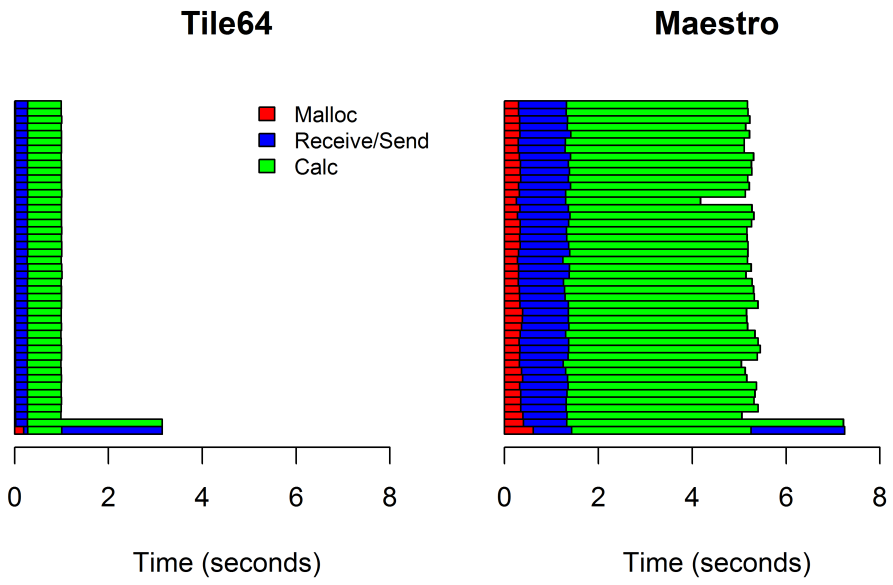


Figure 4.12: Version 5: tile execution times; input size = 625

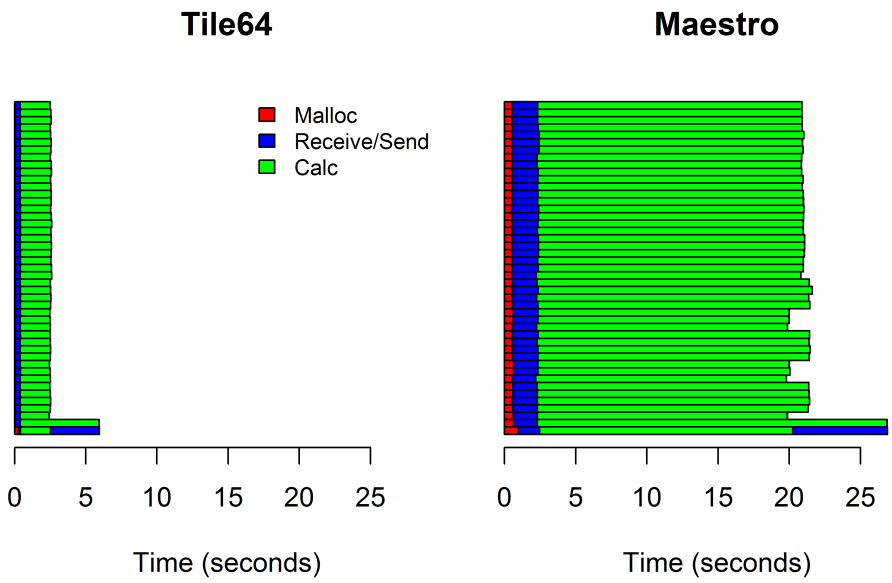


Figure 4.13: Version 5: tile execution times; input size = 750

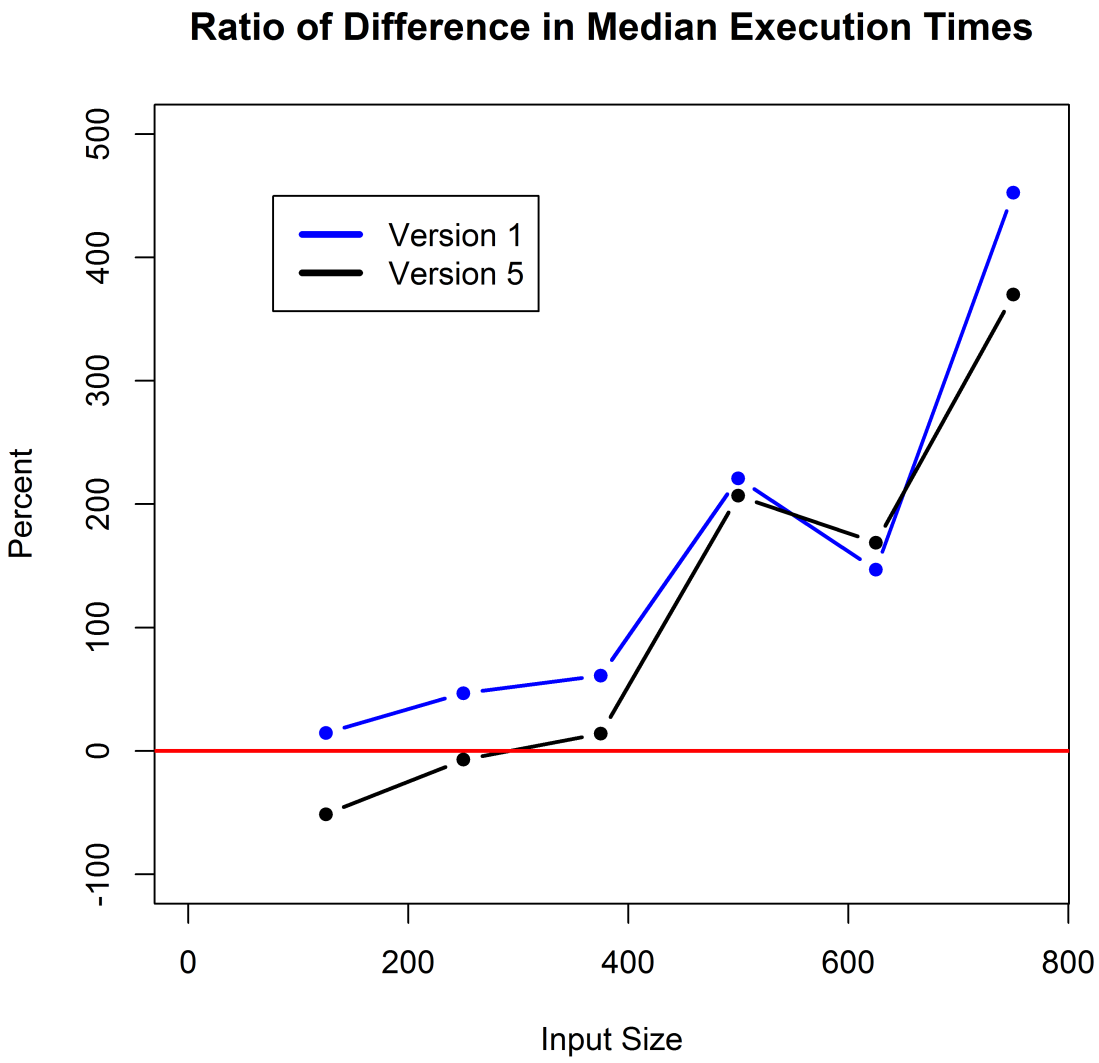


Figure 4.14: Difference between Tile64 and MAESTRO median execution times as a ratio

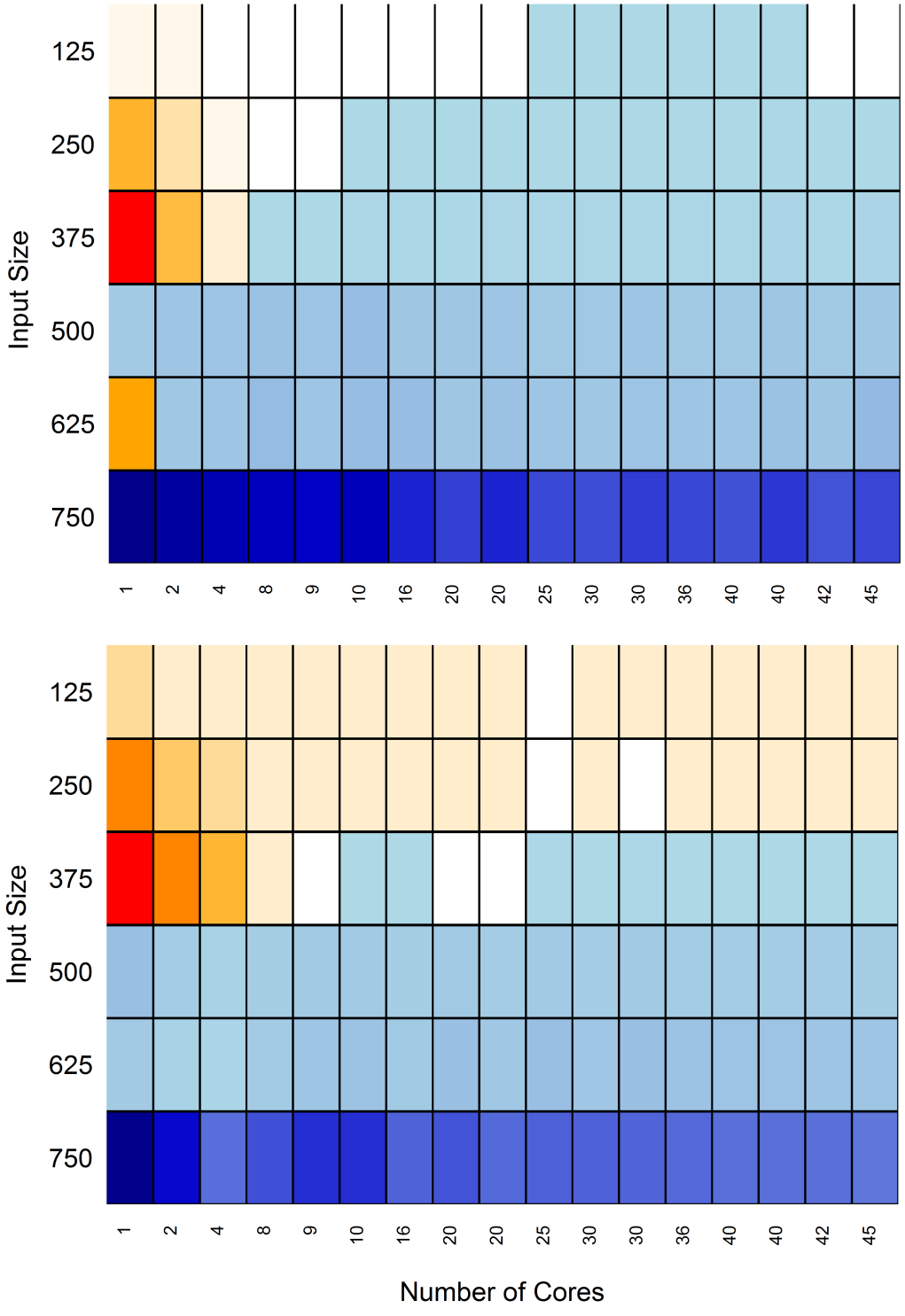


Figure 4.15: Heat Map comparison of MAESTRO and Tile64: Version 1 (top) and Version 5 (bottom). Orange/red = MAESTRO superiority. White = no difference. Blue = Tile64 superiority.

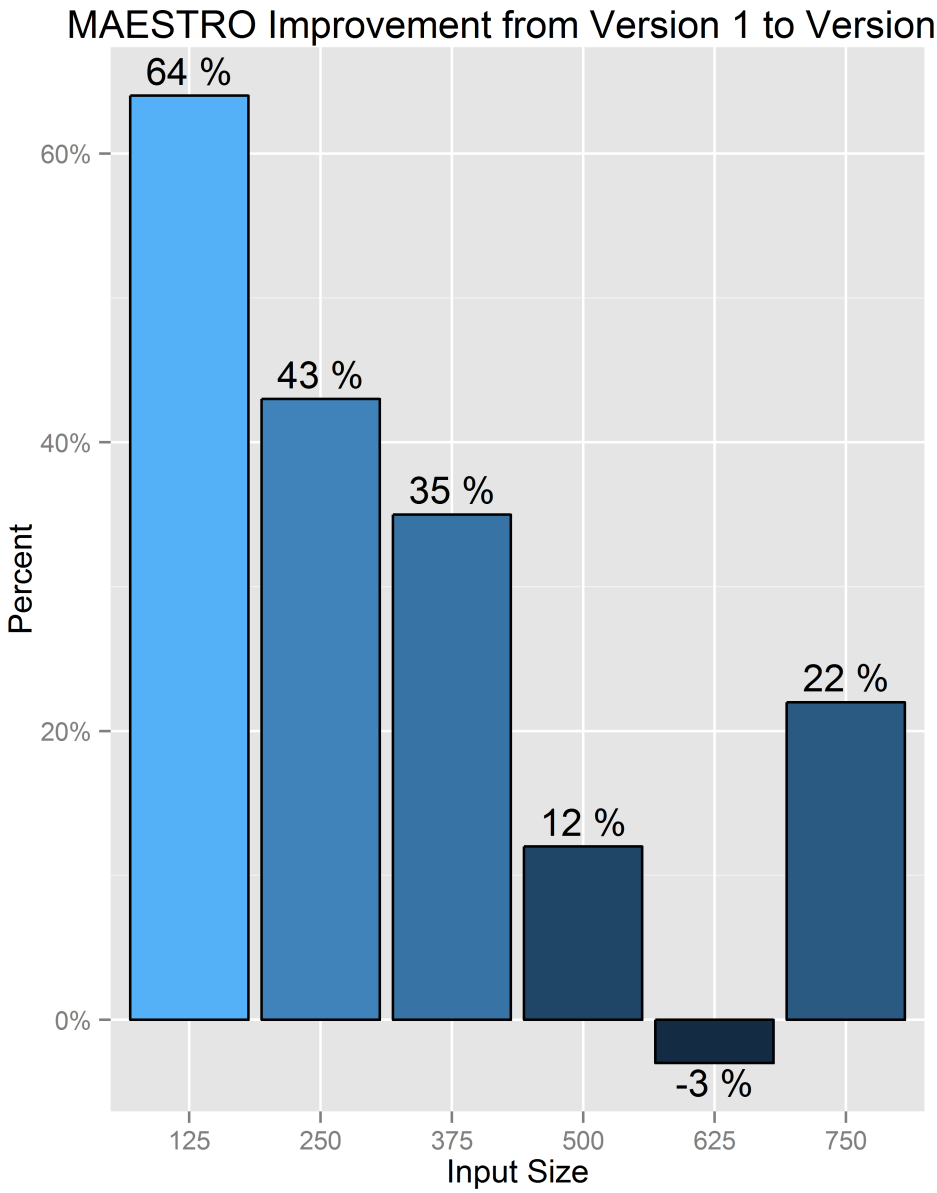


Figure 4.16: MAESTRO percent improvement of Version 5 over Version 1

V. Summary & Conclusions

5.1 Conclusions of Research

For the benchmarking suites as tested in their raw, unoptimized form, the MAESTRO board exhibited computational performance and efficiency that were generally inferior to those of the Tile64 board. These differences, however, were mitigated to some degree by modifying the benchmarking suites so that the code could take advantage of all four available memory controllers. Also, in the case of matrix multiplication operations, the software was able to be modified to allocate only as much memory as each processor would actually use in its calculations. For very large matrices, nearly two-thirds of the memory allocations per chip were superfluous and could be eliminated.

After optimizations (primarily on the MPI suite), the MAESTRO still remained inferior to the Tile64 for large input sizes and large numbers of cores, though to a lesser degree than it did without the optimized implementation, but proved to be superior otherwise. As input size increases, MAESTRO is comparatively slower due to memory saturation. MAESTRO tends to excel at computationally intensive operations because of its FPU. (Tile64 would probably dominate if the workload consisted of integer calculations as opposed to floating-point operations.) MAESTRO's limiting factors appear to be its memory architecture and its inter-tile communication network.

All in all, the hypothesis has been confirmed. The 49-core MAESTRO board can be made to exhibit computational performance that is commensurate with that of the Tile64 board for particular data-processing applications. Furthermore, it is possible to optimize the MAESTRO with a reasonable amount of effort, at least for the types of applications represented by the benchmarking suites used here.

5.2 Significance of Research

These results allow the sponsor and other potential users to see clearly the advantages and the drawbacks to using the MAESTRO for data processing. Workloads that are highly parallelizable are ideal applications for it, but are feasible only if some thought goes into managing the limited amount of memory on the board.

5.3 Recommendations for Future Research

Of interest for future research are the L1 and L2 cache miss rates. These characteristics were immeasurable in this effort due to compatibility problems between the OProfile suite and the MAESTRO. Similarly, the BenchFFT suite would provide additional understanding of the functionality of the MAESTRO, especially because it exercises the hardware's ability to run the Fast Fourier Transform (FFT) operation, which is commonly used in signal-processing applications. Also, the unexpected results from the HPL suite should be investigated further to determine why the Tile64 can execute more FLOPS than the MAESTRO.

Toward the end of this effort it became obvious that, for the MPI suite, a general algorithm needed to be designed to yield an equitable distribution of the cores to each of the four memory controllers. A simple slicing into quadrants does not always produce a balanced distribution, especially when the number of rows or columns is an odd number. One quadrant might be assigned four cores, whereas another could have nine. Also, some experimentation is needed to determine if there are notable differences among the various memory controllers. A quick and incomplete experiment suggested that the individual cores may execute data differently depending on which memory controller they are assigned to. In one case, when a core quadrant was assigned to different controllers, it executed data faster when allocated to a controller that was on the opposite side of the board than when assigned to the nearest controller. This may be worth investigating if further optimizations are desired.

Appendix A: HPL Graphs

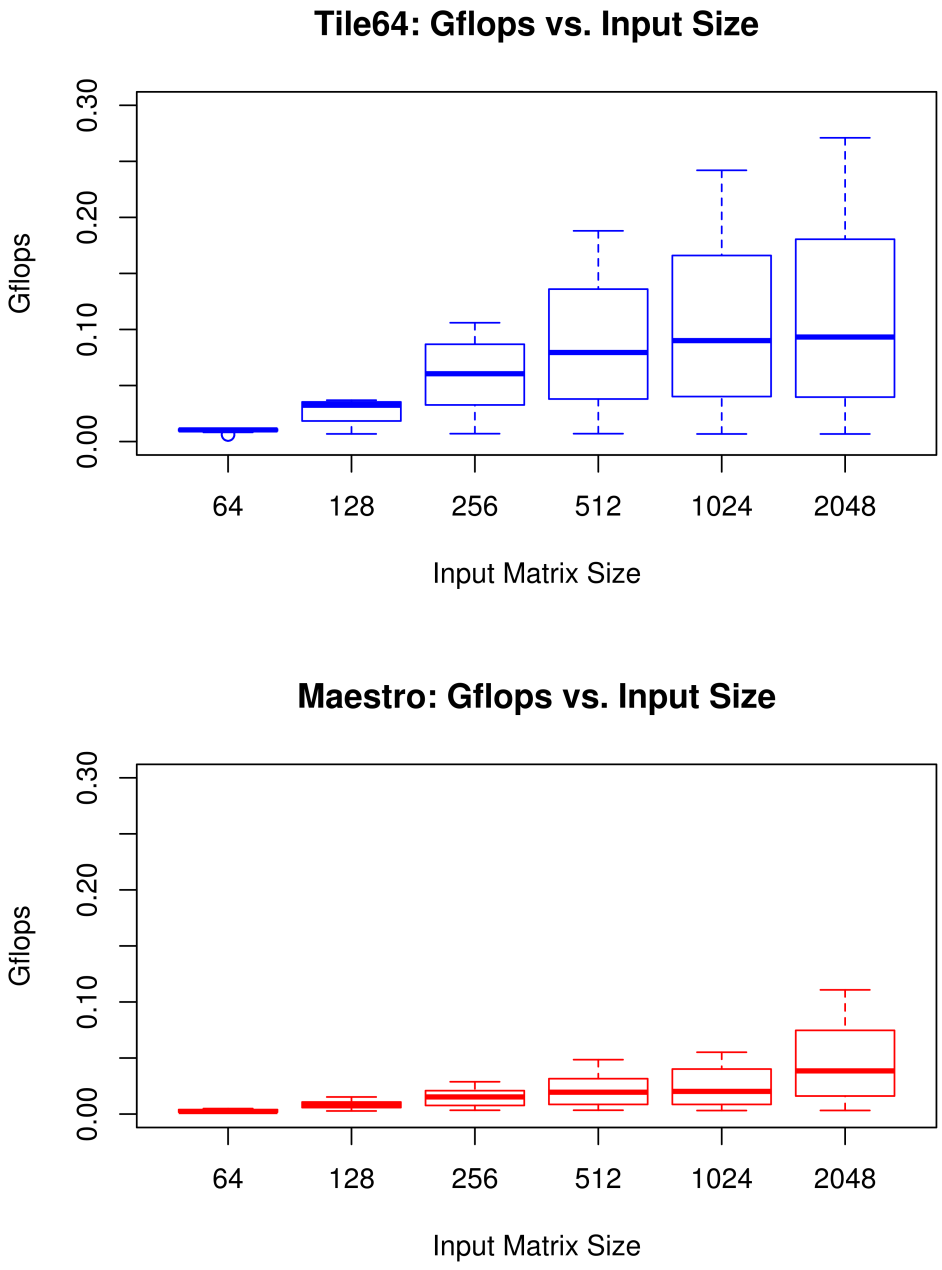


Figure A.1: HPL: GFLOPS per Input Size

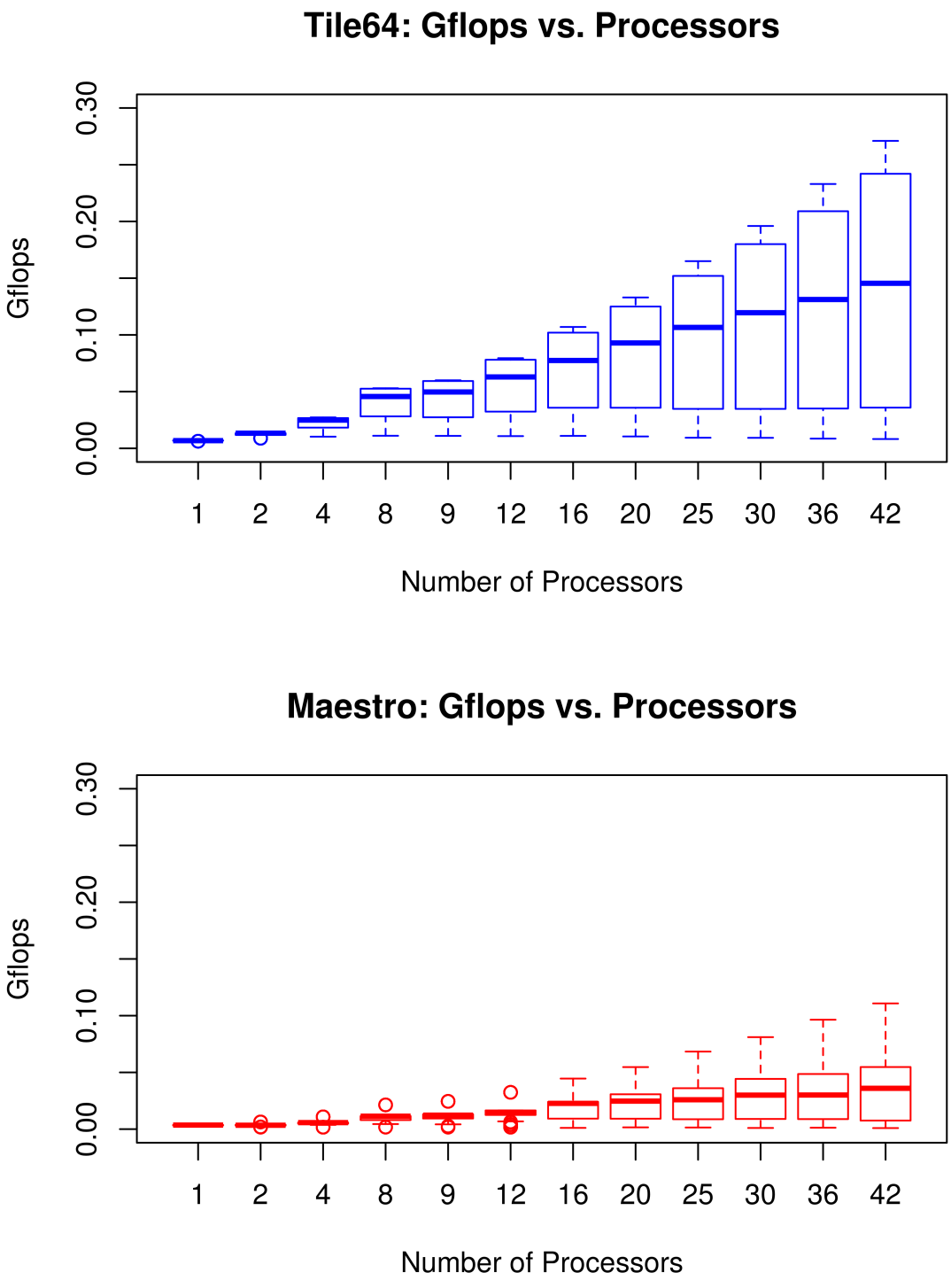


Figure A.2: HPL: GFLOPS per Number of Cores

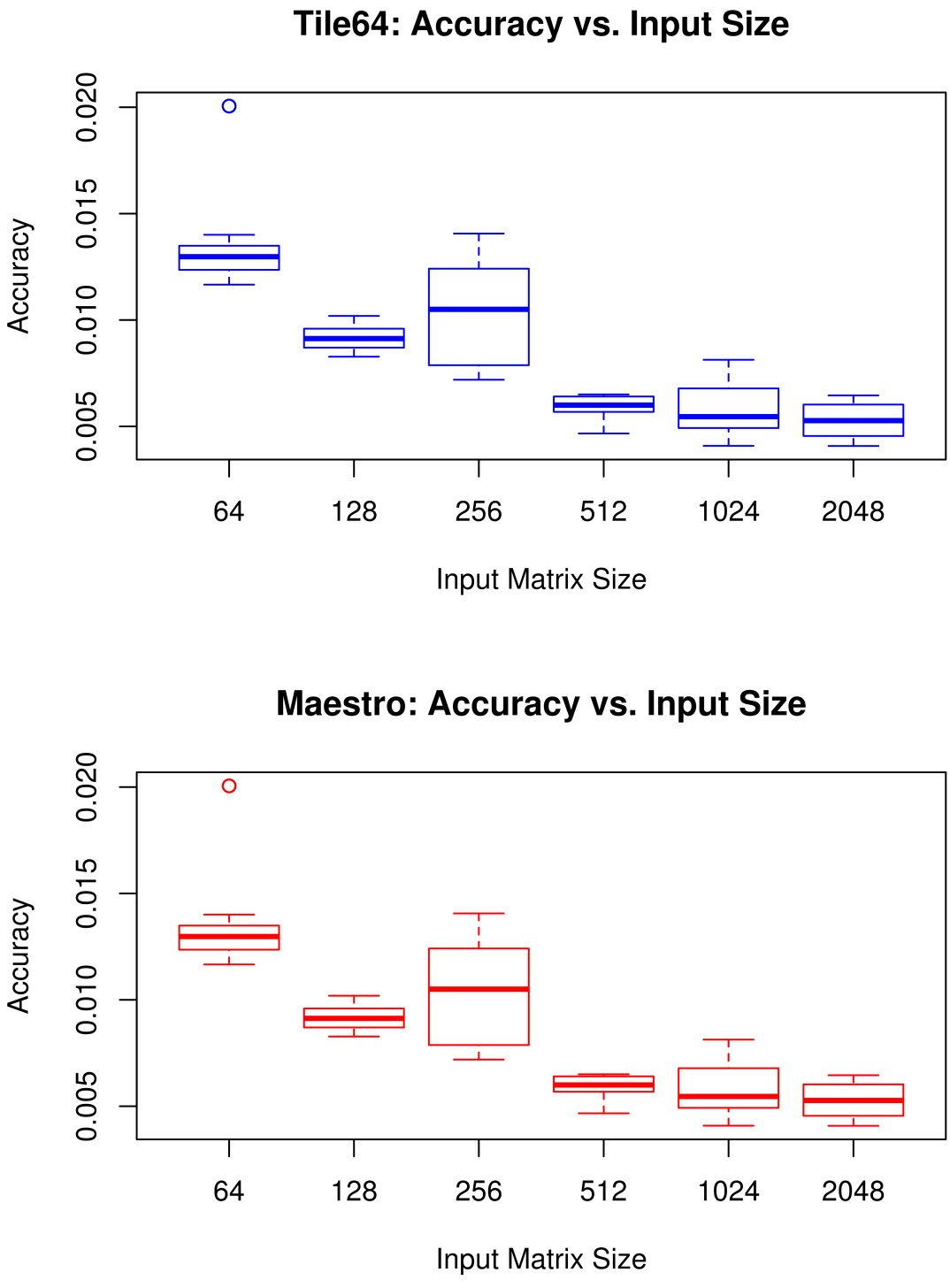
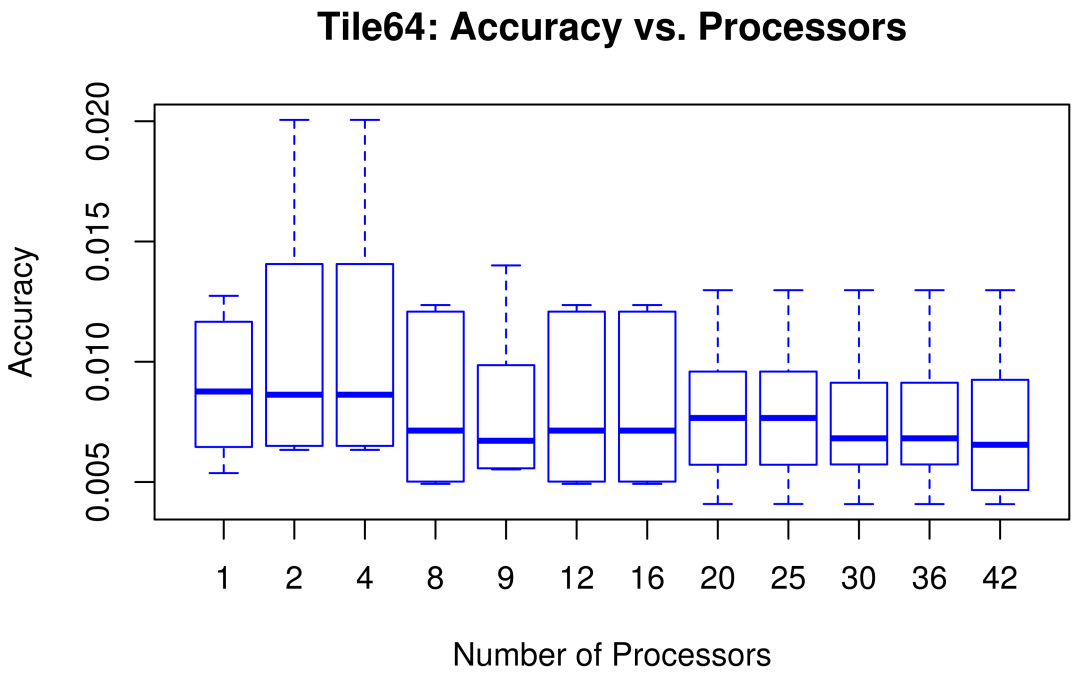


Figure A.3: HPL: Accuracy per Input Size



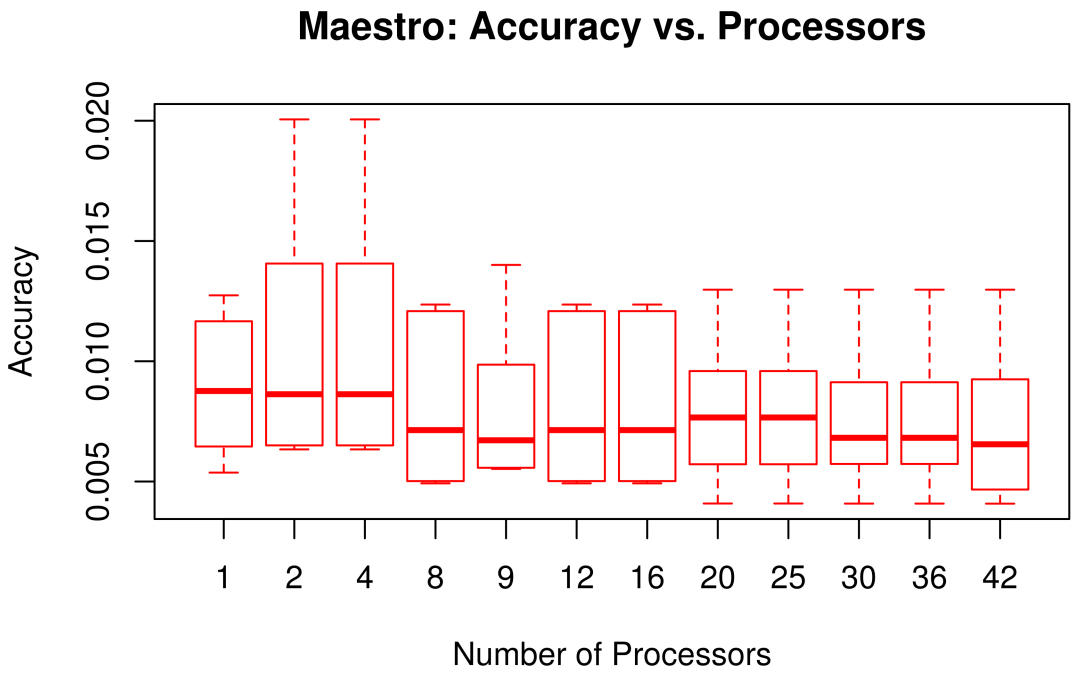


Figure A.4: HPL: Accuracy per Number of Cores

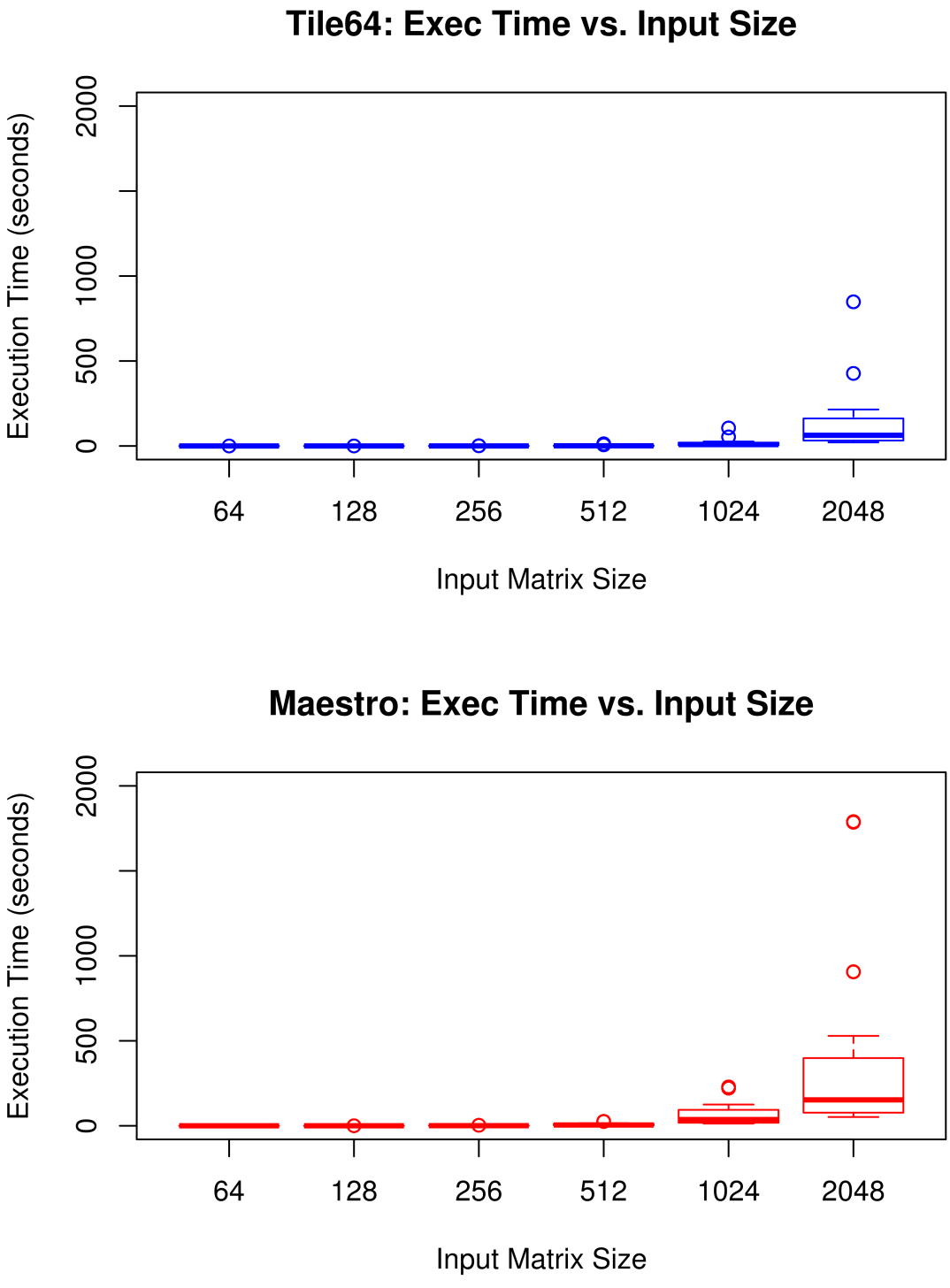


Figure A.5: HPL: Execution Time per Input Size

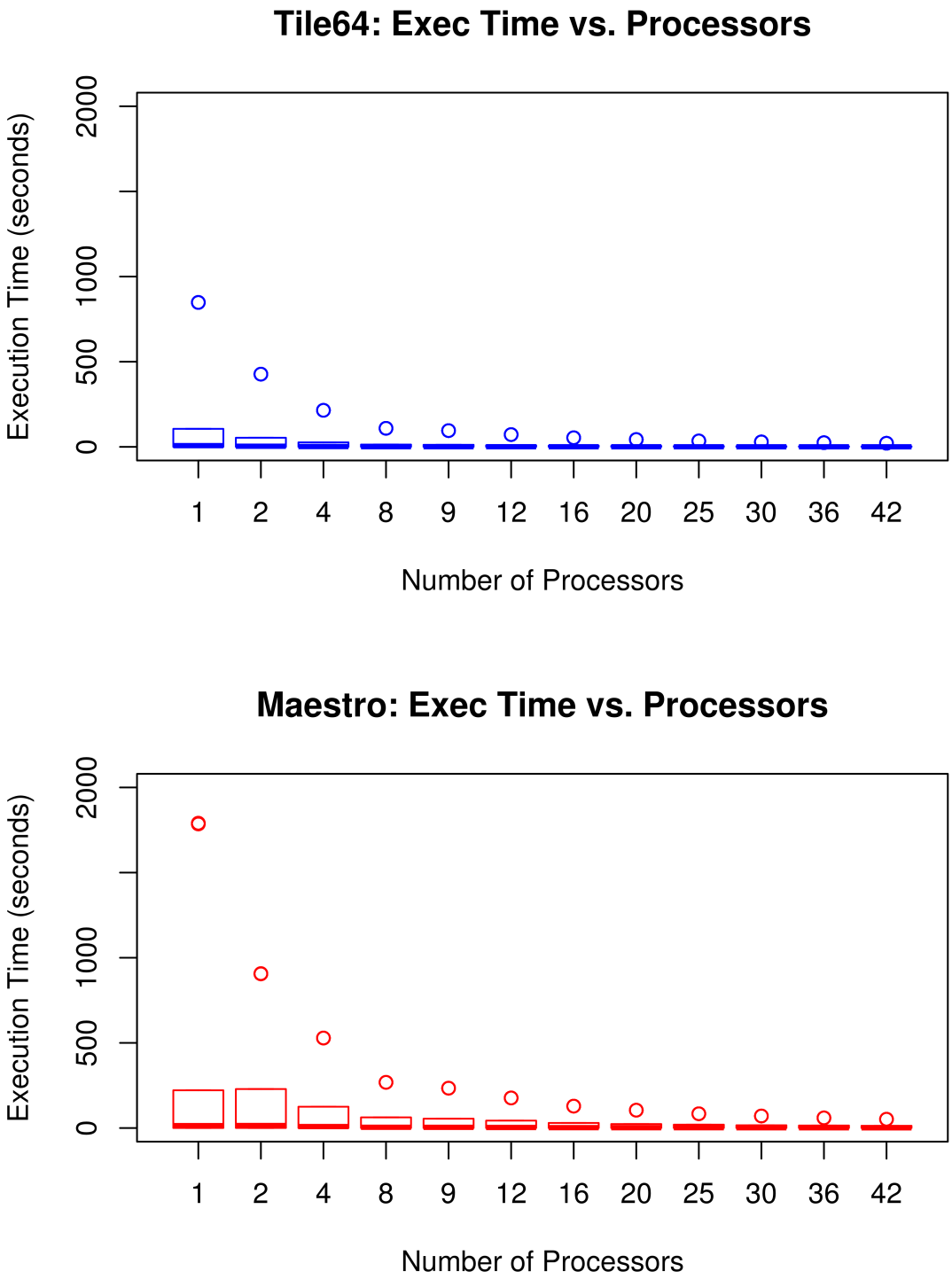


Figure A.6: HPL: Execution Time per Number of Cores

Appendix B: SD-VBS Graphs

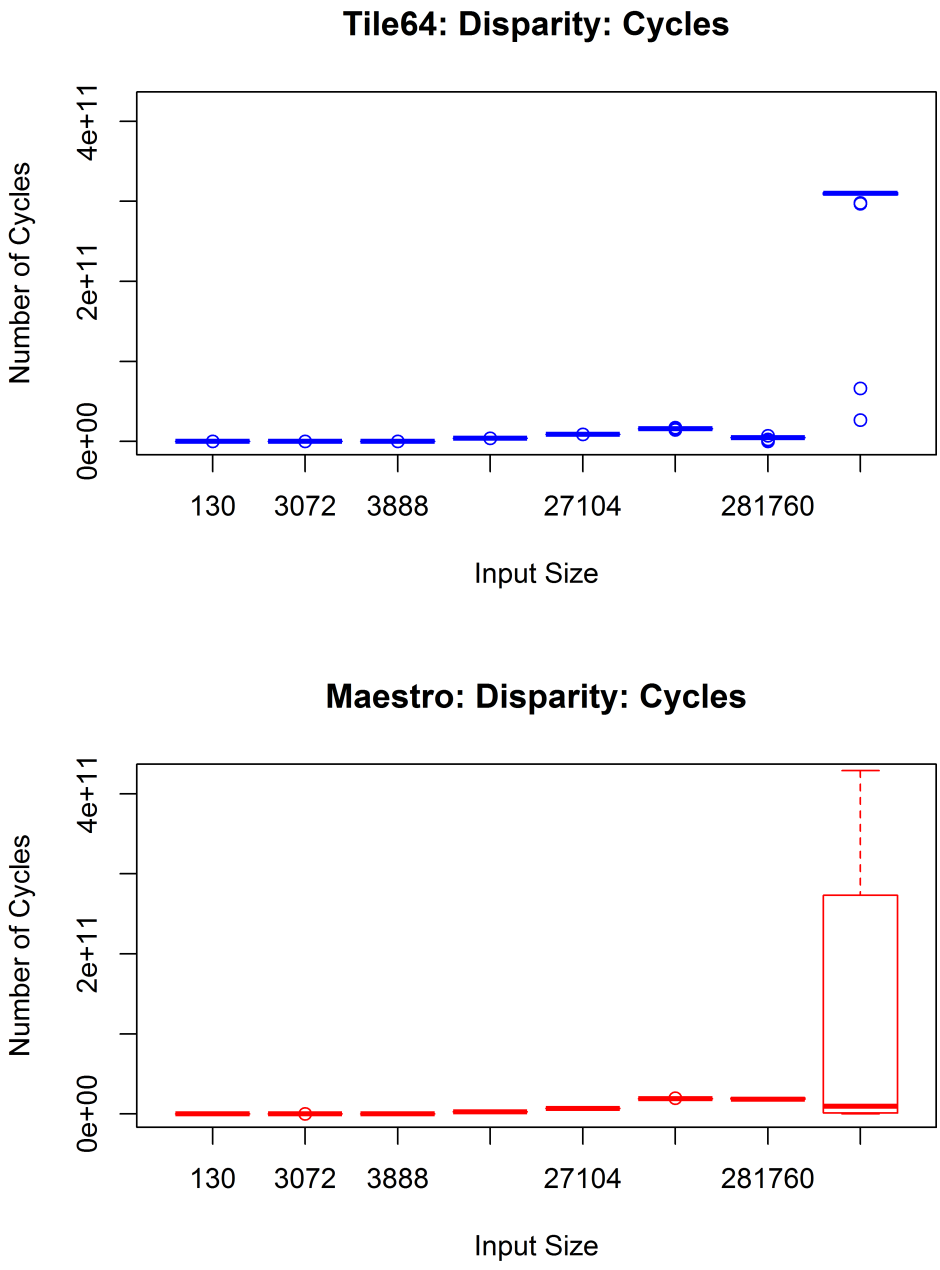


Figure B.1: Cycles per Input Size (Benchmark: Disparity Map)

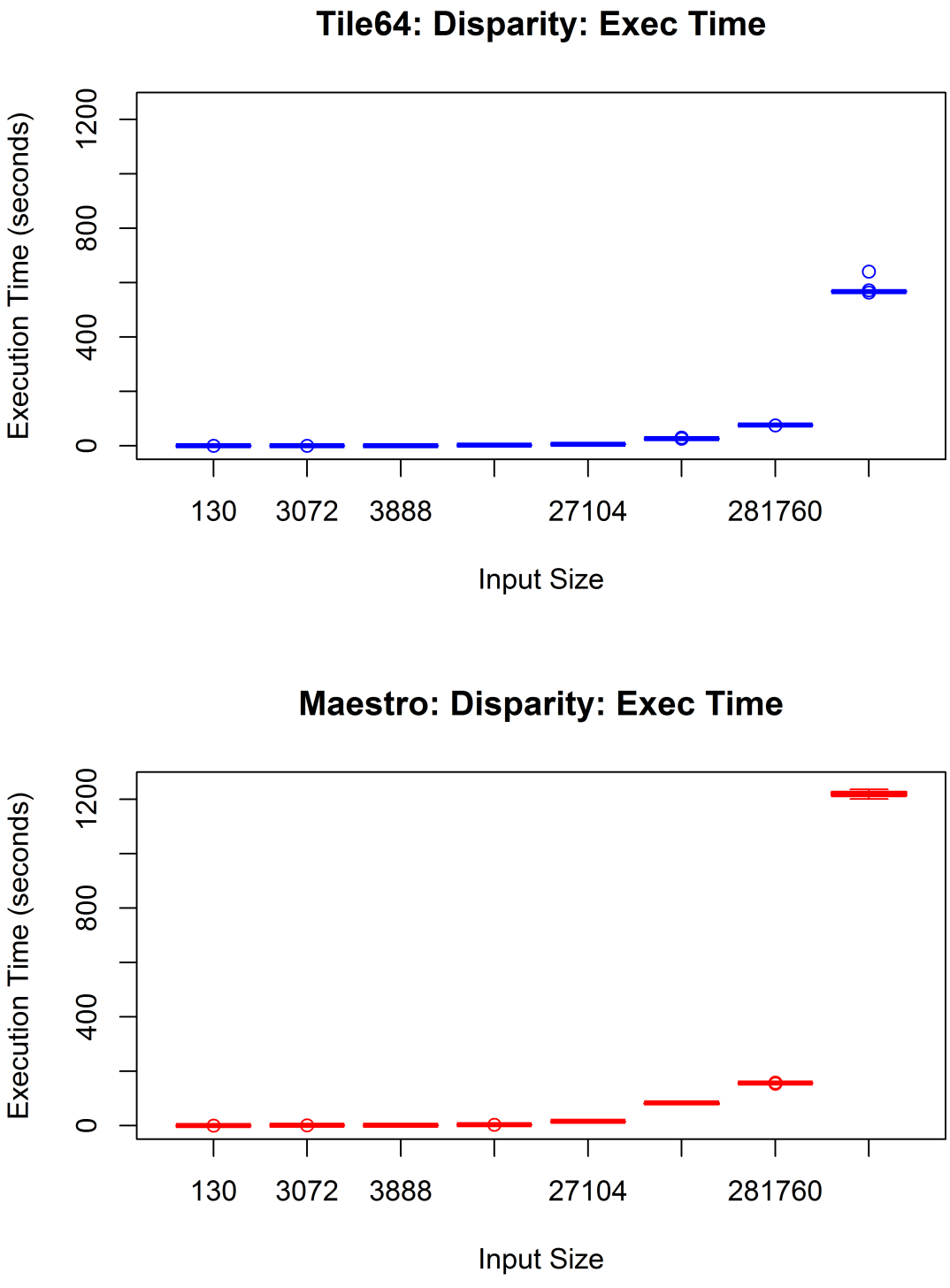


Figure B.2: Execution Time per Input Size (Benchmark: Disparity Map)

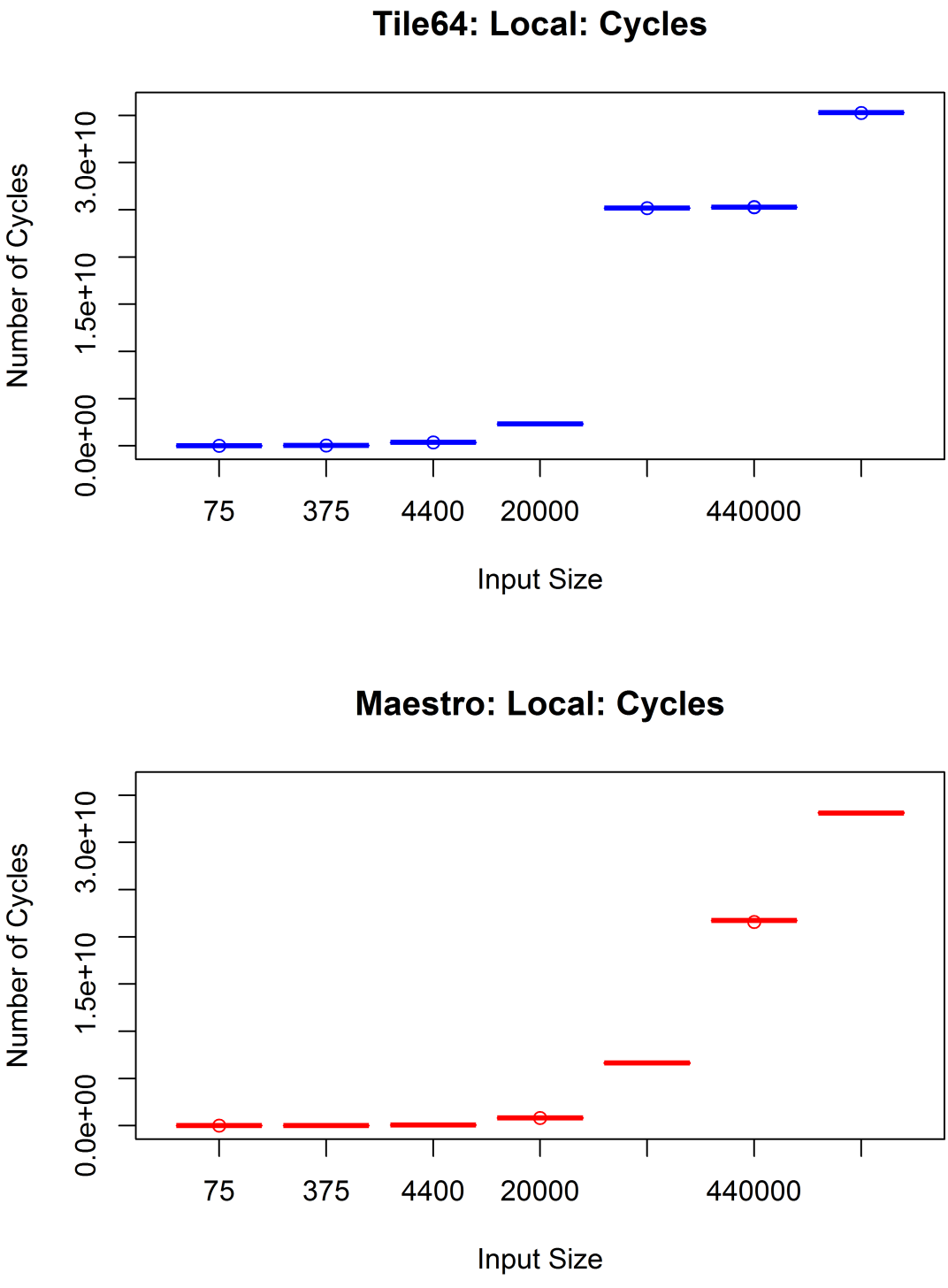


Figure B.3: Cycles per Input Size (Benchmark: Robot Localization)

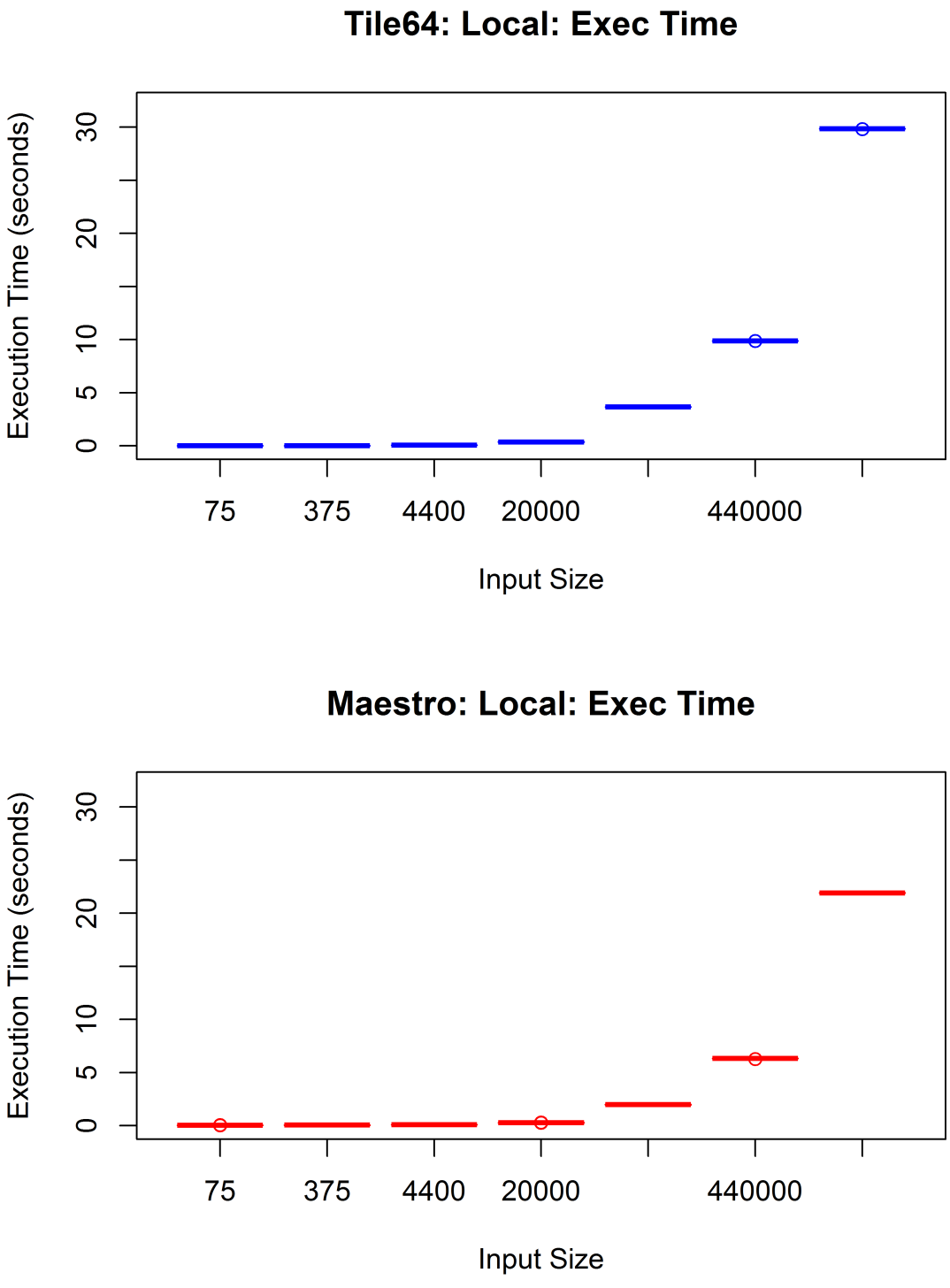


Figure B.4: Execution Time per Input Size (Benchmark: Robot Localization)

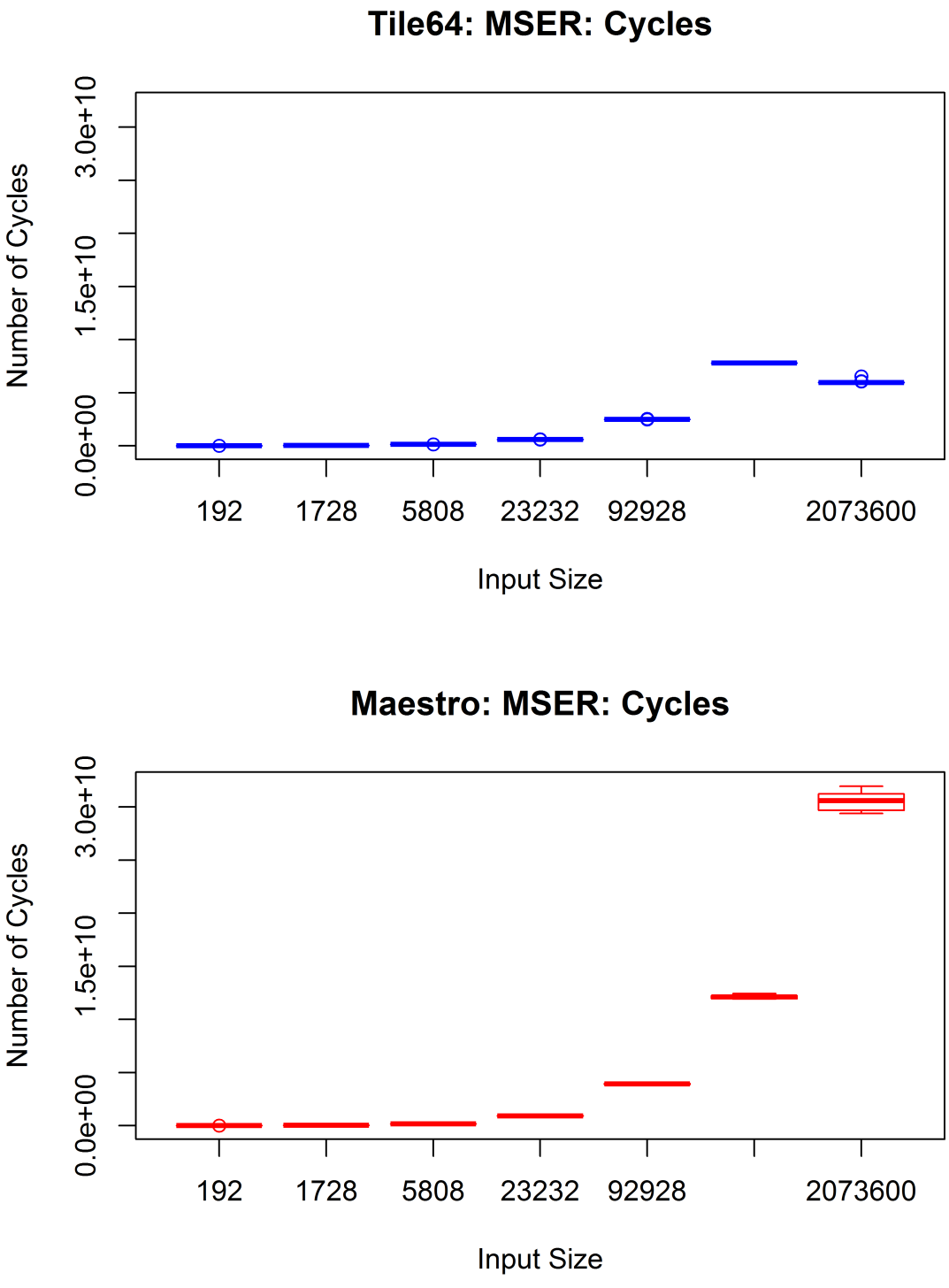
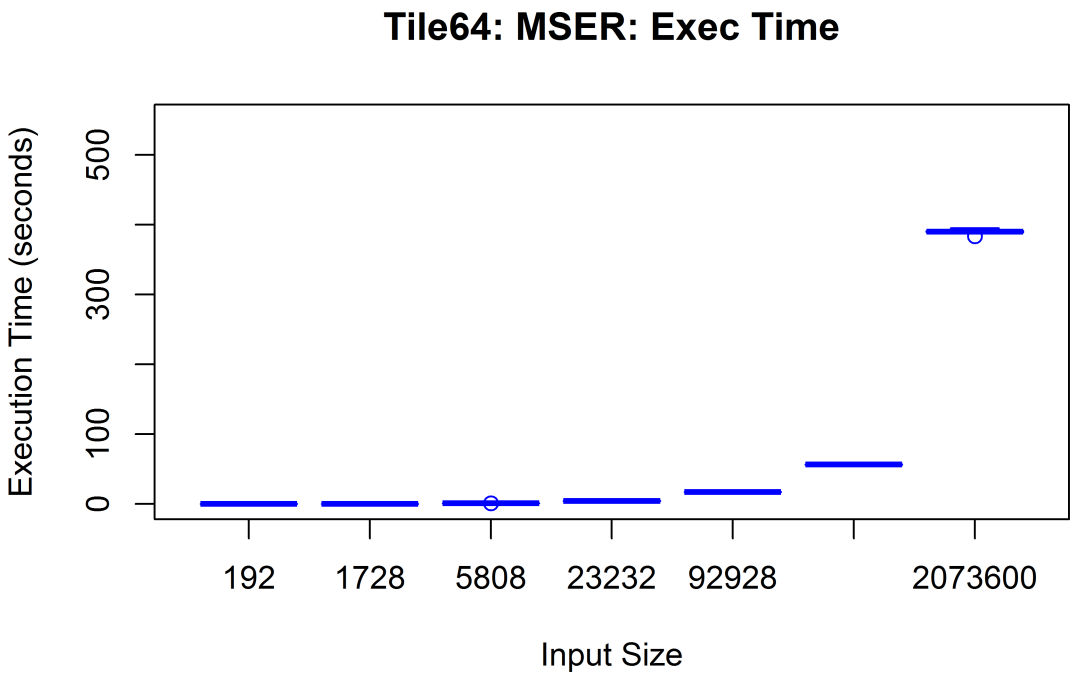


Figure B.5: Cycles per Input Size (Benchmark: Maximally Stable Regions)



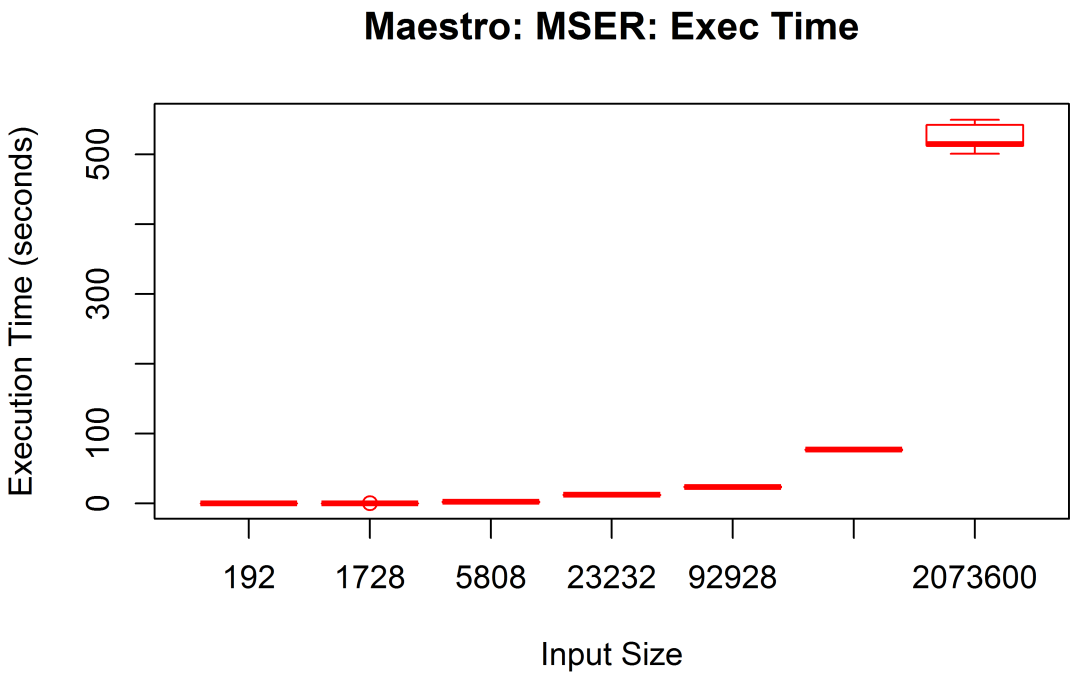


Figure B.6: Execution Time per Input Size (Benchmark: Maximally Stable Regions)

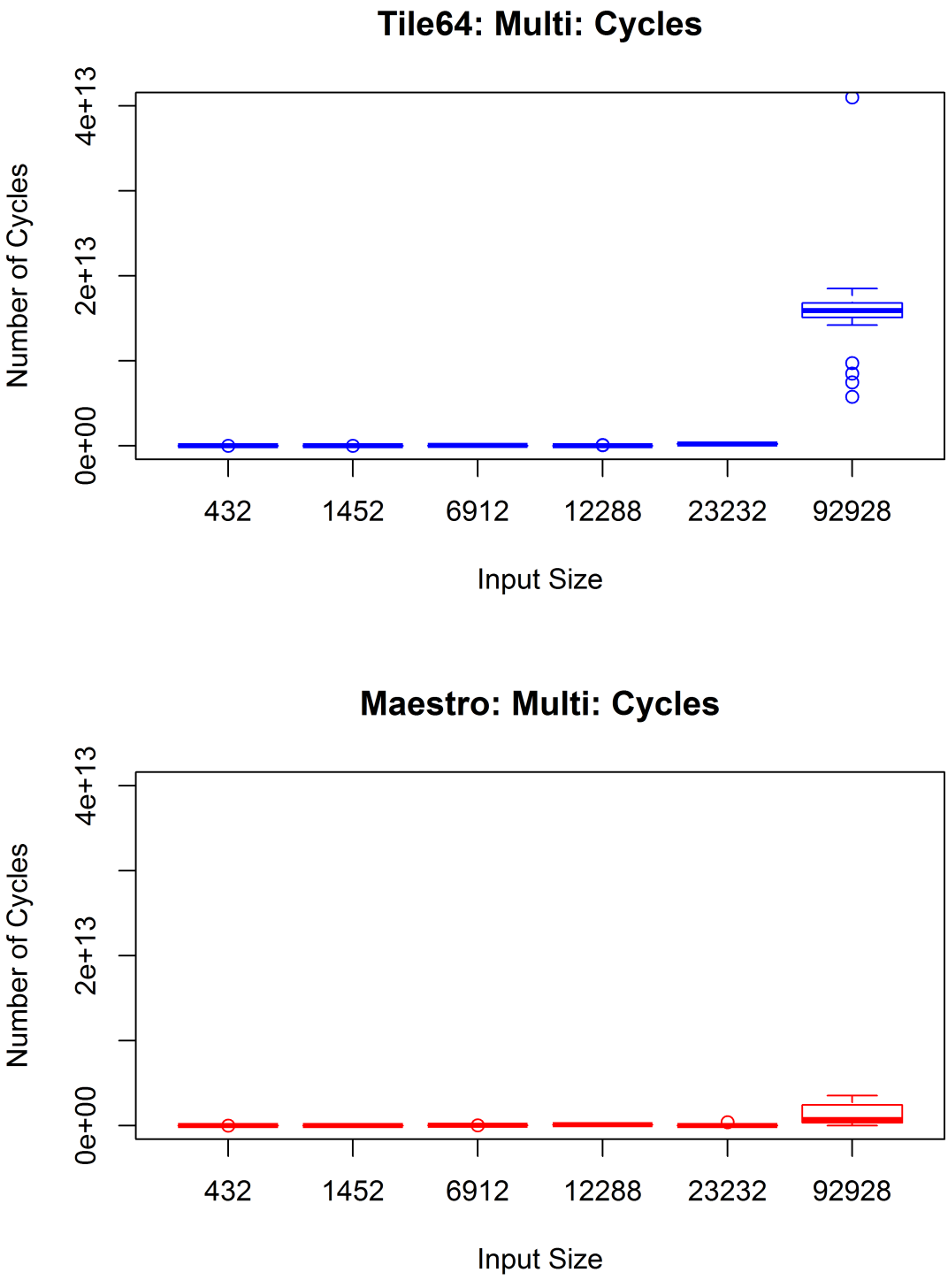


Figure B.7: Cycles per Input Size (Benchmark: Multi)

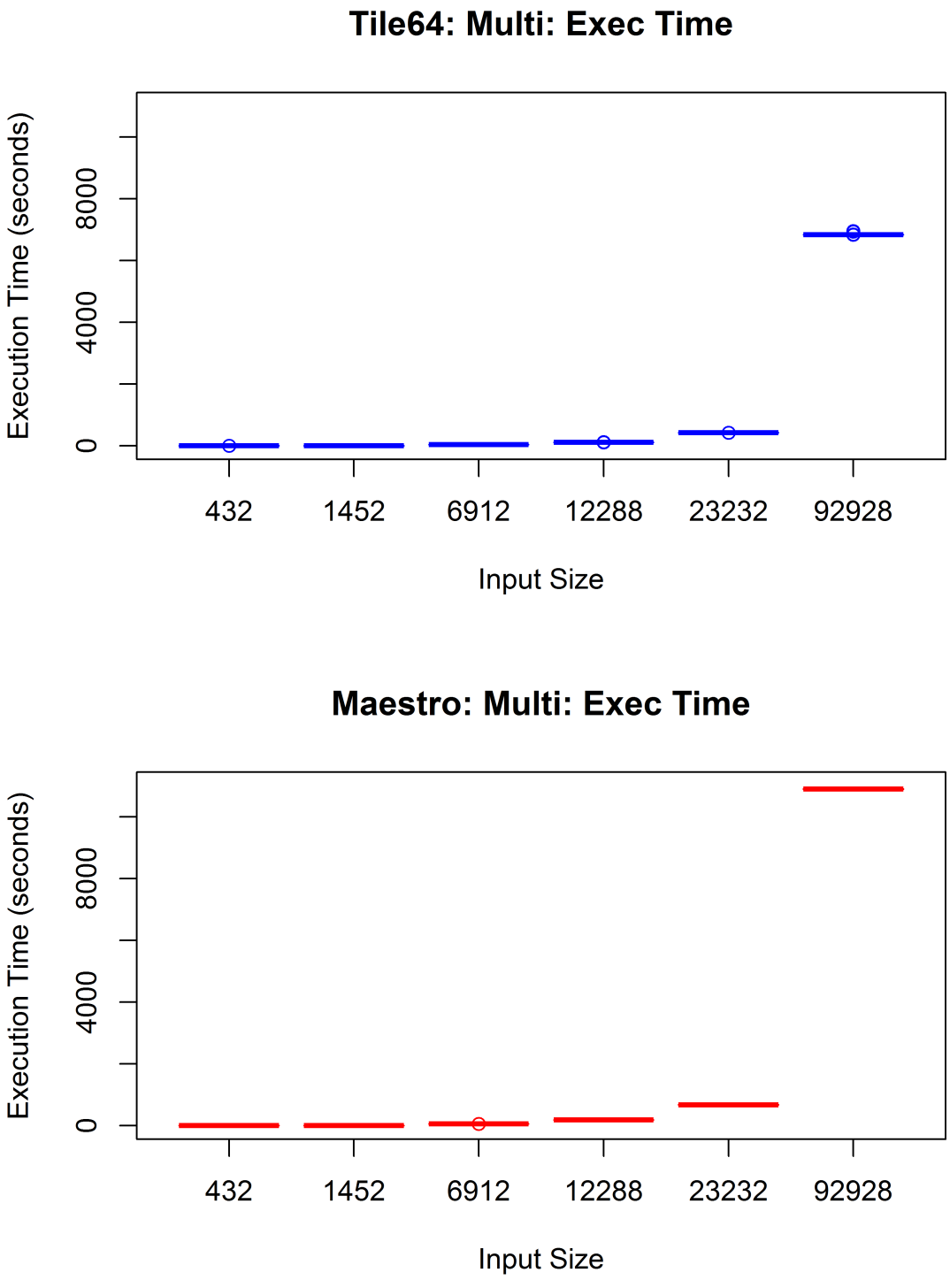


Figure B.8: Execution Time per Input Size (Benchmark: Multi)

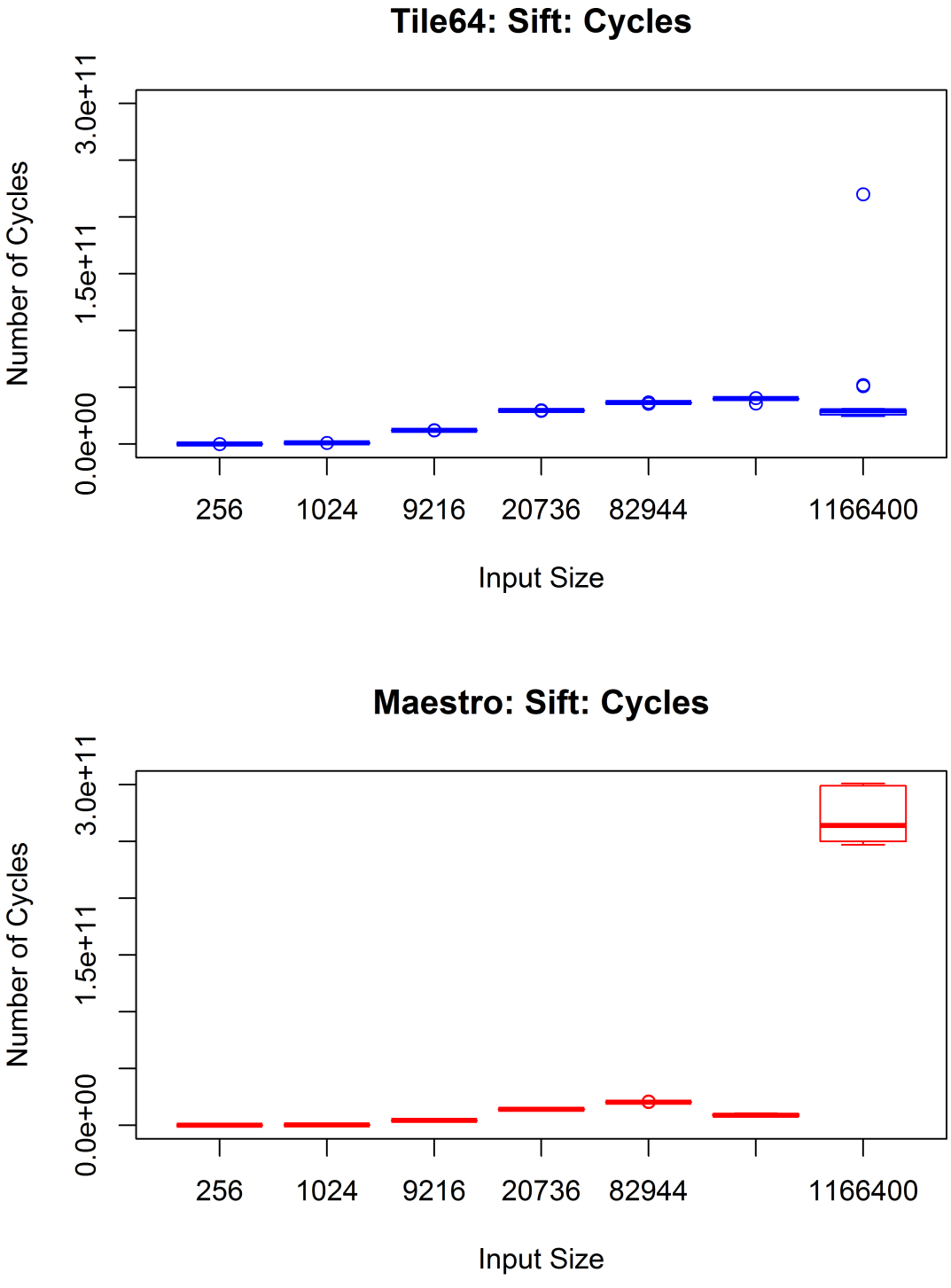
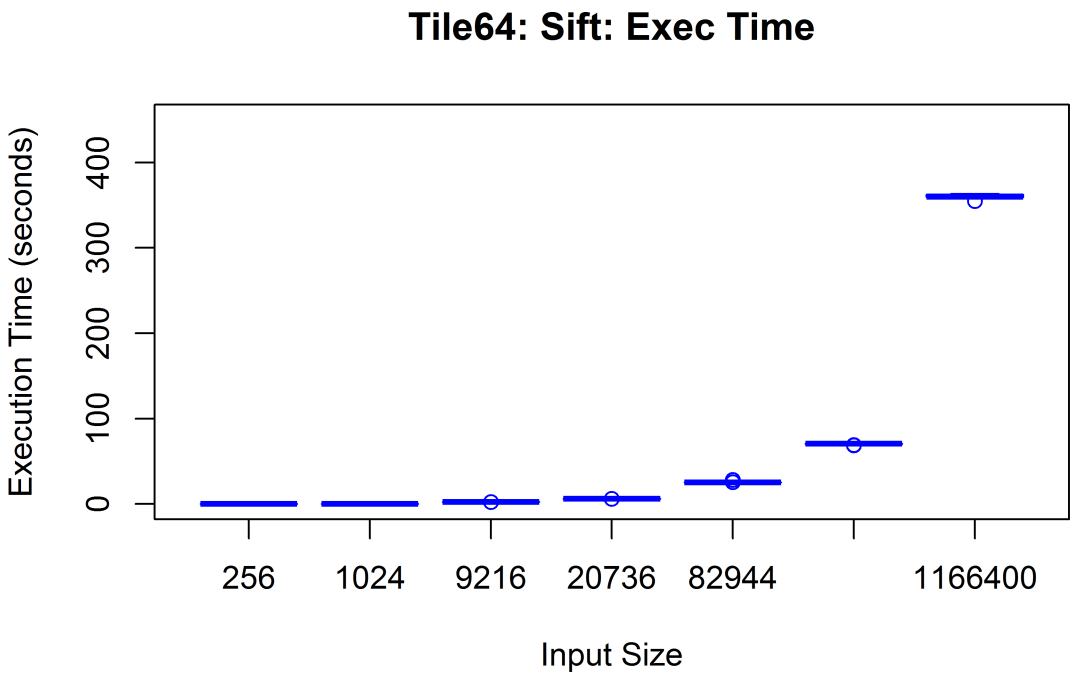


Figure B.9: Cycles per Input Size (Benchmark: Scale Invariant Feature Transform)



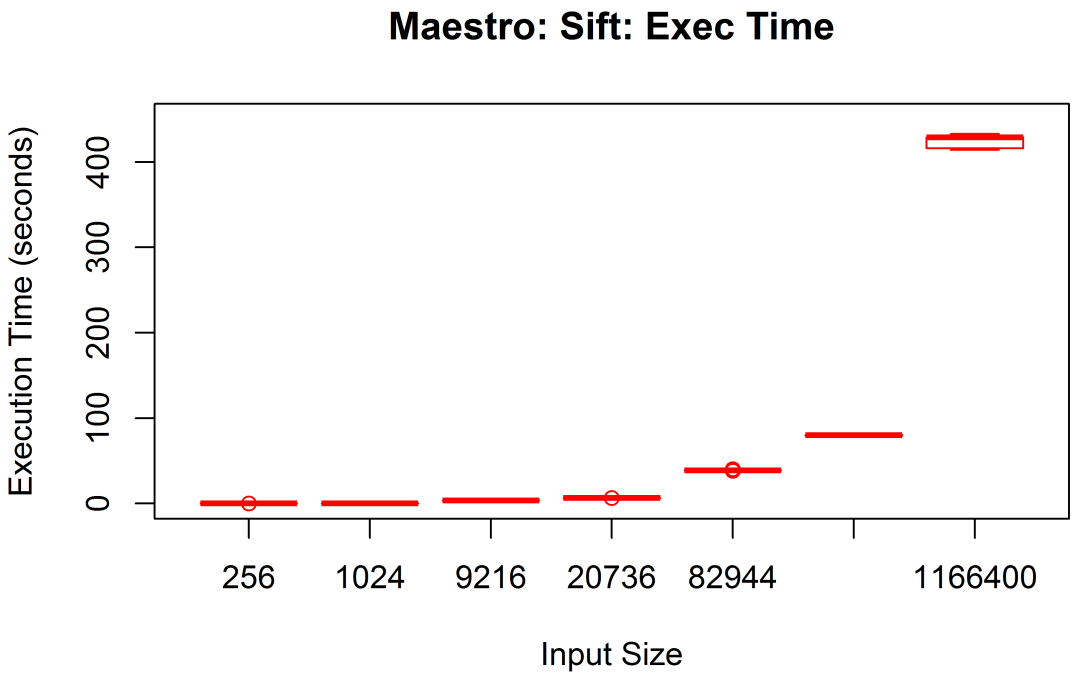
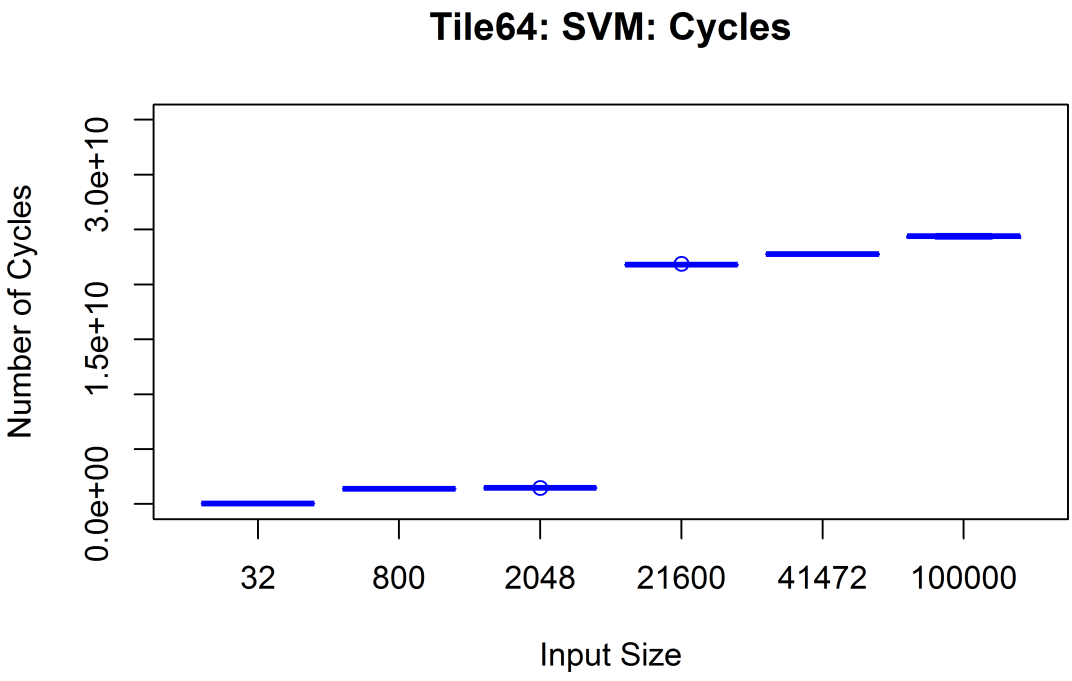


Figure B.10: Execution Time per Input Size (Benchmark: Scale Invariant Feature Transform)



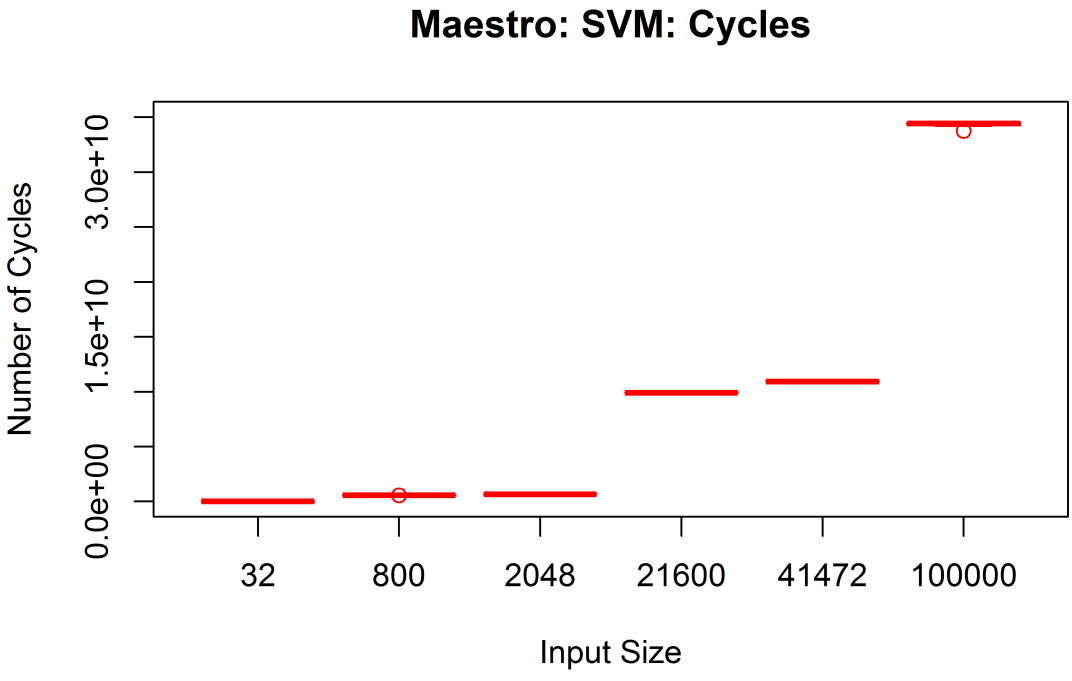
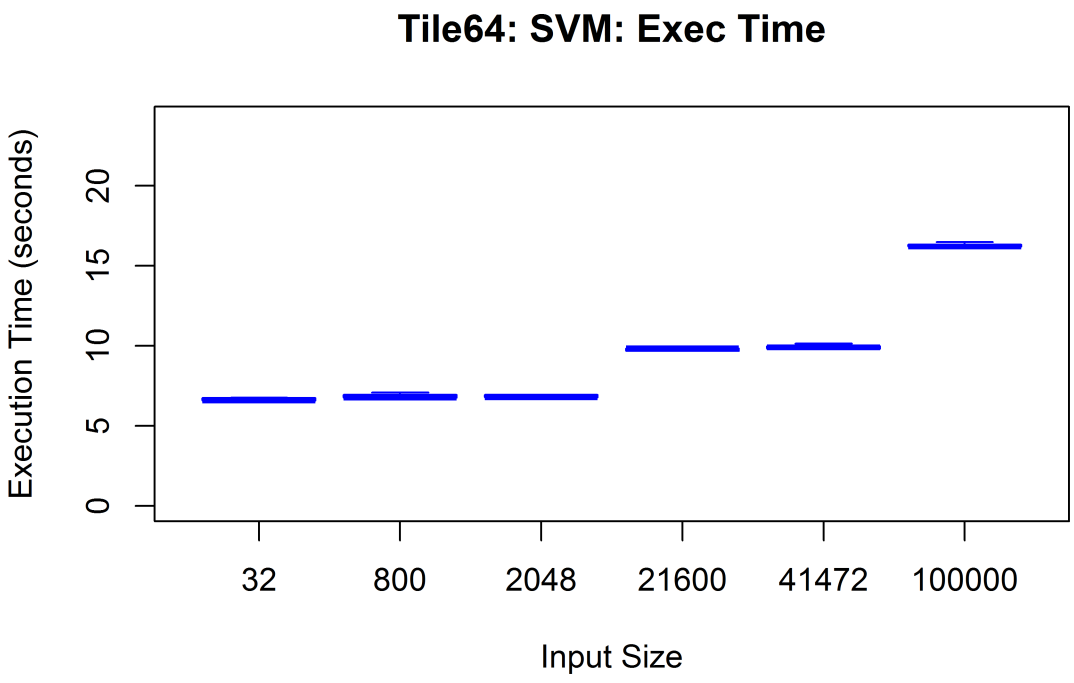


Figure B.11: Cycles per Input Size (Benchmark: Support Vector Machines)



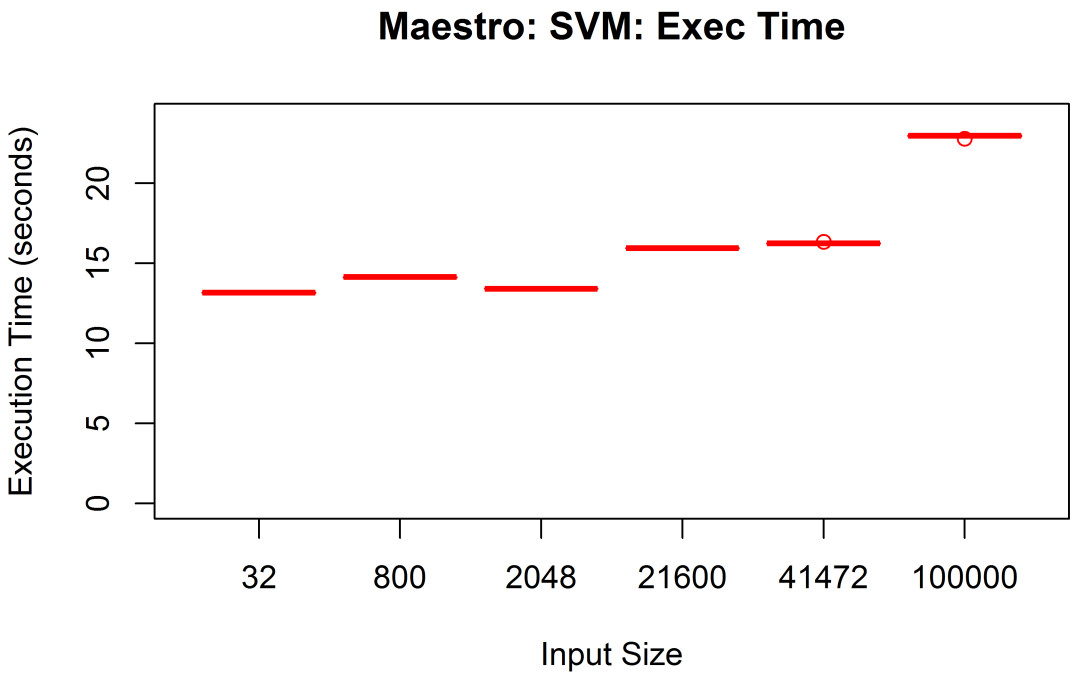


Figure B.12: Execution Time per Input Size (Benchmark: Support Vector Machines)

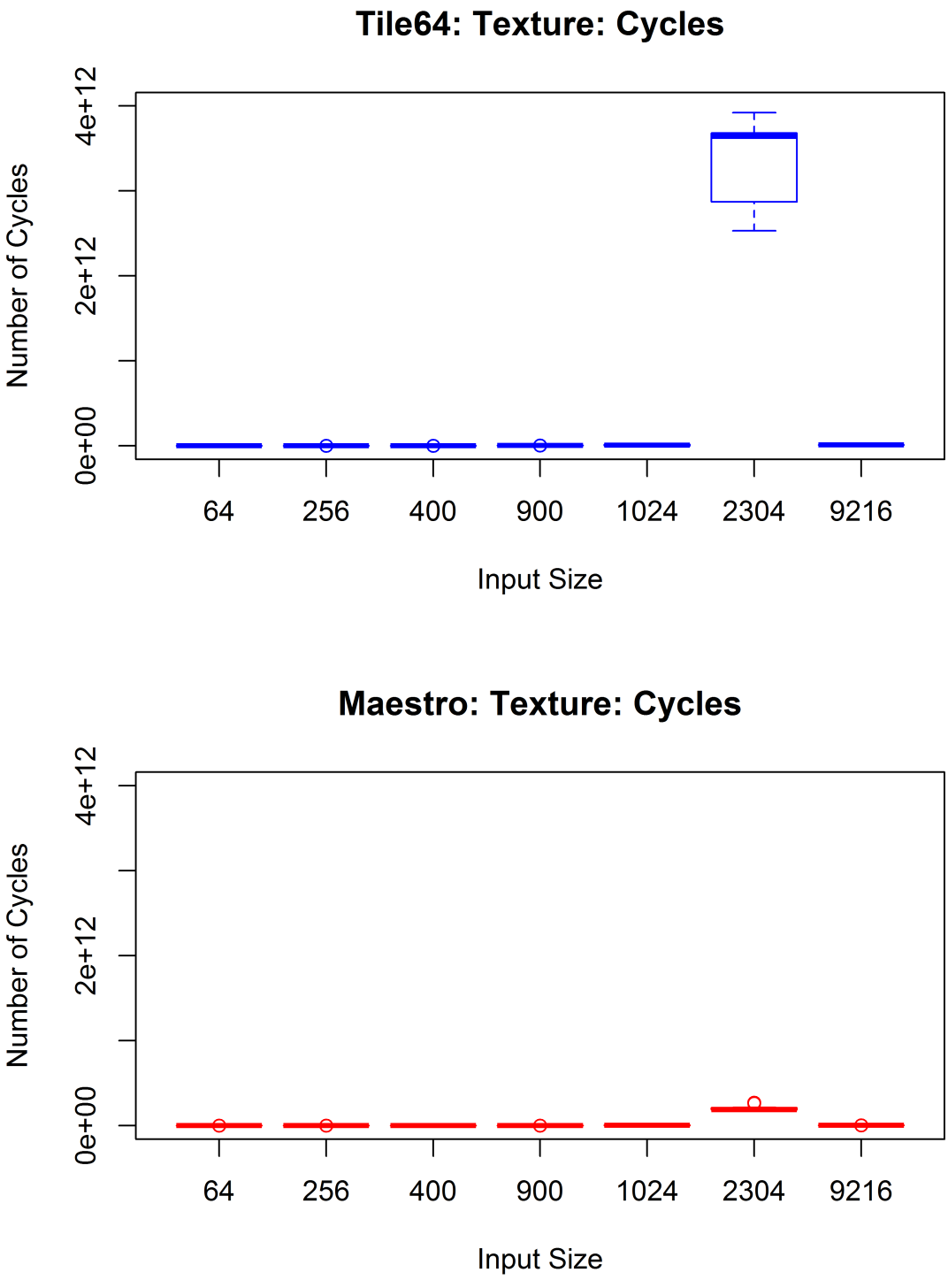


Figure B.13: Cycles per Input Size (Benchmark: Texture Synthesis)

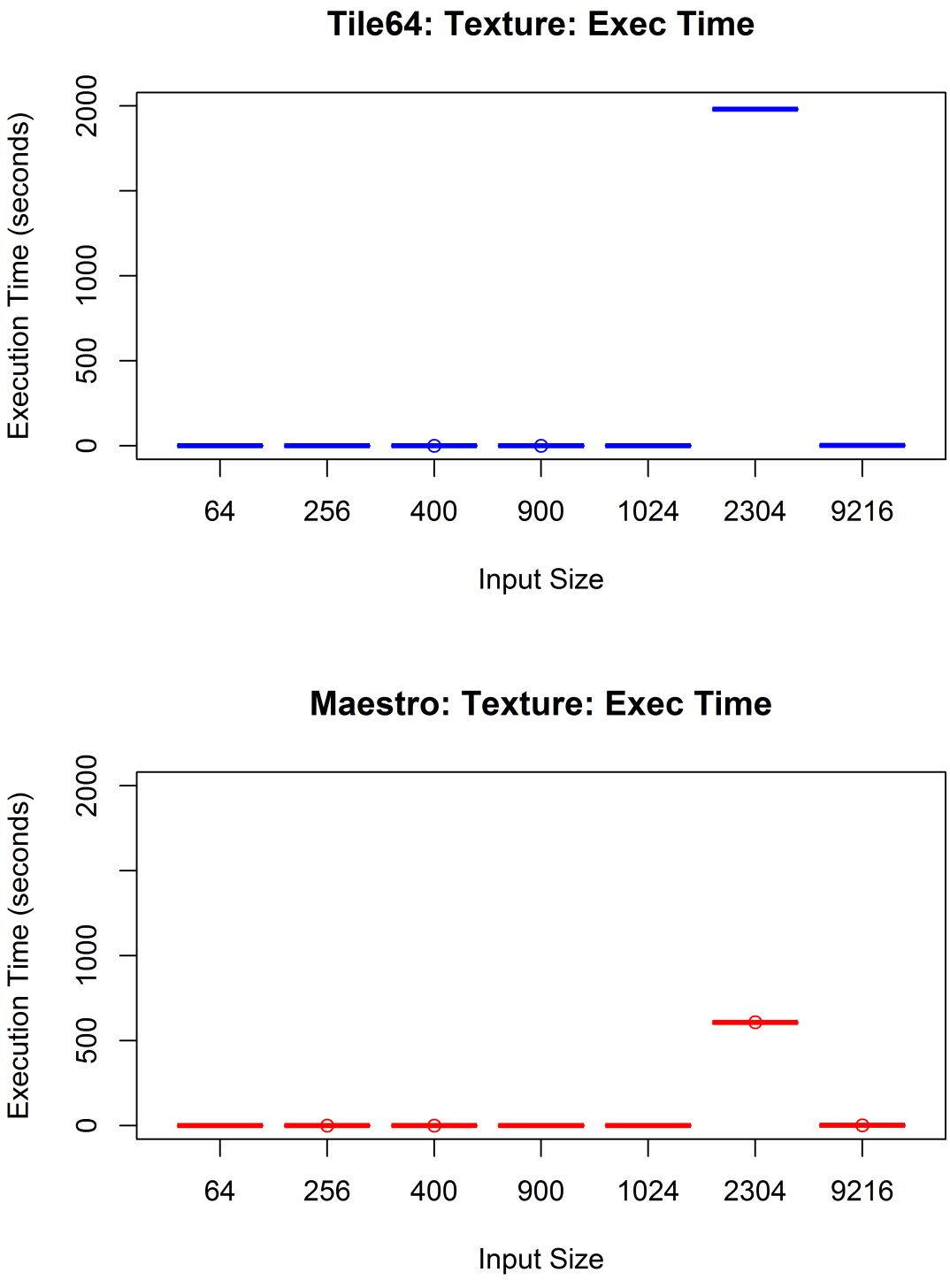
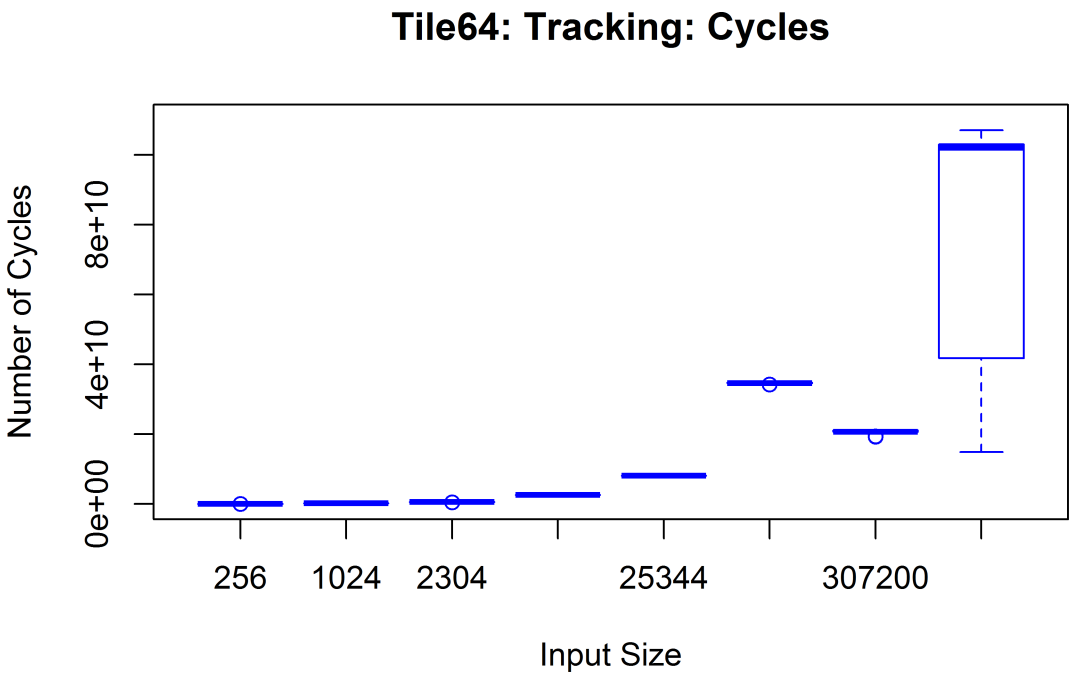


Figure B.14: Execution Time per Input Size (Benchmark: Texture Synthesis)



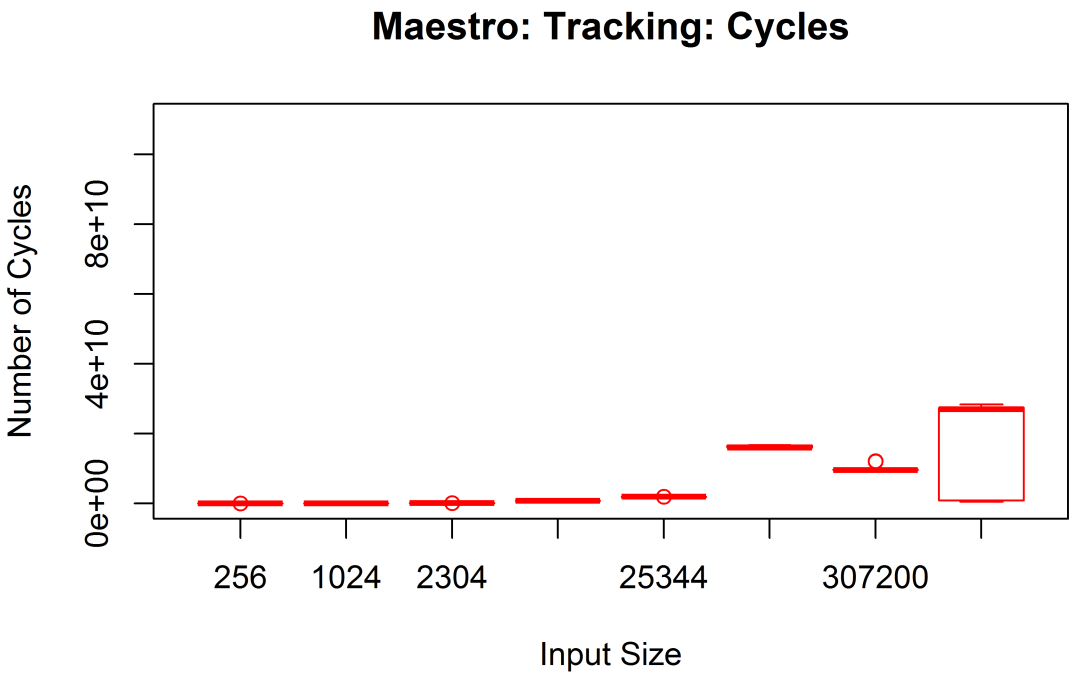


Figure B.15: Cycles per Input Size (Benchmark: Feature Tracking)

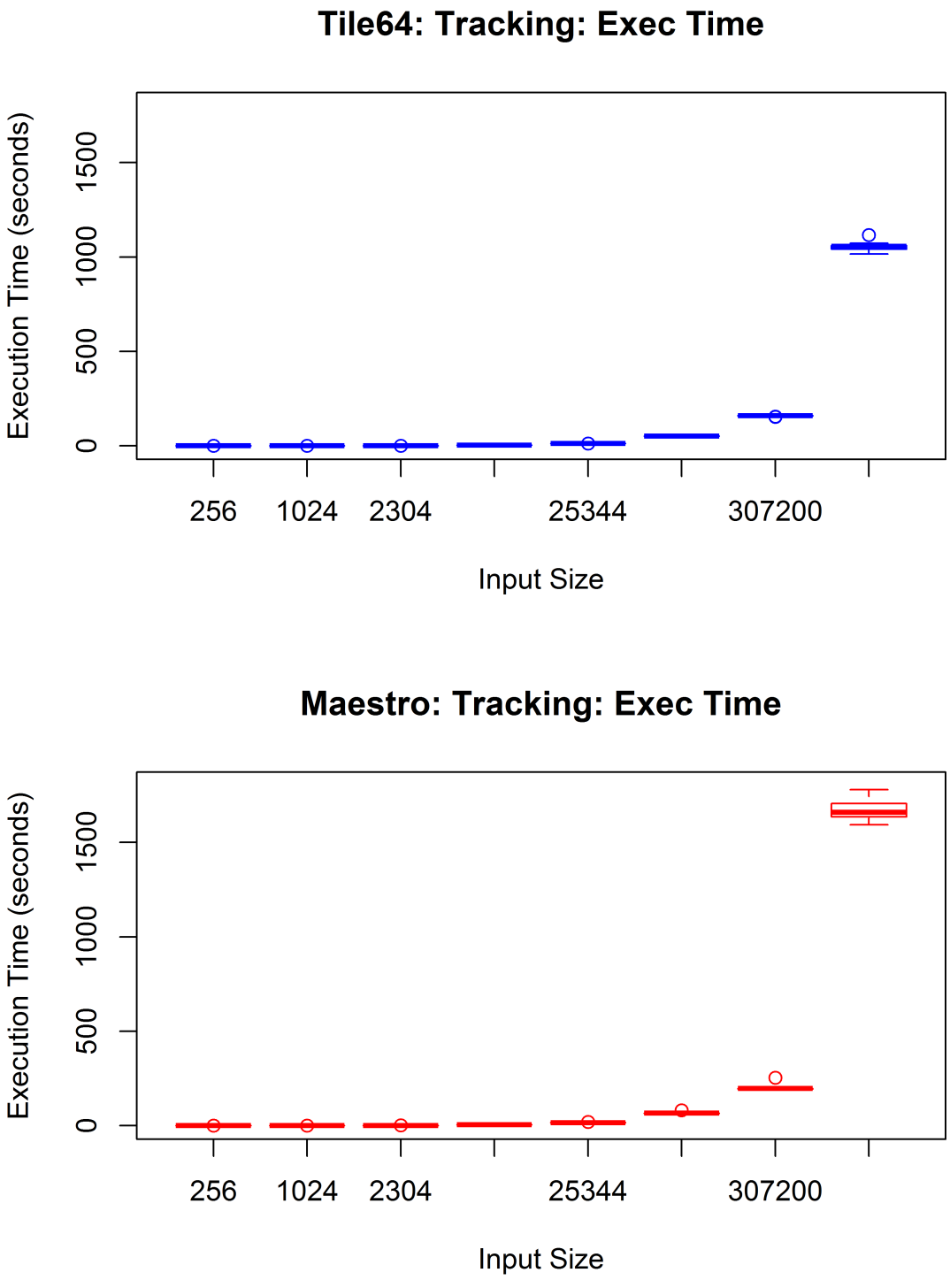


Figure B.16: Execution Time per Input Size (Benchmark: Feature Tracking)

Appendix C: MDB Bugs

The following bugs, seemingly incorrect behaviors, and unexpected results were observed on the Maestro Development Board:

1. The MDB occasionally freezes when left on for long periods of time. For example, leaving MAESTRO on for several consecutive days without doing any work on it causes it to become very slow and even unresponsive. This can be solved by either doing a hard reset of the chip, or by running some sort of intense operation on all 47 cores. The MDB will eventually produce a stack dump and resolve soft lock-ups on all the CPUs.

2. When using OProfile to profile a program using MPI, the MDB occasionally hangs, or the profiled program hangs. For example, running a series of MPI_MatrixMultiply tests on a size 500 matrix with an MPI configuration of 3×3: 8 is likely to cause the program to hang at some point. OProfile and MPI do not seem to be compatible with each other. This is probably due to the fact many of the MPI routines are not interrupt safe. [27] Since OProfile relies on non-maskable interrupts to collect profile data, there seems to be a conflict between it and MPI. The problem occurs at MPI_Finalize(), some MPI processes never complete this function call. This is probably due to unmatched MPI_Send and MPI_Recv calls. Also, the probability of a program hang seems to increase when the size of the matrices or number of MPI processes is increased.

3. Some of the default Opcontrol settings are suboptimal (*i.e.,* buffer-size=131072, buffer-watershed=[25% – 50%] * buffer-size, and cpu-buffer-size=8192). These values sometimes cause numerous buffer overflows. However, they can be reset to larger values, which in turn can reduce or even eliminate buffer overflows.

4. The MDB uses an outdated version of OProfile. When running opcontrol –version, it outputs version 0.9.3. However, that version was released in 2007. Version 0.9.8 included standard Tile64 support in 2012, and the newest version is 0.9.9, released in 2013. This may not cause any problems, but it is noteworthy.

5. When connecting to the MDB using the tile-monitor command with –net option, you must have a matching –resume. For example, tile-monitor –net 65.114.169.86 –resume. Without using the resume option, the tile-monitor will try to use an iso instead of booting off the internal ROM. As a result, the MDB will attempt to write a new image to the ROM.

6. When compiling Benchfft 3.1 with the MDB tile-cc, the benchmarks hang. However, when compiling the library files with the Tile64 tile-cc and then compiling the executables with the MDB tile-cc, some of the benchmarks will run. This could have something to with compiler itself, and maybe optimization levels.

7. In the slides provided by Opera.isi, there is a description of a profiler tool called mprof. According to the slides the mprof plugin in tile-eclipse can be accessed by opening the mProf-p view within the OSA - Parallel Analysis Tools menu. However, mprof-p did not appear in the menu. Not sure if it is just not loaded into tile-eclipse, not complied, or even in the Opera-MDE directory.

Bibliography

[1] "Rad-Hard FPGAs." http://www.atmel.com/products/other/space_rad_hard_ics/rad_hard_fpgas.aspx. Accessed: 2014-02-04.

[2] "TILEPro64 Processor Product Brief." http://www.tilera.com/sites/default/files/productbriefs/TILEPro64_Processor_PB019_v4.pdf. Accessed: 2014-02-11.

[3] M. Malone, "On-board Processing Expandable Reconfigurable Architecture (OPERA) Program Overview." PowerPoint presentation, May 2008.

[4] C. Villalpando, D. Rennels, R. Some, and M. Cabanas-Holmen, "Reliable Multicore Processors for NASA Space Missions," in *IEEE Aerospace Conference*, pp. 1–3, 2011.

[5] D. Schleher, *Electronic Warfare in the Information Age*. Artech House, Inc., 1999.

[6] A. Maini and V. Agrawal, *Satellite Technology*. John Wiley & Sons, Ltd., 2007.

[7] W. Ley, K. Wittman, and W. Hallman, *Handbook of Space Technology*. John Wiley & Sons, Ltd., 2008.

[8] D. Barnhart, "Radiation Hardening by Design of Asynchronous Logic for Hostile Environments," in *IEEE Journal of Solid-State Circuits*, vol. 44, pp. 1619–1620, 2009.

[9] J. Kleinberg and E. Tardos, *Algorithm Design*. Pearson Education, Inc., 2006.

[10] A. Oppenheim and R. Schafer, *Discrete-Time Signal Processing*. Pearson Higher Education, Inc., 2010.

[11] J. Hennessy and D. Patterson, *Computer Architecture*. Elsevier, Inc., 2012.

[12] NASA, *NASA Space Technology Roadmaps and Priorities*. The National Academies Press, 2012.

[13] R. Szeliski, *Computer Vision*. Springer-Verlag, 2011.

[14] K. Rao, D. Kim, and J. Hwang, *Fast Fourier Transform*. Springer Science & Business Media, 2010.

[15] M. Cabanas-Holmen, "Predicting the Single-Event Error Rate of a Radiation Hardened by Design Microprocessor," in *IEEE Transactions on Nuclear Science*, vol. 58, pp. 2726–2727, 2011.

[16] M. Bancroft, "World of MAESTRO." PowerPoint presentation, June 2013.

[17] K. Mighell, "MAESTRO class." PowerPoint presentation, June 2013.

[18] R. Inkol, C. Wilson, and M. Eidus, "Applications of Performance Benchmarking to the Development of Signal Processing Systems Based on Personal Computer Technology," *IEEE*, 2006.

[19] C. Brunelli, R. Airoldi, and J. Nurmi, "Implementation and Benchmarking of FFT Algorithms on Multicore Platforms," *IEEE*, 2010.

[20] K. Mighell, "Benchmarking the CRBLASTER Computational Framework on a 350-MHz 49-core Maestro Development Board," in *Publications of the Astronomical Society of the Pacific*, 2012.

[21] K. Mighell, "CRBLASTER: Benchmarking a Cosmic-Ray Rejection Application on the 49-core Maestro Processor," in *Infotech@Aerospace*, no. AIAA-2011-1448, The American Institute of Aeronautics and Astronautics, 2011.

[22] J. Walters, R. Kost, and K. Singh, "Software-Based Fault Tolerance for the Maestro Many-Core Processor," *IEEE*, 2011.

[23] "MPI Parallel Matrix Multiplication." http://cs.hofstra.edu/~cscccl/csc145/imul.c. Accessed: 2014-02-20.

[24] "The San Diego Vision Benchmark Suite." http://parallel.ucsd.edu/vision/. Accessed: 2014-02-11.

[25] "Standard Performance Evaluation Corporation: SPEC MPI2007." http://www.spec.org/mpi/. Accessed: 2014-02-04.

[26] R. Jain, *The Art of Computer Systems Performance Analysis*. Digital Equipment Corporation, 1991.

[27] "MPI Interrupt Routines." http://www.mcs.anl.gov/research/projects/mpi/www/www3/. Accessed: 2014-02-12.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
27–03–2014	Master's Thesis	Oct 2012–Mar 2014

4. TITLE AND SUBTITLE

Characterizing and Optimizing the Performance of the MAESTRO 49-core Processor

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Mote, Eric W., Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

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

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

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENG-14-M-55

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

Intentionally left blank.

10. SPONSOR/MONITOR'S ACRONYM(S)

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

As space-based imagery-intelligence systems become increasingly complex, processing units are needed that can process the extra data these systems seek to collect. However, the space environment presents a number of threats, such as ambient or malicious radiation, that can damage and otherwise interfere with electronic systems. There is a need, then, for processors that can tolerate radiation-induced faults, and that also have sufficient computational power to handle the large flow of data they encounter.

This research investigates one potential solution: a multi-core processor that is radiation-hardened and designed to provide highly parallelized MIMD execution of applicable workloads. A variety of benchmarking programs are used to explore the capabilities of this processor. Additionally, the source code is modified in an attempt to enhance the processor speed and efficiency; the consequent improvements in performance are documented.

15. SUBJECT TERMS

MAESTRO; radiation-hardening; multi-core processor; benchmarking

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. Ken Hopkinson (ENG)
U	U	U	UU	88	19b. TELEPHONE NUMBER *(include area code)* (937) 255-3636 x4579 kenneth.hopkinson@afit.edu