RISC-V Benchmarking for Onboard Sensor Processing

Michael J. Cannizzaro, Evan W. Gretok, Alan D. George
NSF Center for Space, High-performance, and Resilient Computing (SHREC)
University of Pittsburgh
Pittsburgh, PA, USA
{michael.cannizzaro, evan.gretok, alan.george}@pitt.edu

2021 IEEE Space Computing Conference (SCC) | 978-1-6654-2400-4/21/$31.00 ©2021 IEEE | DOI: 10.1109/SCC49971.2021.00013

***Abstract*—When designing embedded systems, especially for space-computing needs, finding the ideal balance between size, weight, power, and cost (SWaP-C) is a primary goal in the processor selection process. One variable that can have a significant impact on the tradeoffs between performance and power consumption is the processor architecture. Widely adopted architectures such as the ARM Cortex-A series have gained popularity due to their favorable combination of high performance and low power consumption. The RISC-V architecture presents a compelling alternative in part due to its modular instruction set, collaborative development approach, and open-source nature. The recent introduction of a RISC-V processor in the Microchip PolarFire SoC enables performance and power consumption comparisons with competing architectures using application and kernel benchmarks. For application benchmarking, this research employs several image-processing applications, including a histogram equalizer, Sobel filter, and image tiler, to describe real-world device performance. To gain additional insight into a processor's architectural characteristics, kernel benchmarks that perform common operations in sensor processing such as matrix multiplication and convolution are used. In addition, the CoreMark synthetic benchmark suite is used to help quantify overall performance. This study considers several architectures and space-grade computer facsimiles, including the ARM Cortex-A9 SHREC Space Processor, the ARM Cortex-A53 Boeing High Performance Space Computing platform, and the Power e5500 BAE Systems RAD5545 processor. Both single- and multi-core performance are considered. The PolarFire SoC achieves approximately 3.13 CoreMarks per MHz and 15.63 CoreMarks per milliwatt, demonstrating competitive performance and power consumption characteristics under single-threaded workloads. However, RISC-V presents mixed results in kernel and application benchmarks incorporating multiprocessing, with execution times that are average at best. Additionally, while matrix multiplication and addition yield high parallel efficiencies, matrix convolution and transpose are less efficient. Dynamic energy consumption results for the PolarFire SoC were generally average, but the platform does achieve significant reductions in dynamic energy consumption during increased parallel workloads in some tests. Dynamic energy consumption variability was also very low for the PolarFire SoC during most benchmarks. While the RISC-V architecture does not present ideal benchmark results, it provides a competitive balance between performance and power consumption, with future extensions to the instruction set only further enabling its potential for space applications.**

***Index Terms*—Benchmark, Processor Architecture, RISC-V, Sensor Processing, SoC, Space Computing**

This research was supported by the NSF Center for Space, High-performance, and Resilient Computing (SHREC) industry and agency members and by the IUCRC Program of the National Science Foundation under Grant no. CNS-1738783.

I. Introduction

Selecting an embedded processor for space-computing needs involves the careful consideration of multiple factors, including size, weight, power, and cost, also known as SWaP-C. Another important requirement is the desired performance of the final system. This selection process typically considers a set of trade-offs; a highly performant system may also have a power consumption too great to be practical, or it may be too expensive to be realistically deployed. In onboard sensor data processing for space platforms, these trade-offs are amplified by the requirements and restrictions associated with space computing. In this environment, energy is a finite resource and flight systems must therefore be as power-efficient as possible. Conversely, given the growing resolution and data rate of sensors for imaging and data acquisition, processors that maximize performance are often required to meet real-time computing constraints. With these factors in mind, given a mission's power consumption restrictions and/or performance requirements, developing a system that is both high performance and low power while also meeting budget limitations is a difficult goal to achieve.

The recent emergence of the RISC-V architecture in commercially available processors presents a unique solution to this problem. RISC-V is a relatively new architecture compared to industry-standard solutions such as ARM or Power. It also presents an unconventional approach to computer architecture. While alternatives such as ARM are proprietary and require expensive licensing fees for use and modification rights, RISC-V is completely open-source and free to use. Its design methodology is also open and collaborative, allowing anyone to contribute to development. The architecture's creators even claim RISC-V to have both superior performance and superior power consumption characteristics compared to the ARM architecture [1]. The introduction of a commercially available RISC-V processor in the Microchip PolarFire SoC allows for the evaluation of both performance and power consumption of the RISC-V architecture.

This study considers the RISC-V architecture in onboard sensor data processing applications and evaluates its performance and power consumption characteristics. Comparative platforms containing the ARM Cortex-A9, ARM Cortex-A53, and Power e5500 architectures are also evaluated, representing

978-1-6654-2400-4/21/$31.00 ©2021 IEEE
DOI 10.1109/SCC49971.2021.00013

a wide variety of space-capable systems in use today: the NSF Center for Space, High-performance, and Resilient Computing (SHREC) Space Processor (SSP), the Boeing High-Performance Space Processing (HPSC) platform, and the BAE Systems RAD5545, respectively [2]–[4]. Compared to these systems, the suitability of the RISC-V architecture for use in a space processing context is discussed in terms of both single-core and multi-core processing capabilities.

This paper is organized into the following sections. Section II provides a detailed background of existing architectures used in flight systems, platforms used for comparison in this study, benchmarking methods performed, and related literature. Section III gives details on the methodology for obtaining performance and power consumption results. Section IV presents the results collected and an analytical discussion of the findings. Finally, section V demonstrates conclusions made about this research as well as future avenues of exploration.

II. Background

The following sections discuss relevant information regarding the processor architectures and benchmarking tools that are the subject of this study. Space platform facsimiles and their specifications are outlined. Finally, an overview of literature related to this research is also provided.

A. Architectures in Flight SoCs

A wide variety of processor architectures exist in modern computing and are used in a range of devices. However, when considering flight-proven systems, the ARM and Power architectures are the most prevalent. The ARM Cortex-A9 processor was released in 2008 and has extensive flight heritage, with an example being within the CHREC Space Processor (CSP) [2]. The CSP has been used successfully on multiple International Space Station missions, including the Space Test Program - Houston 5 - CSP (STP-H5-CSP) and STP-H6-SSIVP (Spacecraft Supercomputing for Image and Video Processing) [5]. CSP uses the Xilinx ZYNQ XC7Z020 SoC, which contains a 32-bit dual-core ARM Cortex-A9 processor operating with a maximum clock frequency of 667 MHz and supporting the Armv7-A instruction set [6] [7]. An upgraded development based on CSP, called the SHREC Space Processor (SSP), features the same ARM Cortex-A9 processing system as CSP [8]. To represent the ARM Cortex-A9 architecture present in both CSP and SSP in this study, the TUL PYNQ-Z2 development board is used, which features the Xilinx XC7Z020 SoC operating at 650 MHz [9].

The ARM Cortex-A53 processor was released in 2013 and is planned to be incorporated in flight systems such as the High-Performance Spaceflight Computing (HPSC) processor, which was being developed by Boeing in collaboration with the NASA Jet Propulsion Laboratory (JPL) and NASA Goddard Space Flight Center (GSFC). HPSC was planned to incorporate two interconnected 64-bit quad-core ARM Cortex-A53 processors operating with a maximum clock frequency of 800 MHz and supporting the Armv8-A instruction set [3] [10]. To represent the HPSC processor in this study, the 96Boards HiKey LeMaker is used. The Hikey LeMaker uses a HiSilicon Kirin 620 SoC, which contains dual quad-core 64-bit ARM Cortex-A53 processors interconnected in the same manner as the HPSC. These processors have a maximum clock frequency of 1.2 GHz and support the ARMv8-A instruction set [11] [12]. To facilitate benchmarking results comparable to HPSC, the HiKey LeMaker's clock speed is reduced to the closest compatible frequency of 729 MHz for this study. An additional platform, the Hardkernel ODROID-C2 is also introduced to provide insight into ARM Cortex-A53 platforms of differing core counts. The ODROID-C2 uses an Amlogic S905 SoC, which incorporates a quad-core ARM Cortex-A53 processor with a maximum clock frequency of 1.54 GHz [13] [14]. To facilitate benchmarking results comparable to HPSC, the ODROID-C2's clock speed is reduced to the closest compatible frequency of 1.0 GHz.

Both the ARM Cortex-A9 and Cortex-A53 support ARM's single instruction multiple data (SIMD) instruction set extension called Advanced SIMD and also known as Neon [7] [10]. Neon is an optional extension for the Cortex-A9 but is included in the Xilinx XC7Z020 SoC. Neon is required in the Cortex-A53 and is therefore present in both the HiKey LeMaker's Kirin 620 and the ODROID-C2's Amlogic S905 SoCs [8].

The Power architecture was first introduced in the early 1990s through a collaboration between Apple, IBM, and Motorola and has since evolved into an open standard managed by the OpenPOWER Foundation [15] [16]. The Power architecture is especially significant due to its prevalence in critical space missions, with an example being the RAD5545 radiation-hardened SoC by BAE Systems [4]. Predecessors of the RAD5545, the RAD6000 and RAD750, also use the Power architecture and have had success in missions including the Spirit and Opportunity Mars Exploration Rovers, the Curiosity rover, and the Perseverance rover [17]–[21]. As of July 2020, the first RAD5545 processors have been delivered to Lockheed Martin for use in software-defined radio systems [22]. The RAD5545, which is a subject of this study, contains four 64-bit RAD5500 processing cores, which are radiation-hardened variants of Freescale's QorIQ Power e5500 cores [4] [23] [24]. The RAD5545 supports Power architecture v2.03 and has a clock speed of 466 MHz [24] [25]. To represent the RAD5545, the Freescale P5040 is used in this study, which incorporates four Power e5500 cores at a clock speed of 2.267 GHz [26]. Altering the clock frequency in the P5040 was nontrivial on the platform used this study, and therefore scaling of results to reflect the RAD5545's 466 MHz will be performed where appropriate.

B. The RISC-V Architecture

The RISC-V processor architecture represents a departure from conventional instruction set architecture (ISA) design methodologies mainly due to its open-source ISA and its collaborative approach to specification development. RISC-V was developed as an academic project in 2010 at the University of California at Berkeley, with key initial goals including open access to architecture specifications, ease of

implementation in a variety of instruction execution styles and hardware configurations, and suitability for use in educational settings. The architecture leverages a unique, modular design. RISC-V was designed with a small "base" ISA which cannot be changed, with additional instructions and functionalities permitted through the inclusion of extensions to the base instruction set [27]. The RISC-V Foundation, which allows both organizations and individuals to become members and contribute to architectural developments, has officially ratified 32- and 64-bit configurations of the base RISC-V ISA, labeled as RV32I and RV64I, respectively. The RISC-V Foundation has also officially ratified a number of ISA extensions for features including integer multiplication and division, atomic instructions, single- and double-precision floating point operations, control and status registers, instruction-fetch fence, and 16-bit compressed instructions. When labeling a processor's supported RISC-V base ISA and extensions, the base configuration is listed first, followed by the letters of the included extensions. The currently ratified extensions are labeled M, A, F, D, Zicsr (control and status register instructions), Zifencei (instruction-fetch fence), and C, respectively. Additional extensions for features including vector operations are still under development. The general-purpose RISC-V ISA implementation, as defined by the RISC-V Foundation, is "IMAFDZicsr_Zifencei", which is provided with a shortened label of 'G' [28].

Until recently, RISC-V devices realized in silicon have not been available for study. The release of one of the first commercially available RISC-V processors, the Microchip PolarFire SoC, allows the RISC-V architecture to be evaluated and compared to flight-capable ARM and Power architectures [29]. The PolarFire SoC contains five RISC-V cores – one RV64IMAC monitoring core for boot and configuration processes and four RV64GC application processing cores for use in an environment such as Linux, all with a maximum clock speed of 667 MHz [29]. To evaluate the RISC-V architecture inside the PolarFire SoC, we use the Microchip PolarFire SoC Icicle Kit evaluation board, which uses a Microchip PolarFire SoC model MPFS250T with a clock speed of 600 MHz [30]. Table I displays all platforms used in this exploration and their technical specifications.

C. CoreMark Benchmarks

Developed by the Embedded Microprocessor Benchmark Consortium (EEMBC), CoreMark is an industry-standard synthetic benchmarking tool for measuring and comparing the performance of a single core inside embedded systems. Test results are compiled into a single numerical score for easy comparison. CoreMark was released as an improvement to the popular benchmarking tool Dhrystone, which was developed in 1984 and was originally designed to compare computer systems as a whole [31]. The CoreMark benchmarking tool aims to provide more extensive insight into the performance of an embedded processor, while also improving the consistency of reported test results and reducing the impact of individual compiler optimizations when compared to Dhrystone [31].

CoreMark is written in C and contains five key integer operation algorithms [32]. These algorithms include linked list search and sort operations, cyclic redundancy check (CRC), matrix multiplication, and a Moore state machine that identifies numbers inside a string and uses those numbers for division operations. For more details about these algorithms and their specific implementations, CoreMark documentation and source code are available for download in the CoreMark Github repository referenced in [33].

The resulting CoreMark score is presented in units of iterations per second. When comparing platforms of differing clock speed, it is common to divide this score by the respective platform clock speed to obtain a normalized result in CoreMarks per MHz [34]. CoreMark scores that are submitted to the EEMBC are made available to the public on the EEMBC website, with platform specifications and compiler flags typically included in these reports [35].

D. SHREC SpaceBench Benchmarks

The SpaceBench benchmarking suite was developed at SHREC by Dr. Tyler Lovelly for the purpose of investigating the performance of space processors. SpaceBench is a collection of nine different kernel benchmarks, with the following four used in this study: matrix multiplication, matrix addition, matrix convolution, and matrix transpose. The benchmark suite supports computation with 8-, 16- and 32-bit integer as well as single- and double-precision floating-point datatypes. Benchmarking kernels are parallelized using OpenMP and accelerated with ARM Neon for applicable platforms. During execution, the user defines the benchmark operation to perform, the datatype to compute with, the problem size N (to create an NxN matrix), the number of OpenMP threads to execute with, and the number of iterations to perform the operation for [36]. When the benchmark is finished, a single execution time in seconds is presented, which is computed by SpaceBench as the total elapsed computation time divided by the number of iterations specified by the user.

E. SHREC GKSuite Benchmarks

Application benchmarking is used in this study to demonstrate more representative real-world performance on the considered platforms. A collection of seven image-processing apps developed at SHREC, called GKSuite, are considered. This collection includes a color search, a histogram equalizer, an image difference calculator, a Mandelbrot set fractal generator, a Sobel filter, a bilinear thumbnailer, and a tiler. The color search app conducts pixel color thresholding by Euclidean distance calculation. The histogram equalization app performs histogram equalization independently on each color channel. The image difference app calculates the percent difference between two images and outputs a delta comparison image. The Mandelbrot set fractal generator serves as an embarrassingly parallel test case for comparison. The Sobel filter conducts edge detection. The thumbnailer conducts image downsampling via bilinear interpolation. The tiler splits images into tiles of specified size for classification or other use

TABLE I
PROCESSING PLATFORMS AND SPECIFICATIONS

Platform	SoC	Architecture	Core Count	L1 Cache	L2 Cache	Memory	Clock Frequency
TUL PYNQ-Z2 (SSP Facsimile)	Xilinx XC7Z020 (28 nm)	ARM Cortex-A9	2	32 KB	512 KB	512 MB DDR3	650 MHz
Hardkernel ODROID-C2 (HPSC Facsimile)*	Amlogic S905 (28 nm)	ARM Cortex-A53	4	32 KB	512 KB	2 GB DDR3	1 GHz
96Boards HiKey LeMaker (HPSC Facsimile)**	HiSilicon Kirin 620 (28 nm)	ARM Cortex-A53	8	32 KB	512 KB	2 GB LPDDR3	729 MHz
Freescale P5040 (RAD5545 Facsimile)	Freescale P5040 (45 nm)	Power e5500	4	32 KB	512 KB	8 GB DDR3	2.267 GHz
Microchip PolarFire SoC Icicle Kit	Microchip PolarFire SoC (28 nm)	RISC-V RV64IMAC / RV64GC	1 + 4	8 KB	2 MB	2 GB LPDDR4	600 MHz

* Represents a half-chiplet HPSC (single quad-core ARM Cortex-A53)
** Represents a full-chiplet HPSC (two interconnected quad-core ARM Cortex-A53)

cases. The parallel performance of each of these apps varies and helps to demonstrate difference in parallel performance between the platforms [37].

F. Related Research

Studies surrounding both the evaluation of space-grade processor performance and the exploration of the RISC-V architecture have been performed in [25], [38], [39], and [40]. In [25], Lovelly evaluates the performance of radiation-hardened space-capable processors. The study performs its evaluation by calculating both performance in billions of operations per second and the input/output bandwidth in Gigabytes per second to compare processor and overall system performance metrics. Lovelly also discusses the impact of radiation-hardening through comparison of results to COTS equivalents, where applicable. In [38], eight RISC-V core designs, including one proprietary design, are implemented on FPGA platforms and their performance is compared. Each core varies in terms of the supported RISC-V ISA, in addition to having differing pipeline structures. Multiple benchmark tools are used in the study, including Dhrystone, Embench, and CoreMark. Conclusions from this research show that several of the evaluated open-source cores produced greater performance results than the proprietary design. In [39], a RISC-V processor supporting RV32G (the 32-bit variant of the general-purpose RISC-V configuration) is created and implemented on an FPGA. The resulting processor design is then evaluated using Dhrystone and CoreMark. In terms of CoreMarks per MHz, the RISC-V design closely matches the performance of platforms using the ARM Cortex-M3 and slightly underperforms platforms using the ARM Cortex-M4. In [40], an RV64IMC-compatible RISC-V processor realized on silicon is evaluated in terms of performance and energy consumption and is compared to previous RISC-V implementations. The performance benefits of incorporating instruction set extensions versus simply raising clock speeds are compared, and it is concluded that extensions are most effective in terms of performance versus energy consumption tradeoffs. Both space-grade processors and the RISC-V architecture have been explored extensively in previous literature. However, these related works do not incorporate an exploration of the RISC-V architecture in a space-computing context, nor do they evaluate a commercially available and Linux-capable RISC-V processor realized on silicon. Both of these topics are discussed in this study.

III. APPROACH

The following sections outline the procedures followed in this study for acquiring both performance and power consumption data. Where applicable, $make$ flags and configuration flags are provided. All necessary modifications to benchmark tools are also detailed.

A. CoreMark

To maintain fairness to all platforms, the CoreMark score database in [35] was surveyed for insight into the best choice of compiler flags to use when running CoreMark on each platform. Recommended compiler flags for the PolarFire SoC Icicle Kit evaluation board were provided by Microchip [41]. Where compiler flags were not available, some extrapolation was required. For example, while CoreMark results are published for the Cortex-A9, specifically for the Xilinx XC7Z020, there are not any published results for a Cortex-A53 processor [35]. Therefore, the recommended compiler flags for the Cortex-A9, including -march and -mcpu, were taken and altered to best suit execution on the Cortex-A53. This method was used for both the HiKey LeMaker and the ODROID-C2, which both use the Cortex-A53 processor architecture.

TABLE II
COREMARK MAKE FLAGS

Platform	*Make* Flag
PYNQ-Z2	make XCFLAGS="-O3 -march=armv7-a -mcpu=cortex-a9 -mfpu=neon-fp16 -DPERFORMANCE_RUN=1 -lrt"
HiKey LeMaker	make XCFLAGS="-O3 -march=armv8-a -mcpu=cortex-a53 -DPERFORMANCE_RUN=1 -lrt"
ODROID-C2	make XCFLAGS="-O3 -march=armv8-a -mcpu=cortex-a53 -DPERFORMANCE_RUN=1 -lrt"
Freescale P5040	make XCFLAGS="-O3 -mcpu=e5500 -DPERFORMANCE_RUN=1 -lrt"
PolarFire SoC	make XCFLAGS="-O3 -DPERFORMANCE_RUN=1 -DHAS_STDIO -DHAS_TIME_H -DUSE_CLOCK -fno-common -funroll-loops -finline-functions -falign-functions=16 -falign-jumps=4 -falign-loops=4 -finline-limit=1000 -fno-if-conversion2 -fselective-scheduling -fno-tree-dominator-opts -lpthread -DHAS_FLOAT=0 -mtune=sifive-7-series -lrt"

A similar approach was performed with the Freescale P5040, however the gcc compiler on the platform does not accept specified architectures with -march, and therefore only the -mcpu flag was changed to reflect the Power e5500 cores on the P5040 [26]. A list of the CoreMark make command flags used for each platform can be found in Table II. As shown in the table, all app test binaries were compiled with optimization level -O3 on all platforms. It should be noted that -O3 activates the -ftree-vectorize flag which will make use of the ARM Neon vector acceleration engines in the ARM Cortex-A9 and Cortex-A53 architectures. The PowerPC and RISC-V architectures tested do not have similar SIMD vector acceleration support.

Before execution, edits to the CoreMark source code were performed for the PolarFire SoC as recommended by Microchip. Loop index variables originally designated as unsigned integers in the core_portme.h file within the linux64 folder of the source code were changed to be signed integers, due to more efficient handling of signed integer loop indices by the RISC-V architecture [41]. Source files on other platforms were left unchanged. The number of iterations was not provided as an input to CoreMark, which results in CoreMark automatically choosing a number of iterations during runtime to reflect a total elapsed execution time of at least 10 seconds [33]. CoreMark results in units of iterations per second were recorded from the generated log files on each platform.

B. SpaceBench

Unused benchmarks within SpaceBench rely on the use of Basic Linear Algebra Subprograms (BLAS) to run, and these portions were removed for this study due to compilation issues on the PolarFire SoC and the P5040. The omission of BLAS does not affect the operation of the four SpaceBench benchmarks used in this study. Because the benchmark has multiple variable inputs, a shell script was created and run on all platforms to simplify data collection. A listing of the benchmarks, datatypes, problem sizes, and OpenMP threads can be found in Table III. Each test was run for 100 iterations, and the average of those iterations was recorded as the final result.

TABLE III
SPACEBENCH TEST CONFIGURATIONS

Size	Datatypes	OpenMP Threads
512 (Matrix Multiplication)	int8	1
3000 (Matrix Addition)	int16	2
2500 (Matrix Convolution)	int32	4*
3500 (Matrix Transpose)	fp32	8**
	fp64	

* Not applicable for PYNQ-Z2
** Only applicable for HiKey LeMaker

C. GKSuite

Each of the nine apps within GKSuite was written in C and parallelized using OpenMP. For parallelization, all but one app are configured for static division of image lines between threads as load is equally balanced. The exception is the Mandelbrot set application, which sees improved performance with dynamic parallelization due to the load imbalance near the center bulb of the fractal. All apps that require an image input used the same five-megapixel Earth-observation image to simulate a space computing use case. App parameters were tuned to achieve desired parallel performance and consistent timing between all apps on all platforms. Color search was conducted for blue, the RGB value (0, 0, 255), at 13%, 14%, and 15% distance thresholds. A Mandelbrot set of 1024 pixels square was generated. An input image was resized to 1600 by 1200 pixels via the thumbnailer application. The tiler was set to split the same input image into 224-pixel square tiles. Execution time was averaged over one hundred runs. Speedup and parallel efficiency calculations were made for each app and thread count.

D. Power Consumption

Power consumption was measured for all benchmarks using the same methodology. The Poniie PN2000 power meter was used to record idle power consumption before benchmark execution, and average runtime power during execution was recorded. Measurements provided by the PN2000 are accurate

TABLE IV
CoreMark Scores and Iterations

Platform	CoreMark Score (Iterations/Sec)	Iterations (Thousands)	Dynamic Power Consumption (Watts)
PolarFire SoC	1875	30	0.12
HiKey LeMaker	2065	30	0.13
PYNQ-Z2	2228	30	0.15
ODROID-C2	3420	60	0.30
Freescale P5040	6627	110	1.84

within 1% of the true power consumption [42]. The difference between idle and average runtime power consumption was taken to produce average dynamic power consumption for each test. For SpaceBench and GKSuite, which produce execution time results, the dynamic energy consumption in Joules was calculated. This was performed by multiplying the average execution time of each algorithm by the respective dynamic power consumption.

IV. Results

Performance and power consumption results and analysis are presented together in order of benchmark suite. First, single-core performance and power consumption results from CoreMark are discussed. Then, multi-core performance alongside dynamic energy consumption results are presented for SpaceBench and GKSuite.

A. CoreMark

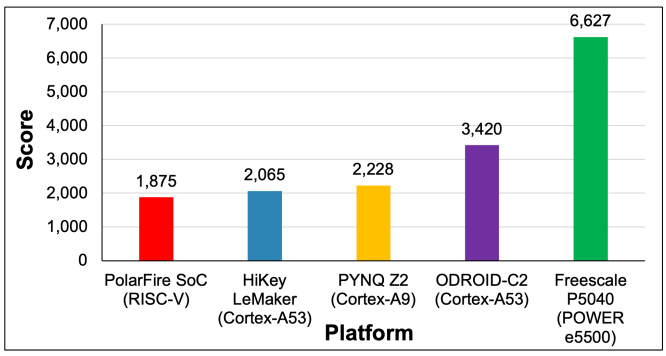

Fig. 1. CoreMark benchmarking scores.

The CoreMark scores, number of iterations performed, and dynamic power consumption measurements for each platform can be found in Table IV. Fig. 1 shows a visual representation of CoreMark scores as well. It is clear from these results that the PolarFire SoC is very competitive in terms of dynamic power consumption for this benchmark, with results comparable to the HiKey LeMaker and the PYNQ-Z2. The high dynamic power consumption in the P5040 is not surprising given its deployment in a larger scale system with higher-power supporting components. Based on observing raw CoreMark scores alone, the Freescale P5040 appears to far outperform the other platforms in this benchmark. However, normalizing around clock speed and factoring in dynamic power consumption provide an improved perspective.

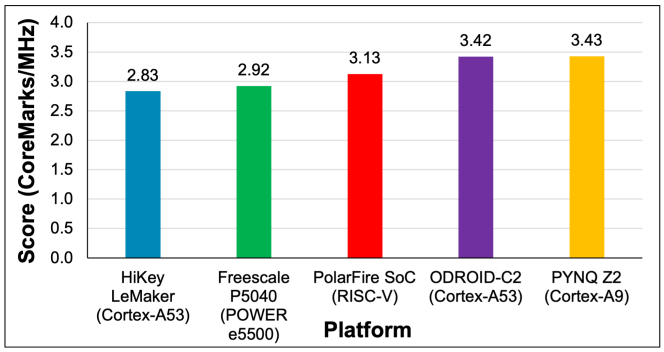

Fig. 2. CoreMark scores normalized by clock speed.

Fig. 2 displays CoreMark scores normalized by platform clock frequency, presented in units of CoreMarks per MHz. From this representation, it can be determined that performance capabilities of all platforms are fairly close to each other after accounting for each platform's differing clock speed. The Freescale P5040, while seeming to far outperform ARM and RISC-V platforms based on raw score alone, is actually quite similar to the others in a normalized CoreMark score representation. Furthermore, taking the P5040's score of about 2.924 CoreMarks per MHz and multiplying by the RAD5545 clock speed of 466 MHz, a raw CoreMark score for the RAD5545 can be estimated to be about 1362.5 iterations per second. One interesting result is the large difference in performance between the HiKey LeMaker and ODROID-C2, which both have the same ARM Cortex-A53 architecture. While the HiKey LeMaker has both the largest number of cores and the lowest dynamic power consumption compared to the ODROID-C2, this result makes sense due to architectural differences. The HiKey LeMaker's use of an interconnect between two Cortex-A53 processors likely poses as a slight disadvantage here compared to the ODROID-C2's use of a single die. Observing the results of the RISC-V PolarFire SoC shows that RISC-V performs comparably to ARM and Power in terms of CoreMarks per MHz, with scores just underneath the ARM Cortex-A9 and single-die ARM Cortex-A53, and just above the Power e5500 and the interconnected ARM Cortex-A53. These results show the RISC-V architecture is quite capable of performing competitively against ARM and Power architectures in CoreMark testing, while also maintaining a minimal dynamic power consumption.

Fig. 3 displays CoreMark scores normalized by platform dynamic power consumption, presented in units of CoreMarks per milliwatt. The data show that the PolarFire SoC has the second-highest performance at approximately 15.6 CoreMarks per milliwatt, just underneath the HiKey LeMaker's score of approximately 15.9 CoreMarks per milliwatt, providing

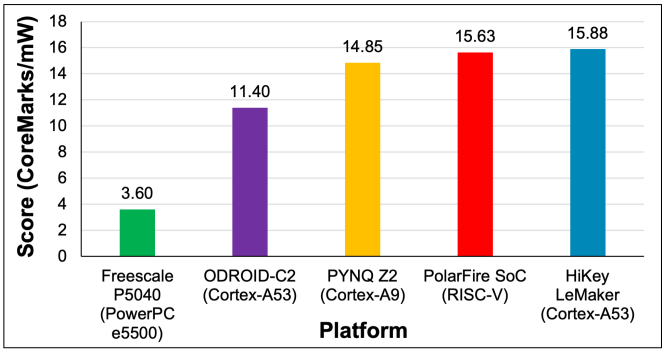

Fig. 3. CoreMark scores normalized by dynamic power consumption.

evidence of both high performance and high efficiency for RISC-V using a single core. While the Freescale P5040 power-normalized results are significantly lower than those of other platforms, the P5040 may have a larger dynamic power consumption only due to its higher-power supporting components. However, this is outside the scope of this research and presents a potential avenue of further exploration.

Performance and power consumption results from CoreMark show that RISC-V is a competitive architecture in comparison to ARM and Power for single-core computation. Single-core performance per unit of power consumed is especially notable in this section. Additional results from SpaceBench and GKSuite application benchmarking provide further insight into multi-core capabilities of the RISC-V architecture.

B. SpaceBench

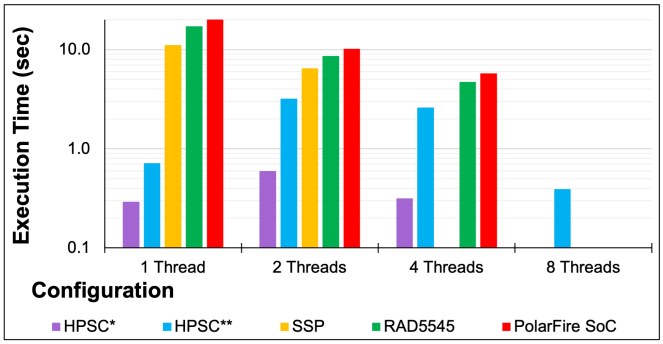

Fig. 4. Execution times of 32-bit integer matrix multiplication.

Figs. 4, 5, 6, and 7 display the execution times of matrix multiplication, addition, convolution, and transpose algorithms, respectively, using 32-bit integers on each platform. Please note the log scale in the Y-axis of each chart. Please also note the chart legends, which display names of the emulated flight platforms, with exception to the Microchip PolarFire SoC. A single asterisk (*) indicates an emulated HPSC half-chiplet (single quad-core ARM Cortex-A53), and two asterisks (**) indicate an emulated HPSC full chiplet (two interconnected quad-core ARM Cortex-A53). These calculations produce similar trends when run using single-precision

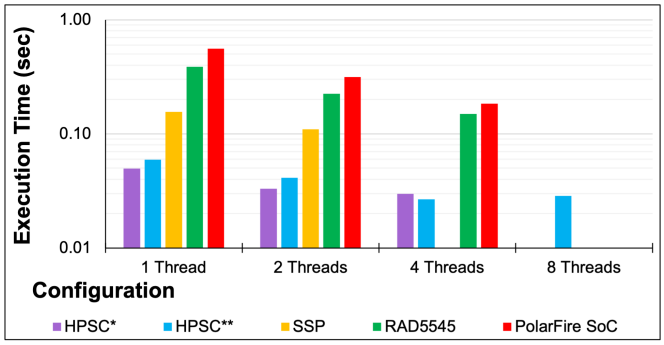

Fig. 5. Execution times of 32-bit integer matrix addition.

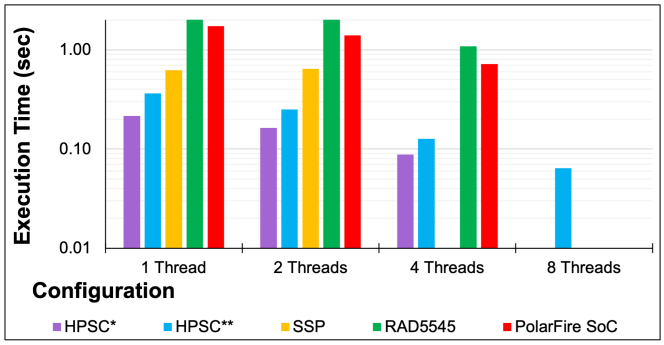

Fig. 6. Execution times of 32-bit integer matrix convolution.

floating-point (SPFP) numbers. The results and trends for SPFP calculations are available for reference in the appendix as Figs. 21, 22, 23, and 24. Execution times for the Freescale P5040 are scaled to reflect operation on a system clocked to 466 MHz to best represent and estimate the performance of the RAD5545 in these tests. It is clear from these results that both the scaled P5040 and the PolarFire SoC have the highest execution times in the set of platforms. In matrix multiplication and addition, the P5040 performs slightly better in all thread counts compared to the PolarFire. However, the opposite is true for convolution and transpose, with the PolarFire showing performance more comparable to the other platforms. This result is likely due to a combination of factors, including memory frequency and bandwidth, device cache configuration,

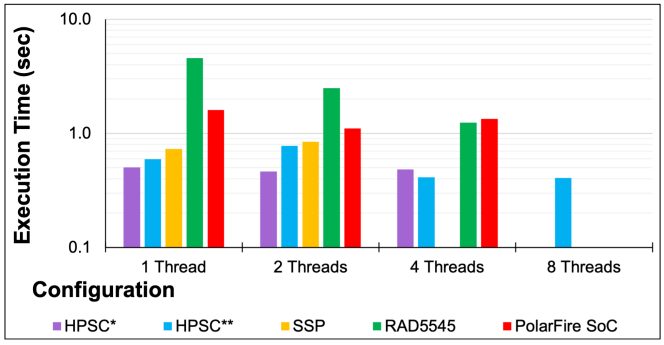

Fig. 7. Execution times of 32-bit integer matrix transpose.

and varying development board topologies. However, analysis was not performed at this depth and is therefore left for future work. Only at higher thread counts for matrix transpose does the P5040 show slightly better performance than the PolarFire SoC. Additionally, most likely due to the benefits of ARM Neon vector acceleration, performance of the PYNQ-Z2 at its maximum thread count of two threads is comparable to the PolarFire's performance at four threads. Overall, the RISC-V architecture shows average performance at best based on execution times alone.

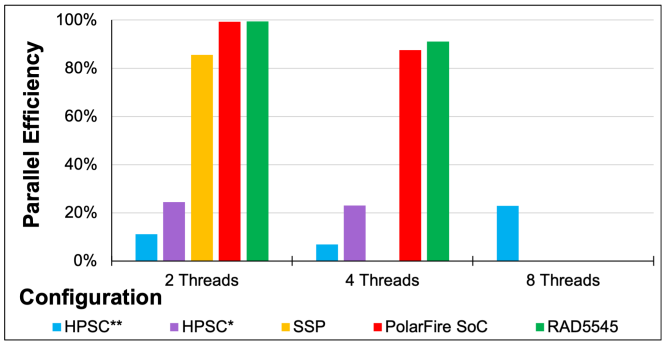

Fig. 8. Parallel efficiency of 32-bit integer matrix multiplication.

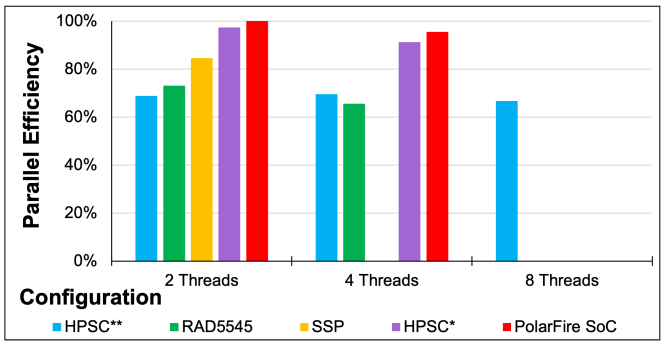

Fig. 9. Parallel efficiency of SPFP matrix multiplication.

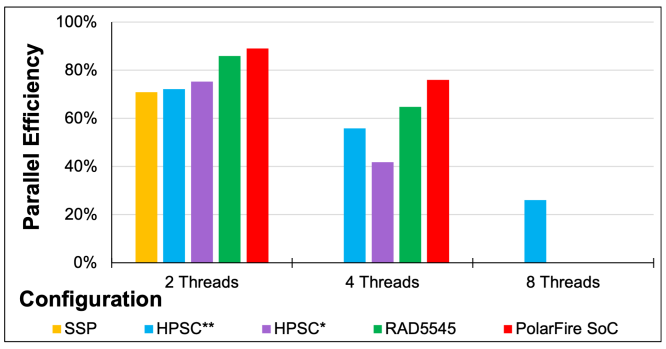

Fig. 10. Parallel efficiency of 32-bit integer matrix addition.

Figs. 8 and 9 display the parallel efficiencies of the SpaceBench matrix multiplication algorithm for 32-bit integer and SPFP numbers, respectively. As seen in the diagrams,

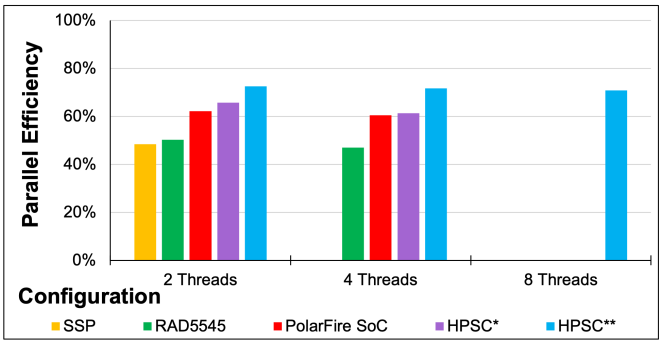

Fig. 11. Parallel efficiency of 32-bit integer matrix convolution.

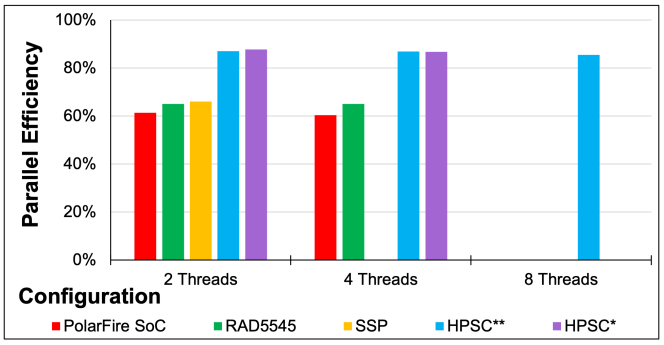

Fig. 12. Parallel efficiency of SPFP matrix convolution.

the PolarFire SoC has very high efficiencies for both integer and floating-point operations during matrix multiplication. One interesting trend seen is the fact that most of the platforms are highly efficient with either integer or floating-point operations, but not both. The PYNQ and PolarFire SoC, however, maintain high efficiency for both datatypes. Additionally, the PolarFire SoC shows only a small reduction in efficiency with an increase in threads.

Fig. 10 displays the parallel efficiencies of the SpaceBench matrix addition algorithm for 32-bit integer numbers. Similar results and trends can be found for SPFP values, and are available for reference in the appendix as Fig. 25. While efficiencies are overall lower than what was seen during matrix multiplication for all platforms, the PolarFire SoC still shows

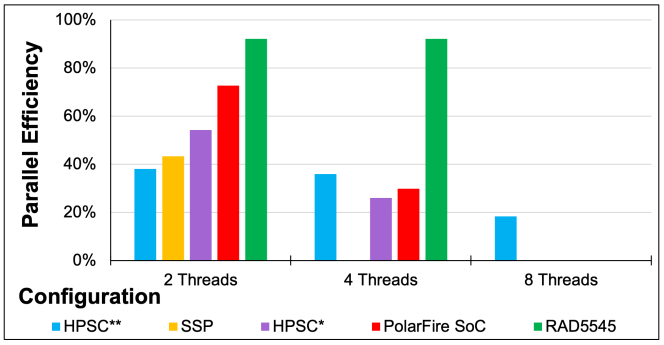

Fig. 13. Parallel efficiency of 32-bit integer matrix transpose.

high efficiency relative to ARM- and Power-based platforms during matrix addition. These results suggest that RISC-V is a competitive architecture for performing efficient parallel matrix multiplication and matrix addition operations.

Figs. 11 and 12 display the parallel efficiencies of the SpaceBench matrix convolution algorithm for 32-bit integer and SPFP numbers, respectively. As seen in the diagrams, matrix convolution produces very different trends from matrix multiplication and addition. For 32-bit integers, the PolarFire SoC has average efficiencies compared to the other platforms, with every platform maintaining consistency during thread count increases. SPFP operations produce this same level of consistency, however the PolarFire SoC has the lowest efficiency out of the group of platforms for SPFP values.

Fig. 13 displays the parallel efficiencies of the SpaceBench matrix transpose algorithm for 32-bit integer numbers. Similar results and trends can be found for SPFP, and are available for reference in the appendix as Fig. 26. All platforms except for the Freescale P5040 experience a reduction in efficiency with increased thread counts for this algorithm, however the PolarFire SoC experiences a severe reduction moving from two threads to four threads. Additionally, the P5040 maintains very high efficiencies compared to the other platforms. These results may be attributable to the more compute-bound nature of the former two operations compared to the more memory-bound nature of the latter.

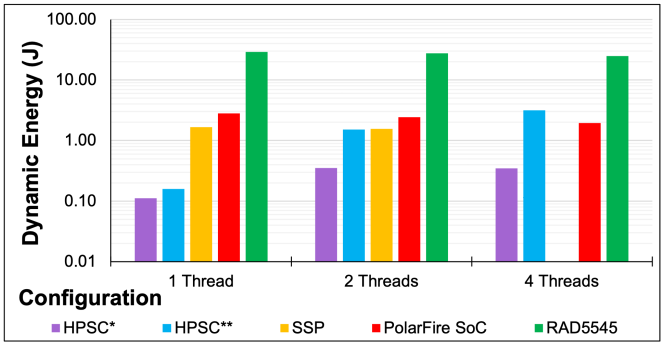

Fig. 14. Dynamic energy consumption of 32-bit integer matrix multiplication.

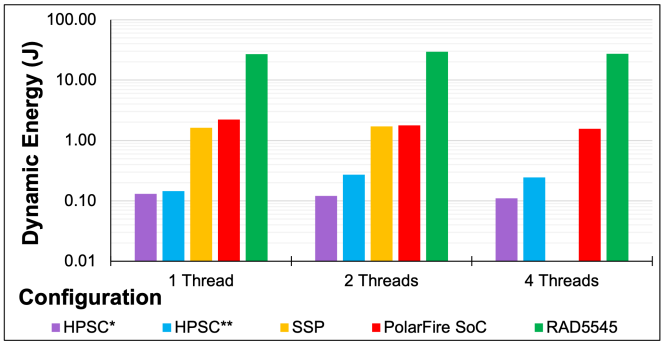

Fig. 15. Dynamic energy consumption of SPFP matrix multiplication.

Figs. 14 and 15 display SpaceBench dynamic energy consumption results for matrix multiplication of 32-bit integer and SPFP numbers, respectively. Matrix convolution results show similar results and trends and are available for reference in the appendix as Figs. 27 and 28. As seen in the diagrams, the PolarFire SoC does not have a competitive energy footprint compared to the other platforms. However, the PolarFire SoC has a very stable dynamic energy consumption between both changes in datatype and changes in thread count. The only other platforms showing these characteristics are the Freescale P5040, which has an overall higher energy consumption, and the PYNQ-Z2.

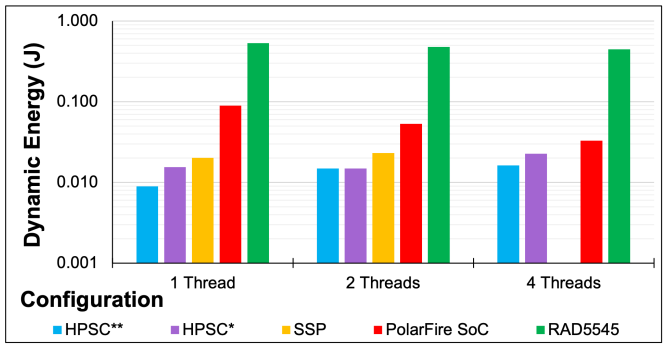

Fig. 16. Dynamic energy consumption of 32-bit integer matrix addition.

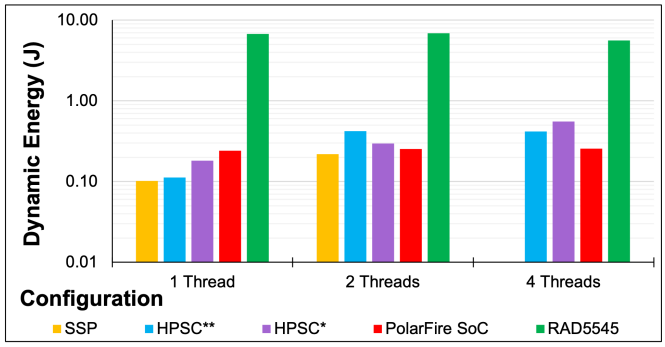

Fig. 17. Dynamic energy consumption of 32-bit integer matrix transpose.

Fig. 16 displays SpaceBench dynamic energy consumption results for matrix addition of 32-bit integer numbers. Similar results and trends can be found for SPFP values, and are available for reference in the appendix as Fig. 29. This result is particularly interesting because while the other platforms experience increases in dynamic energy consumption with an increased number of threads, the PolarFire SoC shows significant decreases in dynamic energy consumption with increased thread counts. However, it should be noted that dynamic energy consumption for the PolarFire SoC is still higher than most other platforms, even with the decreasing trend.

Fig. 17 displays SpaceBench dynamic energy consumption results for matrix transpose of 32-bit integer numbers. Similar results and trends can be found for SPFP values, and are available for reference in the appendix as Fig. 30. In terms of energy consumption, the PolarFire SoC is most competitive during matrix transpose operations. While energy consumption

is not the lowest overall, the device's low energy variability results in the PolarFire using the lowest energy at the highest thread counts for this test.

C. GKSuite

GKSuite application benchmarking execution times are included as Fig. 18. Results are divided by app and color-keyed to platform. The clear victors in raw performance are the ARM Cortex-A53 platforms, but this should come as no surprise considering their higher clock speeds. The 729 MHz HiKey LeMaker is the closest facsimile to the 800 MHz HPSC. It is postulated that the improved performance of the ARM platforms is partially owed to Neon vector acceleration, though further investigation beyond the scope of this study is required to confirm the extent of its benefit. The PolarFire and its RISC-V architecture are competitive in these real-world app performance benchmarks, especially given its only 600-MHz clock frequency. The Freescale P5040, with results scaled to the 466 MHz frequency of the BAE Systems RAD5545, demonstrates the longest execution times as the lowest-clocked platform. Additional results with per-thread trends are visible in the appendix as Fig. 31.

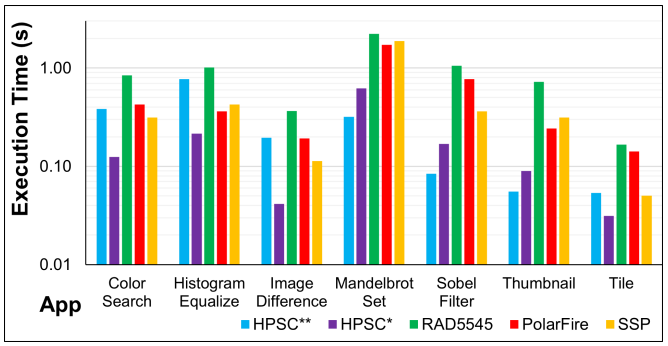

Fig. 18. Execution times of GKSuite application benchmarking.

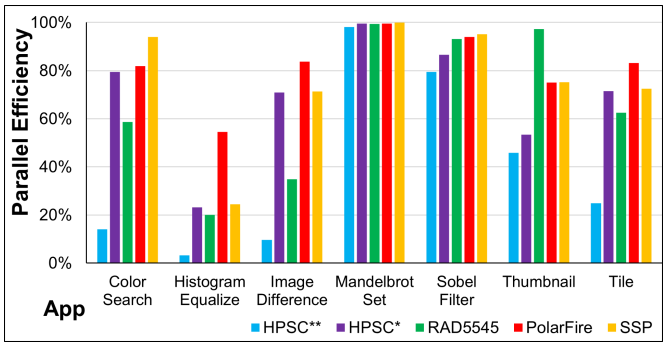

Fig. 19. Parallel efficiencies of GKSuite application benchmarking.

Parallel efficiencies from this application-benchmarking component of this research can be referenced in Fig. 19. This serves as a visualization of the parallel performance of each app on each platform. Among all platforms, the Mandelbrot set provides the most consistently ideal parallel performance results. Some of the least parallel efficient apps, however, such as the color search, histogram equalize, and tile, provide the most competitive insight between platforms. The RISC-V platform is consistently among the top two for parallel efficiency, performing the best for multiple cores in the image difference, image tile, and especially the histogram equalization app. The Power architecture shows the best and most consistent results for the Mandelbrot set and thumbnail apps. The ARM architectures tested struggle more in this comparison. The authors postulate that, while vector acceleration benefits raw performance, the overhead for vectorization across multiple threads has a higher negative impact. It's important to note that the PYNQ's two threads result in higher peak efficiencies compared to the four-threaded platforms while the HiKey's eight threads bias it to worse parallel performance despite faster execution time. Additional parallel efficiency results, including those for all thread counts, are included in the appendix as Fig. 32.

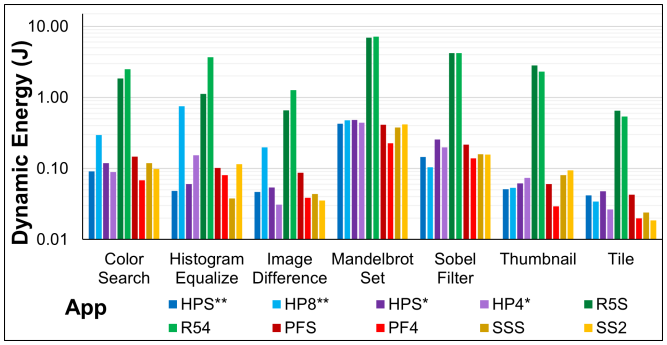

Fig. 20. Dynamic energy consumption of GKSuite application benchmarking.

Dynamic energy consumption from the application-benchmarking phase of this study can be seen in Fig. 20. The minimal power envelope of the RISC-V PolarFire is very apparent here. The RISC-V platform is among the lowest dynamic energy consumers for the color search, Mandelbrot set, thumbnail, and tile. Even more beneficial, the PolarFire sees the highest reduction in energy with parallelization, even on apps for which it did not achieve high parallel efficiency. This is seen here as a comparison between dynamic energy for the serial baseline and maximum thread counts. While the ARM platforms are very competitive on overall dynamic energy consumption, they demonstrate poorer trends for the reduction of dynamic energy consumption with parallelization. In fact, the HiKey LeMaker platform tested sees considerable increases in dynamic energy consumption for apps that parallelize poorly, especially visible for the color search, histogram equalize, and image difference. While the ZYNQ's ARM Cortex-A9 architecture is competitive in some metrics, this architecture is antiquated compared to the others. While the Power architecture appears significantly less competitive, the authors again note that this platform takes the form of a significantly larger system with higher-power supporting components. Considering only dynamic energy consumption helps to offset some of these effects.

V. Conclusions

In this study, the RISC-V architecture implemented in the Microchip PolarFire SoC is evaluated and benchmarked within a space-computing context. Comparisons were made to the ARM Cortex-A9, ARM Cortex-A53, and Power e5500 architectures prevalent in today's space-capable systems. By using industry-standard synthetic benchmarks such as CoreMark, kernel benchmarks such as SpaceBench, and application benchmarks such as GKSuite, the RISC-V architecture's performance and power consumption in both single- and multi-threaded computation scenarios have been extensively explored. Results show that the RISC-V architecture demonstrates competitive single-core performance, high efficiencies in CoreMark, and average multi-core performance results when evaluated under embedded system-focused workloads including SpaceBench and GKSuite. While highly favorable parallel efficiencies were observed during tests such as matrix multiplication and addition, average and low efficiencies were also observed in tests including matrix convolution and matrix transpose. Dynamic power and energy consumption of the RISC-V architecture also showed mixed results. Though the PolarFire SoC showed good power consumption results during CoreMark application benchmarking tests, dynamic energy observed during many SpaceBench algorithms was quite high. The PolarFire SoC did, however, show very low dynamic energy consumption variability during these tests, even under maximum thread loads.

Additional avenues of exploration also allow this research to be extended in the future. While dynamic energy consumption is a very useful result when comparing processing platforms, it is not without its inadequacies. The higher-power supporting components of the P5040 resulted in less than ideal comparisons in terms of power and energy consumption. As such, additional insight will be necessary in order to make distinct power consumption conclusions for the Power architecture as a whole. Vector acceleration is another area of further study. The RISC-V architecture does not currently have a completed vector-operation extension of the ISA, and therefore current hardware does not support vector acceleration. The addition of vector operations to RISC-V hardware could impact the results of these tests and make the architecture more competitive for complex workloads and matrix operations. Additional exploration can also be performed to better extend these results to flight-capable systems. Because this research focuses on COTS development boards, the impacts of radiation hardening and fault-tolerant design on processor performance and power consumption are not able to be analyzed. Incorporating flight-capable processing platforms in future research will allow these factors to be considered.

The results of this study show a promising future for the modern and open-source RISC-V architecture, especially in the context of space computation. It certainly presents competitive characteristics when compared against well-established ARM Cortex and Power systems. However, this is only the beginning for relatively recent ISA development. As RISC-V continues to evolve and new extensions are ratified and implemented into devices, its competitiveness in onboard sensor data processing systems calls for continued evaluation, while in the process satisfying the critical system design considerations of maximizing performance and minimizing power consumption.

References

[1] K. Asanovic and D. A. Patterson, "Instruction sets should be free: The case for risc-v," technical report, University of California at Berkeley, 2014. http://www2.eecs.berkeley.edu/Pubs/TechRpts/2014/EECS-2014-146.pdf.

[2] A. D. George and C. M. Wilson, "Onboard processing with hybrid and reconfigurable computing on small satellites," *Proceedings of the IEEE*, vol. 106, no. 3, pp. 458–470, 2018.

[3] W. Powell, "High-performance spaceflight computing (hpsc) project overview." https://ntrs.nasa.gov/citations/20180007636, Nov. 2018.

[4] BAE Systems, "Rad5545 multi-core system-on-chip power architecture processor." www.baesystems.com/en/download-en/20190327203103/1434571328901.pdf, 2017.

[5] S. Sabogal, P. Gauvin, B. Shea, D. Sabogal, A. Gillette, C. Wilson, A. George, G. Crum, A. Barchowsky, and T. Flatley, "Ssivp: Spacecraft supercomputing experiment for stp-h6," in *Proceedings of the AIAA/USU Conference on Small Satellites*, AIAA/USU, 2017.

[6] D. Rudolph, C. Wilson, J. Stewart, P. Gauvin, A. George, H. Lam, G. Crum, M. Wirthlin, A. Wilson, and A. Stoddard, "Csp: A multi-faceted hybrid architecture for space computing," in *Proceedings of the AIAA/USU Conference on Small Satellites*, AIAA/USU, 2014.

[7] ARM, *Cortex-A9 MPCore Technical Reference Manual*. ARM, 2008. https://developer.arm.com/documentation/ddi0407/latest/.

[8] Xilinx Inc, "Zynq-7000 soc data sheet: Overview." https://www.xilinx.com/products/silicon-devices/soc/zynq-7000.html#documentation, 2018.

[9] TUL Corporation, "Pynq-z2." https://www.tul.com.tw/ProductsPYNQ-Z2.html, Date Accessed: 2021-01-29.

[10] ARM, *Cortex-A53 MPCore Processor Technical Reference Manual*. ARM, 2013. https://developer.arm.com/documentation/ddi0500/latest.

[11] 96Boards, "Hikey (lemaker)." https://www.96boards.org/product/hikey/, Date Accessed: 2021-01-29.

[12] HiSilicon Technologies Co Limited, "Hi6220v100 multi-mode application processor function description," datasheet, HiSilicon Technologies Co Limited, 2014. https://github.com/96boards/documentation/raw/master/consumer/hikey/hikey620/hardware-docs/Hi6220V100_Multi-Mode_Application_Processor_Function_Description.pdf.

[13] Hardkernel Co. Ltd, "Odroid-c2." https://www.hardkernel.com/shop/odroid-c2/, Date Accessed: 2021-01-29.

[14] Amlogic Inc, "S905 datasheet," datasheet, Amlogic Inc, 2016. http://dn.odroid.com/S905/DataSheet/S905_Public_Datasheet_V1.1.4.pdf.

[15] A. Pollack, "I.b.m. now apple's main ally," *New York Times*, vol. Section D, p. 1, Oct. 1991. https://www.nytimes.com/1991/10/03/business/ibm-now-apple-s-main-ally.html.

[16] The OpenPOWER Foundation, "Openpower overview." https://openpowerfoundation.org/about-us/, 2021.

[17] J. R. Marshall and R. W. Berger, "A processor solution for the second century of powered space flight," in *19th DASC. 19th Digital Avionics Systems Conference. Proceedings (Cat. No.00CH37126)*, vol. 2, pp. 8.A.2_1–8.A.2_8, 2000.

[18] BAE Systems, "Bae systems' current processors and single board computers." https://www.baesystems.com/en/download-en/20190124214317/1434554723043.pdf, 2013.

[19] NASA, "Mars exploration rovers technologies of broad benefit: Avionics." https://mars.nasa.gov/mer/mission/technology/avionics/.

[20] NASA, "Nasa seeks high-performance spaceflight computing capabilities." https://www.nasa.gov/topics/technology/features/spaceflight-comp.html, 2013.

[21] NASA, "Mars 2020 mission perseverance rover: Brains." https://mars.nasa.gov/mars2020/spacecraft/rover/brains/.

[22] BAE Systems, "Bae systems delivers first radiation-hardened rad5545 radios." https://www.baesystems.com/en-us/article/bae-systems-delivers-first-radiation-hardened-rad5545-radios.

[23] J. R. Marshall, "Spacewire satellite usage," conference paper, BAE Systems, Manassas, VA, 2013. https://apps.dtic.mil/sti/citations/ADA579510.
[24] I. Freescale Semiconductor, "Introducing the e5500 core," white paper, NXP Semiconductors, 2010. http://www.nxp.com/docs/en/white-paper/64BTTCHNLGYWP.pdf.
[25] T. M. Lovelly and A. D. George, "Comparative analysis of present and future space-grade processors with device metrics," *Journal of Aerospace Information Systems*, vol. 14, no. 3, pp. 184–197, 2017.
[26] Freescale Semiconductor Inc, "P5040 qoriq integrated processor data sheet," datasheet, NXP Semiconductors, 2014. https://www.nxp.com/products/processors-and-microcontrollers/power-architecture/qoriq-communication-processors/p-series/qoriq-p5040-5021-64-bit-dual-and-quad-core-communications-processors:P5040?&tab=Documentation_Tab&linkline=Data-Sheet.
[27] A. Waterman, Y. Lee, D. A. Patterson, and K. Asanovici, *The RISC-V Instruction Set Manual, Volume I: Base User-Level ISA*. University of California at Berkeley, May 2011. https://www2.eecs.berkeley.edu/Pubs/TechRpts/2011/EECS-2011-62.pdf.
[28] E. A. Waterman and K. Asanovic, *The RISC-V Instruction Set Manual*. RISC-V Foundation, Dec. 2019. https://github.com/riscv/riscv-isa-manual/releases/download/Ratified-IMAFDQC/riscv-spec-20191213.pdf.
[29] Microchip Technology Inc, "Polarfire soc product overview," datasheet, Microchip Technology Inc, 2019. https://www.microsemi.com/document-portal/doc_download/1244584-polarfire-soc-product-overview.
[30] Microchip Technology Inc, "Polarfire soc icicle kit," datasheet, Microchip Technology Inc, 2021. https://www.microsemi.com/document-portal/doc_download/1245042-ug0882-polarfire-soc-fpga-icicle-kit-user-guide.
[31] A. R. Weiss, "Dhrystone benchmark: History, analysis, "scores" and recommendations," white paper, EEMBC Certification Lab, Austin, Texas, Oct. 2002.
[32] EEMBC, "Coremark pro: An eembc benchmark." https://www.eembc.org/coremark-pro/.
[33] EEMBC, "Coremark." https://github.com/eembc/coremark, 2018.
[34] S. Gal-On and M. Levy, "Exploring coreMark - a benchmark maximizing simplicity and efficacy," white paper, EEMBC, 2020. https://www.eembc.org/techlit/articles/coremark-whitepaper.pdf.
[35] EEMBC, "Scores." https://www.eembc.org/coremark/scores.php.
[36] T. M. Lovelly, T. W. Wise, S. H. Holtzman, and A. D. George, "Benchmarking analysis of space-grade central processing units and field-programmable gate arrays," *Journal of Aerospace Information Systems*, vol. 15, no. 8, pp. 518–529, 2018.
[37] E. W. Gretok, E. T. Kain, and A. D. George, "Comparative benchmarking analysis of next-generation space processors," in *2019 IEEE Aerospace Conference*, pp. 1–16, 2019.
[38] C. Heinz, Y. Lavan, J. Hofmann, and A. Koch, "A catalog and in-hardware evaluation of open-source drop-in compatible risc-v softcore processors," in *2019 International Conference on ReConFigurable Computing and FPGAs (ReConFig)*, pp. 1–8, 2019.
[39] A. Birari, P. Birla, K. Varghese, and A. Bharadwaj, "A risc-v isa compatible processor ip," in *2020 24th International Symposium on VLSI Design and Test (VDAT)*, pp. 1–6, 2020.
[40] 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, 2019.
[41] Microchip Technology Inc, "Polarfire soc cpu performance benchmarking," white paper, Microchip Technology Inc, 2020. https://onlinedocs.microchip.com/pr/GUID-8CA810D1-5A91-4A25-B308-E300BB856CF7-en-US-2/index.html.
[42] Poniie, *User Manual*. Poniie. https://poniie.com/download_manual/12.

APPENDIX

A. Additional SpaceBench Results

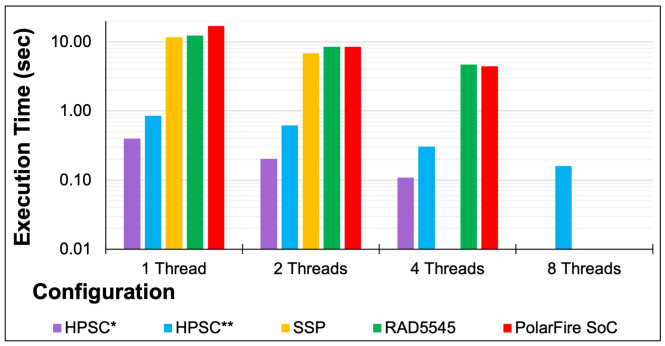

Fig. 21. SpaceBench SPFP matrix multiplication algorithm execution times.

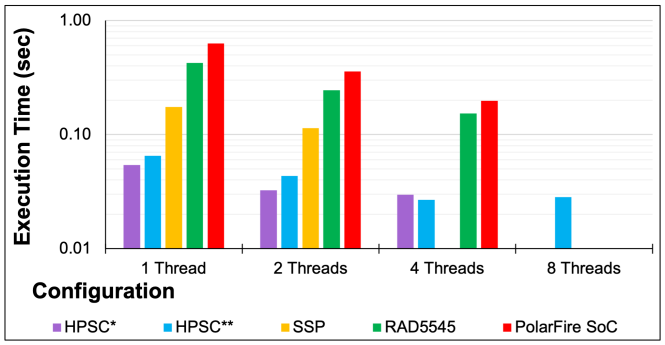

Fig. 22. SpaceBench SPFP matrix addition algorithm execution times.

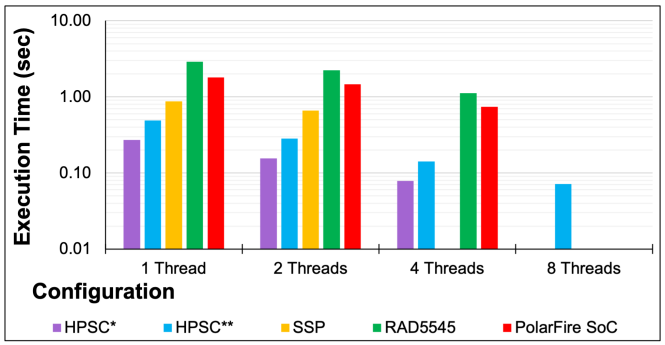

Fig. 23. SpaceBench SPFP matrix convolution algorithm execution times.

B. Additional GKSuite Results

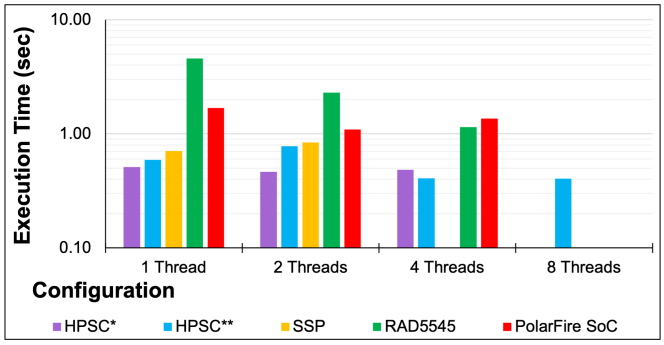

Fig. 24. SpaceBench SPFP matrix transpose algorithm execution times.

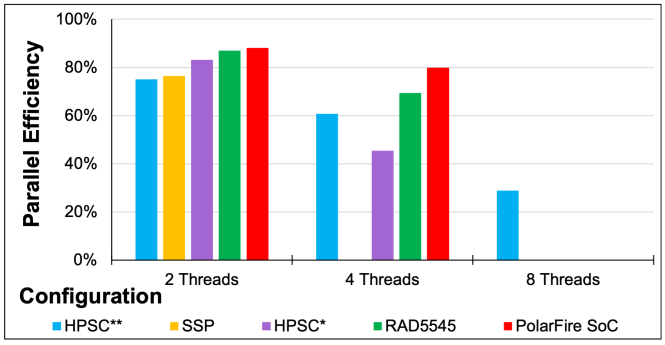

Fig. 25. SpaceBench SPFP matrix addition algorithm parallel efficiencies.

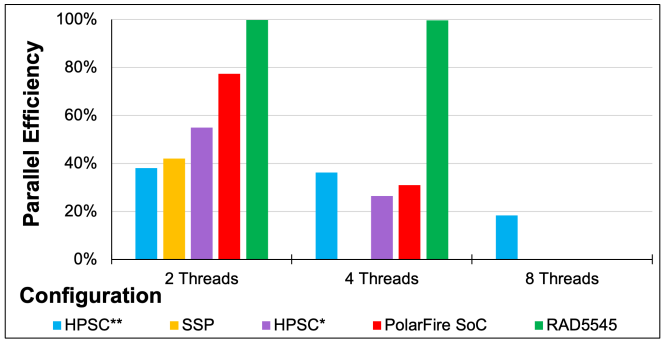

Fig. 26. SpaceBench SPFP matrix transpose algorithm parallel efficiencies.

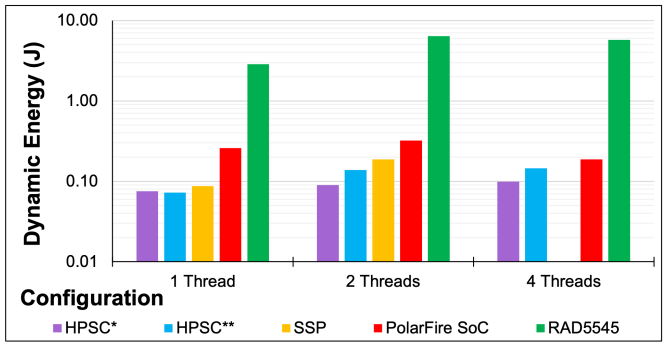

Fig. 27. SpaceBench 32-bit integer matrix convolution algorithm dynamic energy consumption.

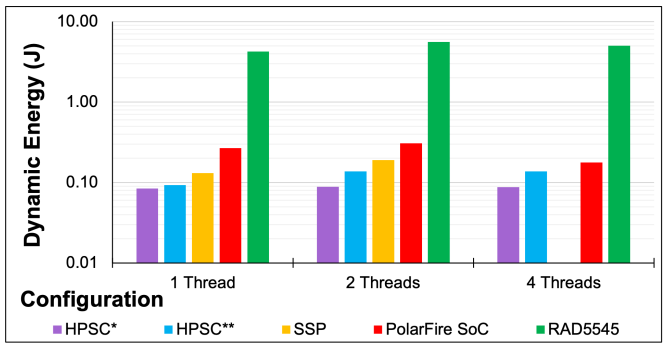

Fig. 28. SpaceBench SPFP matrix convolution algorithm dynamic energy consumption.

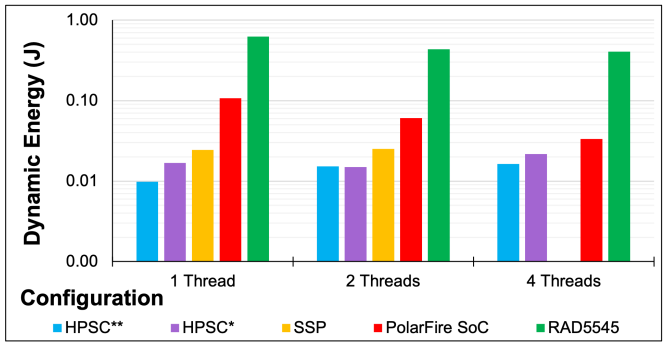

Fig. 29. SpaceBench SPFP matrix addition algorithm dynamic energy consumption.

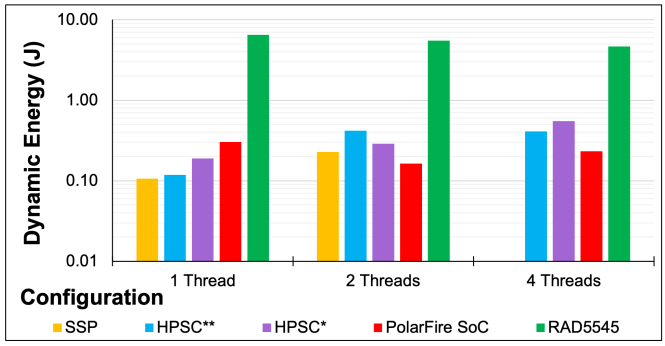

Fig. 30. SpaceBench SPFP matrix transpose algorithm dynamic energy consumption.

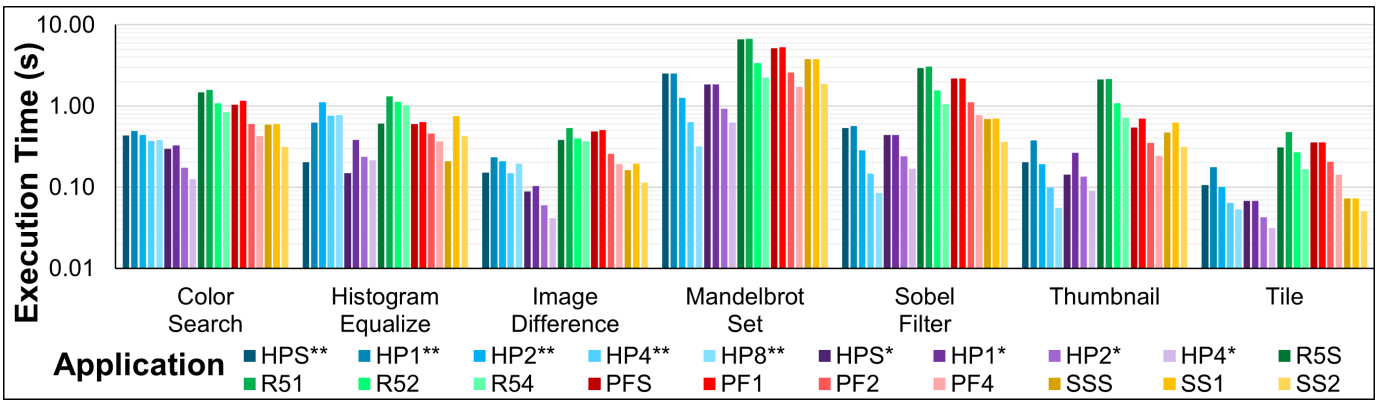

Fig. 31. Application benchmarking execution times. Legend entries represent platform name followed by number of threads or S indicating serial baseline.

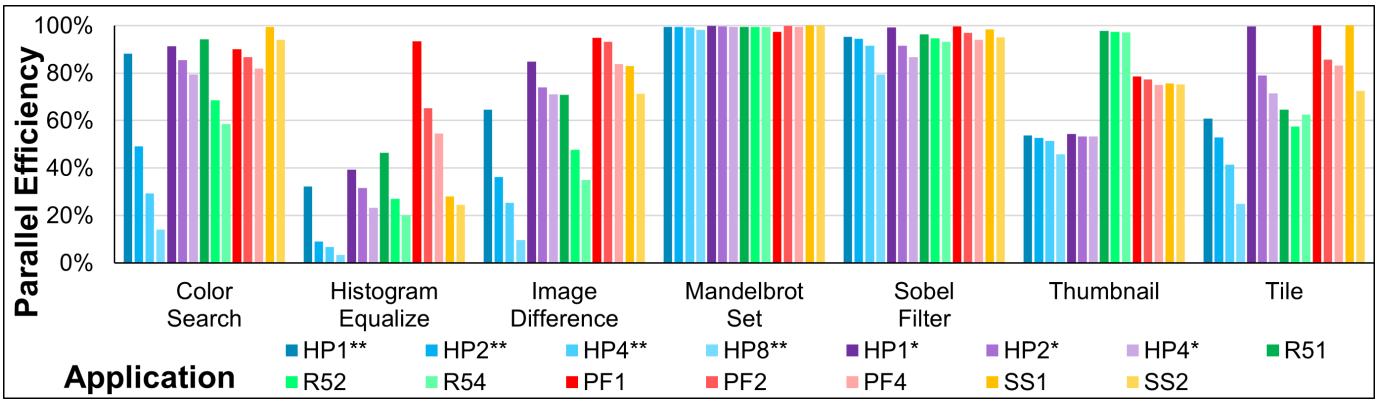

Fig. 32. Application benchmarking parallel efficiencies. Legend entries represent platform name followed by number of threads or S indicating serial baseline.

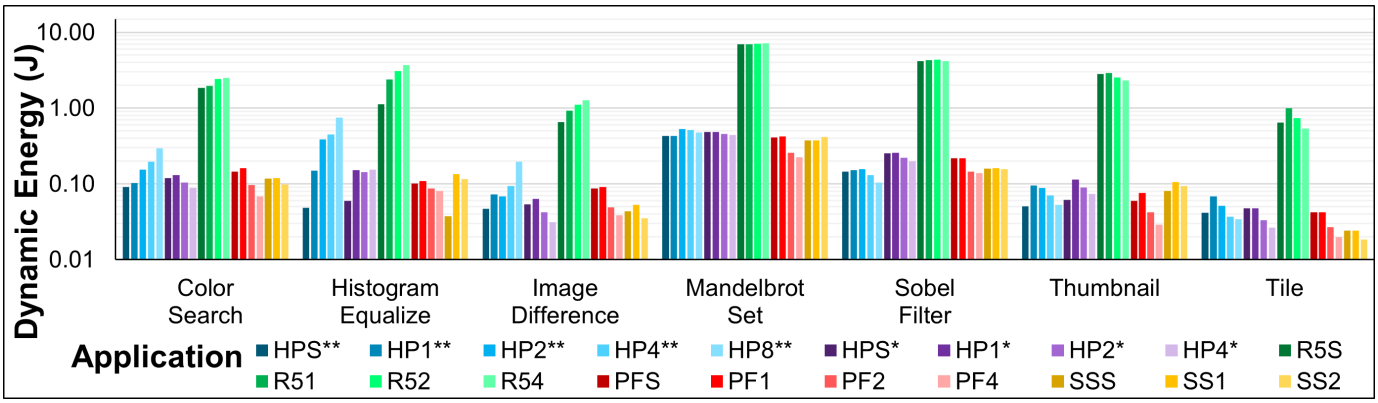

Fig. 33. Application benchmarking dynamic energy consumption. Legend entries represent platform name followed by number of threads or S indicating serial baseline.