Investigating TI KeyStone II and Quad-Core ARM Cortex-A53 Architectures for On-Board Space Processing

Benjamin Schwaller, Barath Ramesh, Alan D. George
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Department of Electrical and Computer Engineering
University of Pittsburgh
Pittsburgh, PA 15260
schwaller@pitt.edu, barath.ramesh@pitt.edu, Alan.George@pitt.edu

Abstract—**Future space missions require reliable architectures with higher performance and lower power consumption. Exploring new architectures worthy of undergoing the expensive and time-consuming process of radiation hardening is critical for this endeavor. Two such architectures are the Texas Instruments KeyStone II octal-core processor and the ARM® Cortex®-A53 (ARMv8) quad-core CPU. DSPs have been proven in prior space applications, and the KeyStone II has eight high-performance DSP cores and is under consideration for potential hardening for space. Meanwhile, a radiation-hardened quad-core ARM Cortex-A53 CPU is under development at Boeing under the NASA/AFRL High-Performance Spaceflight Computing initiative. In this paper, we optimize and evaluate the performance of batched 1D-FFTs, 2D-FFTs, and the Complex Ambiguity Function (CAF). We developed a direct memory-access scheme to take advantage of the complex KeyStone architecture for FFTs. Our results for batched 1D-FFTs show that the performance per Watt of KeyStone II is 4.5 times better than the ARM Cortex-A53. For CAF, our results show that the KeyStone II is 1.7 times better.**

I. Introduction

On-board processing is a major requirement and challenge for space missions. Sensor technology has become so advanced that sending the mass amount of data gathered to Earth for processing has become infeasible. Therefore, space processors need to be able to run the complex applications previously run on ground systems. Creating a radiation-hardened (rad-hard) processor, capable of withstanding the extremes of space, is a lengthy and costly process mandating extensive research prior to choosing a candidate architecture. Two potential architectures are the TI KeyStone II (KS2) and a quad-core ARM Cortex-A53. The KS2 is a multi-core processor with four ARM cores and eight DSP cores. The DSP cores are more suited for computing the functions considered in this paper so the ARM cores were only used to dispatch the code to the DSP cores as further explained in the methodology. DSPs can achieve efficient solutions to many complex space applications. The European Space Agency has used DSPs on their missions for many years. In 2009, they established the Next-Generation DSP (NGDSP) project in search of an architecture to meet their projected needs and replace their current rad-hard DSP [1]. In a similar effort, NASA and AFRL launched the High-Performance Spaceflight Computing (HPSC) initiative to investigate a new multi-core processor for future space missions [2]. Recently it was revealed that the main component of this architecture will be a quad-core ARM A53 processor, and hence our interest in it.

Our methodology for examining these architectures is by studying a key space kernel and a key space app on both devices. The kernel chosen is the Fast Fourier Transform (FFT), specifically the batched 1D and 2D-FFT. FFTs are prevalent in almost every signal-processing app today. The FFT transforms a signal from the time domain to the frequency domain which is useful in a wide variety of applications. FFTs are known to be memory bound functions so the memory bandwidth of each device will play a large factor in their performance.

The app chosen is the complex ambiguity function (CAF). CAF is a radar signal-processing app that can be used for geolocation. Many global positioning systems calculate the time difference of arrival (TDOA) and frequency difference of arrival (FDOA) of the same signal received at two different locations to pinpoint the origin of the signal. CAF can be employed to jointly calculate these values. Mathematically, CAF is defined as (1).

$$\mathrm{CAF}(\tau,\mathrm{f}) = \int_0^T S_1(\mathrm{t})S_2{}^*(\mathrm{t}+\tau)\mathrm{e}^{-\mathrm{j}2\pi\mathrm{ft}}\,\mathrm{dt} \quad (1)$$

S_1 and S_2 are continuous-time signals in analytic signal format., T is the integration period, τ is the time delay between the two signals, and f is the frequency offset between the signals. The * symbol denotes the complex conjugate of the variable. In discrete time the function becomes similar to the definition of a Discrete Fourier Transform (DFT) which is always applied in software via an FFT. Equation (1) can be rewritten as (2).

$$\mathrm{CAF}(\tau,k) = \mathrm{FFT}[S_1(n)S_2{}^*(n+\tau)] \quad (1)$$

The magnitude of the CAF surface produced will peak when τ and k/N (the fractional frequency difference) are equal to the TDOA and FDOA. The final step in computation is to find the maximum of the absolute value of the CAF surface. CAF contains complex computations and is highly parallelizable, making it a good function to evaluate the computational ability and memory management of the candidate architectures.

As previously noted, CAF requires that the S_1 and S_2 signals are in analytic signal format. Signals typically include both positive and negative frequencies. An analytic signal has only positive frequency components. This alteration can be accomplished using a Hilbert Transform.

This paper's contributions are twofold. The first stems from the new direct memory-access (DMA) transfer scheme we created that is tailored to the KS2 architecture. Our scheme takes advantage of the L2 and L1 on-chip memories in the DSP cores that can be partially configured as scratchpad. Pre-fetching data into these locations can accelerate execution time for batched processes. The second contribution is the benchmarking of FFTs and CAF on the KS2 architecture and on a quad-core ARM A53 architecture. For comparison, we examined the performance and performance per Watt for each system.

Section II of this paper details the KS2 and quad-core ARM A53 architectures evaluated in this research. Section III gives background information on research related to the KS2 and the complex ambiguity function. Section IV describes the methodology for mapping the functions onto each architecture and how the power consumption data was collected. Section V reviews the results collected and analyzes the key trends observed in the data. Section VI concludes the paper and summarizes the results and insights gained from this research.

II. Architecture Overview

The TI KeyStone II architecture examined in this paper is part of the K2EVM-HK board seen in Figure 1. This system has four ARM Cortex-A15 MPCore processors with eight TMS320C66x high-performance DSPs. The DSPs are housed within the C66x CorePacs which also include 32 KB of L1 and 1 MB of L2 on-chip memory. These memories can be configured to be entirely cache or part cache, part scratchpad. The EDMA peripherals enable the device to perform DMA transfers between DDR3 memory and the L1 and L2 memory space on the C66x cores. These DMA transfers do not involve the CPU so computation can run in parallel with DMA transfers without consuming extra cycles. This gives developers flexibility in tuning the architecture to their app [3].

The other architecture examined in this paper is a quad-core ARM Cortex-A53. The board chosen to examine this structure is the ODROID-C2. The ODROID-C2 is a 64-bit quad-core

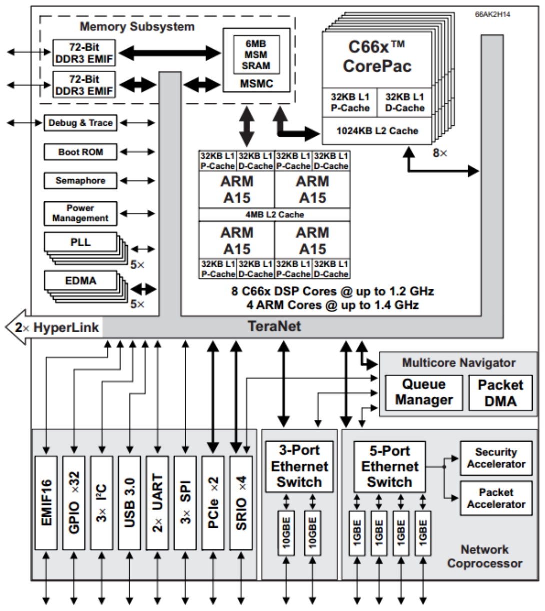


Fig 1. K2EVM-HK functional block diagram [3]

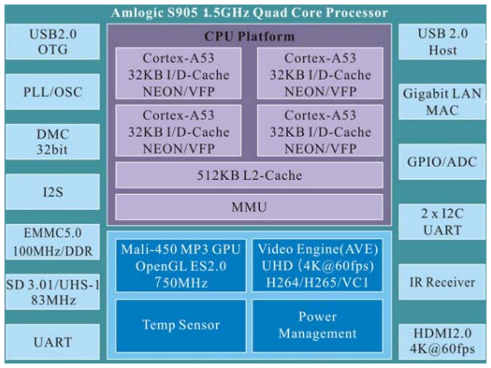


Fig 2. Amlogic S905 quad-core processor functional block diagram [4]

single board computer using an Amlogic S905 ARM Cortex-A53 (ARMv8) 1.5GHz quad-core processor whose functional block diagram is shown in Figure 2. It is self-described as one of the most cost-effective 64bit development boards available for ARM and the most advanced architecture for embedded computing. As a result, the ODROID-C2 is less expensive than the K2EVM-HK and consumes less power [4].

Our NSF Center for High-Performance Reconfigurable Computing (CHREC) has previously conducted metric studies of the K2EVM-HK and another device with quad-core A53s, the Samsung Exynos 5433 [5]. Metrics demonstrate the maximum theoretical performance in different areas of a device. Metrics for the ODROID-C2 were derived from the Exynos by changing the clock speed in the calculations and finding the DDR3 memory bandwidth in the Amlogic S905 datasheet [6].

Table I shows the Computational Density (CD), Computational Density per Watt (CD/W), Internal Memory Bandwidth (IMB), and External Memory Bandwidth (EMB) of the K2EVM-HK and ODROID-C2 boards. CD is a representation of a processor's raw performance and CD/W is this factor normalized by power used. IMB and EMB represent how fast a processor can send information between internal units (IMB) and to attached memory (EMB). CD and CD/W are calculated here for single-precision floating point datatypes. IMB for the K2EVM-HK is calculated only for the DSP cores, since those were the focus of this paper. Table I also presents the power used for CD/W calculations on each board for these metrics. 21.69 W was used for the K2EVM-HK in [5]. 8W was used for the ODROID-C2 because that was the maximum power seen used by the board with our power meter during performance data collection. The K2EVM-HK scores higher in all metrics compared to the ODROID-C2, even in CD/W, despite consuming more power.

TABLE I. K2EVM-HK and ODROID-C2 Metrics

Board	CD (GFLOPS)	CD/W (GFLOPS/W)	IMB (GB/s)	EMB (GB/s)	Power (W)
K2EVM-HK	198.40	9.15	1075.2	28.80	21.69
ODROID-C2	16.00	2.00	416.00	1.87	8.00

III. Related Research

There has been previous research on CAF on different rad-hard space processors such as the Boeing Maestro many-core CPU [7] and the BAE RADSPEED™ DSP [8]. Direct comparisons between results of these papers are difficult. Maestro data only reveals iteration time for CAF rather than operations per second as considered in this paper. The RADSPEED paper did not include the preprocessing techniques to turn the signal into analytic format presented in this paper in software and instead did the techniques on their signal generator. Additionally, input parameters for these papers were not well defined and could have been much larger than our inputs. However, the Maestro paper did present FFTW performance for 1K point FFTs which had lower performance than the architectures described in this paper. It is important to note that rad-hard processors naturally have slower clock rates and other hardware limitations to deal with the harshness of space, so comparing the KS2 and quad-core ARM to the Maestro and RADSPEED requires data about the former after hardening.

Previous research has also been conducted on optimizing other functions on the KS2 architecture using the L1 and L2 space such as [9]. This paper details the complex tradeoffs in using part of the SRAM as scratchpad. However, this paper did not attempt to perform computations as transfers were happening as ours does. [10] applied radar pulse compression on this architecture using the L2 scratchpad. However, the authors left the L1 space as all cache and did not attempt to use the complex transfer scheme described in our paper.

IV. Methodology

This section describes the method in which FFTs and CAF were executed on the KS2 and the quad-core ARM A53 architectures. The DMA scheme created for batched FFTs for the KS2 is described in this section. Additionally, the method of evaluating CAF on both architectures is detailed here.

A. *FFT Mapping on KS2*

TI provides several libraries that perform FFT functions. The DSPLIB and FFTLIB libraries were investigated to deliver a baseline performance on the KS2. The functions under investigation were complex-to-complex single-precision batched 1D-FFT and 2D-FFTs. All functions were verified with MATLAB generated inputs and outputs. These functions were dispatched from a single ARM core to a single DSP core using OpenCL. Then OpenMP was called on the single DSP core to parallelize across other DSP cores. FFTLIB was found to be the faster library, as seen in Figure 5, and thus was chosen to build a new scheme upon. A custom DMA transfer scheme, seen in Figure 3, was made to accelerate batched FFT computations using the FFTLIB FFT function. This scheme was nicknamed the "ping-pong" scheme and was built upon the FFTLIB scheme. While the FFTLIB batched functions can take advantage of the L2 SRAM space, using L1 SRAM on top of that is a new approach.

The idea behind our scheme is to accelerate computation by manually pre-fetching data into the L1 SRAM which can save cycles. The L1 memory is closest memory space to the C66x core. Placing data here enables the core to access the input data faster and reduces overall execution time by effectively

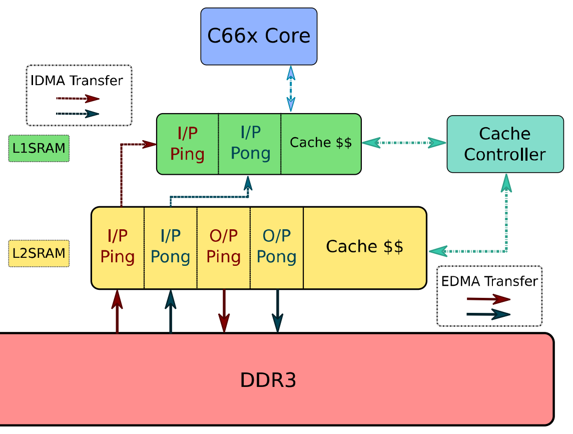


Fig 3. DMA transfer scheme for batched computations on TI KeyStone II

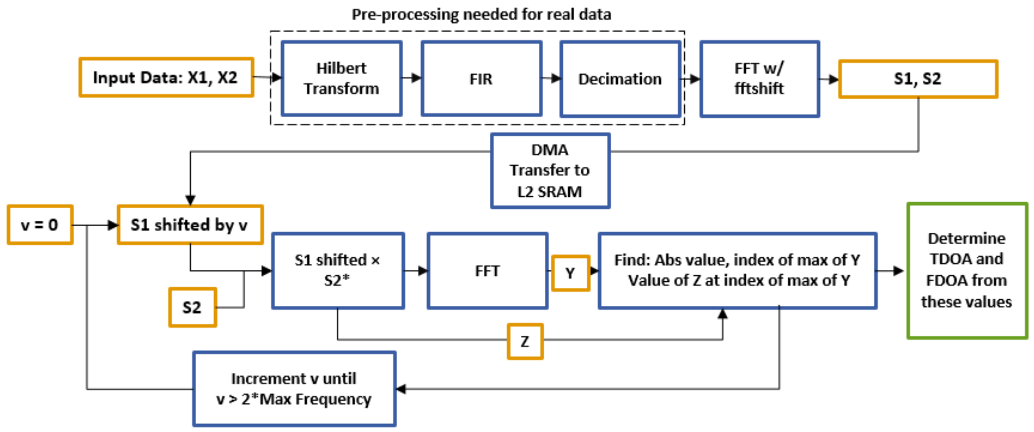
increasing the bandwidth. Four buffers are set in the L2 SRAM space (two input and two output buffers) and two buffers (both input buffers) in the L1 SRAM space along with additional room for the twiddle factors. Batched 1D-FFTs are performed in the following way. The first FFT input is loaded into the input ping buffer in L1 from DDR3 memory along with the second FFT input into L2 input pong buffer. Then the first FFT is computed from L1 ping buffer while at the same time the second input set is loaded into the L1 pong buffer from L2 pong buffer. The third FFT set is also loaded into L2 ping at this time. This scheme can be useful because the EDMA peripherals operate separate from the CPU. The FFT result is stored into the L2 output ping buffer which is then transferred back to DDR3 memory. This process repeats until all FFTs have been computed as the DMA managers ping-pong between buffers.

Since the L2 space is much larger than the L1 space the program could load, for example, 10 sets of FFTs and then 10 L1 DMA transfers and FFT computations would occur before transferring the data back from L2 to DDR3 memory. Balancing the transfer time and computation time of the data is critical to achieving maximum performance. Ideally the time it takes to transfer to the L1 and L2 space would be the same as the FFT computation time since both operations can occur simultaneously. For the final results, these transfer sizes were tuned for each problem size to obtain maximum performance.

The problem is broken up across DSP cores by having each core work on a different part of the batch of FFTs to be computed. 2D-FFTs are performed in the same way. However, there is an additional transpose that happens in the L2 SRAM space before being transferred back to DDR3 memory. This batched 1D-FFT with transpose is then computed twice to compute the 2D-FFT. The maximum FFT size studied was 1K because of the size limitation of L1 SRAM. To fit two input buffers and the twiddle factors for a 1K FFT requires 24KB of space using single precision. Note the L1 space is only 32KB so FFTs any larger would not be feasible with this scheme.

B. *FFT Mapping on ODROID-C2*

The FFTW3 library was used to calculate the FFTs. FFTW is a free software library that can perform a variety of FFT functions but the same functions described earlier were tested here. FFTW was chosen because of its wide use, high



Pre-processing needed for real data
Input Data: X1, X2
Hilbert Transform
FIR
Decimation
FFT w/ fftshift
S1, S2
DMA Transfer to L2 SRAM
v = 0
S1 shifted by v
S1 shifted × S2*
S2
FFT
Y
Find: Abs value, index of max of Y Value of Z at index of max of Y
Determine TDOA and FDOA from these values
Z
Increment v until v > 2*Max Frequency

Fig 4. CAF flow diagram

performance, and simplicity to realize in software. FFTW detects the number of OpenMP cores, so the problem was easily parallelized across the four ARM cores available [11-12]. Overall, programming for the ODROID-C2 was a much faster process because we could not access the L1 and L2 memory spaces and most optimization was done by the FFTW library.

C. CAF Mapping

CAF is computed in a similar fashion on both the KS2 and Cortex-A53 cores. The input data is based on a boat emitting a pure sine wave as it moves away from a receiver in a ship port tower. X1 is a pure sine wave at 9GHz and X2 is the same sine with a frequency offset of 4kHz and time offset of 100us and both had 16K samples. This input ideology was chosen in conjunction with our partners at Harris Corporation. The format of CAF was based off MATLAB code from [13]. The fine computation described in the paper is not used in our function because the computation needed for a nearly negligible improvement in output was deemed unnecessary. Figure 4 shows the flow diagram for CAF. The input data is real and turned into analytic format for correct computation in the pre-processing box in Figure 4. This process is composed of a Hilbert Transform, a lowpass finite response filter (FIR), and decimation by 2. This process also mimics the quadrature demodulation that would need to happen in a real system. The Hilbert and FIR have been optimized using OpenMP and the decimation with SIMD instructions. The ODROID-C2 did not have this SIMD optimization. Next, FFTs of that output are performed along with a SIMD-optimized version of MATLAB's fftshift function to create the S_1 and S_2 signals for the next block as denoted in Figure 4. Again, the ODROID-C2 did not have the SIMD optimization. For the K2EVM-HK, these signals are DMA-transferred to the L2 space, but not for the ODROID-C2, since it lacks that functionality.

Next, S_1 is shifted by a certain amount and then multiplied by the complex conjugate of S_2. The FFT of the product of these signals is performed and the maximum value of the FFT, index of this value, and value at that index of the S_1 / S_2 product determine the TDOA and FDOA. A *for* loop surrounds this procedure as S_1 is shifted by different amounts with each iteration of the loop with the shift ranging from the negative to positive maximum possible frequency, which is app specific. This loop is parallelized with OpenMP by having different cores work on different shifted S_1 signals. The final results are compared to MATLAB results for verification.

The optimized batched FFT method described above was not applicable for CAF on K2EVM-HK, since the same data set is computed in the *for* loop rather than loading new data every time. Attempts were made to ping-pong the data to L1 and back, but this step resulted in slower performance. The transfer of S_1 and S_2 to L2 SRAM did result in positive speedup however (2.3x). Additionally, the pre-processing did not occur in the L2 space because the gain in computation speed was too small to offset the additional transfer time for these functions due to the small problem size.

D. Power Measurements

Power for the apps was measured using a *Watts Up? Pro* power meter. A baseline power reading was taken before beginning any computation. For the K2EVM-HK, this resting power was 23W and for the ODROID-C2 it was 5.5W. Then the function under testing was run many times in succession so that the power would come to a steady-state value. The power used for calculating the performance per Watt is this steady-state value during computation minus the baseline power reading. In this way, we try to specify the power used solely for the computation of the function and eliminate any peripherals on the boards that are drawing power.

V. Results

This section describes the results and accomplishments of the research described in methodology, starting with the FFT results. Performance here is described with units of giga floating-point operations (GFLOPS) and mega floating-point operations (MFLOPS).

A. FFT Results

Figure 5 shows the performance of DSPLIB, FFTLIB, and FFTLIB with the ping-pong scheme for batched 1D-FFTs on a single core. Across the problem sizes investigated, FFTLIB outperformed DSPLIB which is why we chose it for a baseline library. Further, the FFTLIB with ping-pong scheme also outperformed FFTLIB across all problem sizes for single-core, 1D-batched FFTs.

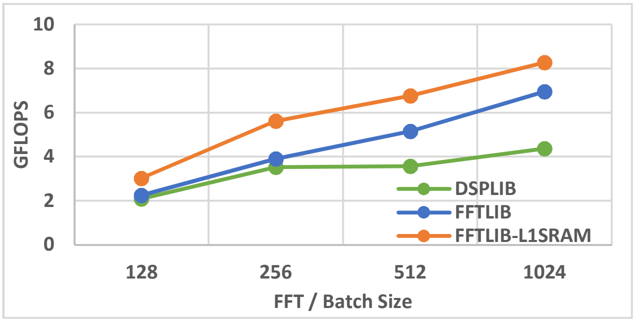


Fig 5. DSPLIB vs FFTLIB vs FFTLIB with pingpong DMA scheme for single precision batched 1D-FFTs on single core

Interesting trends are revealed as more cores are exploited for the FFTs. Figure 6 shows the performance for 1K × 1K batched 1D-FFTs. The performance benefit for both schemes decreases as the number of cores increases. This dip occurs because the FFT is a memory bound computation. The bandwidth becomes increasingly limited as more cores are used. Transfers take longer because of the limited bandwidth and the program stalls waiting for them to finish. For our ping-pong scheme, this effect is augmented since even more transfers occur resulting in a dip in performance moving from four to eight cores. Using L1 as partially SRAM also limits how much L1 cache is available which gives the cache controller less flexibility with memory management. This trend of dips in performance from four to eight cores follows for all batched 1D-FFTs. For 2D-FFTs, this trend only occurs on smaller sizes but there is still minimal gain from four to eight cores. The extra transposes at larger sizes may hide some of the transfer time behind them leading to minimal gain rather than a dip. Another trend seen in this graph is that the ping-pong scheme performs better than the base FFTLIB scheme with fewer cores. Again, this trend continues for all batched 1D-FFTs. When enough bandwidth is available the ping-pong scheme provides an advantage over the FFTLIB scheme for batched 1D-FFTs.

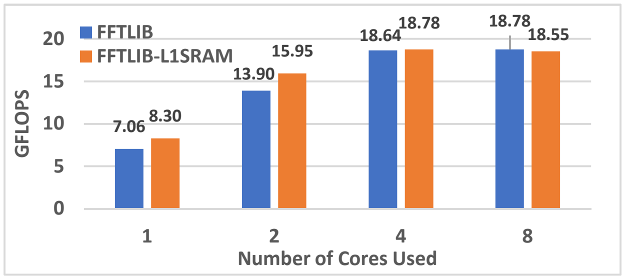


Fig 6. 1K × 1K 1D-FFT on TI K2EVM-HK

Figure 7 shows the maximum performance for 2D-FFTs. The ping-pong scheme leads to better performance than the FFTLIB scheme at larger problem sizes. As previously mentioned, as the transpose size increases more of the transfer time could be hidden. Figure 8 shows the performance and power usage for 1D batched FFTs on the K2EVM-HK and ODROID-C2. The power and performance follow the same trends on both processors, including the dip at eight cores on the K2EVM-HK. In general, as more cores are used more

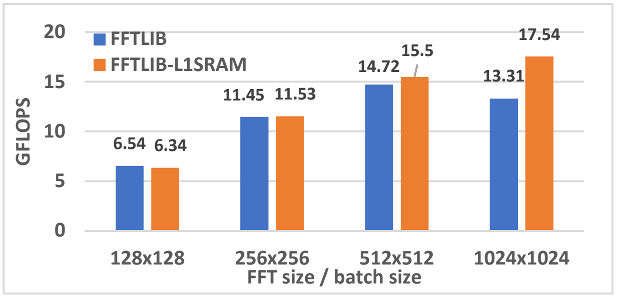


Fig 7. Maximum performance of 2D-FFTs with FFTLIB and FFTLIB with L1 transfers

power is consumed. The dip in power for four to eight cores on the K2EVM-HK happens for the same reason as the aforementioned dip in performance. When the stalls occur, there are fewer computations performed, thereby consuming less power.

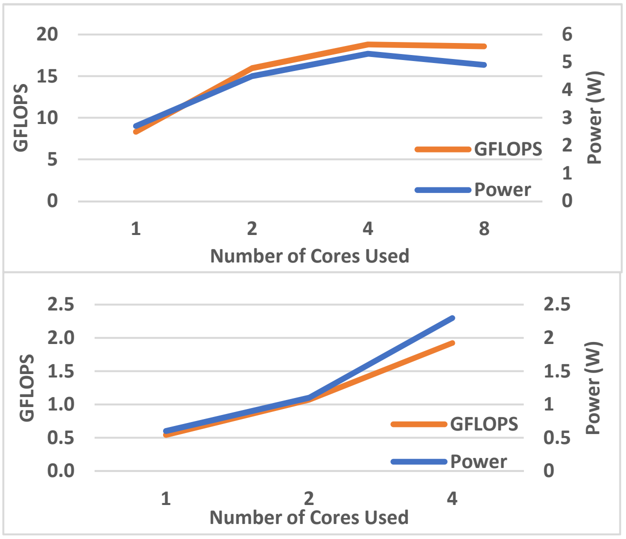


Fig 8. (a) 1K × 1K 1D batched FFT on K2EVM-HK performance and power usage, and (b) 1K × 1K 1D batched FFT on ODROID-C2, illustrating both performance and power usage

B. *CAF Results*

Table II describes the number of operations each computation contributes to CAF. FFTs comprise the majority of operations which is a large reason that CAF was chosen as a

TABLE II. CAF COMPUTATION OPERATIONS AND CYCLES ON KS2

Computation	Number of Operations	Number of Cycles at Runtime (single core)
Hilbert Transform	8,192	76,255
FIR	395,328	356,898
FFTs	1,075,200	110,984
Complex Conjugate Multiplication	116,736	63,689
Absolute Value Multiplicaiton	58,368	42,818
Other	0	187,167

case study. At runtime, the K2EVM-HK can quickly compute these FFTs, as evidenced by the nearly 10× drop between operations and cycles. DSPs in general are specially built for FFTs by including a large amount of multiplier-accumulator (MAC) units. Additionally, it is important to note in Table II that cycles allocated to other operations, such as memory management, encompass roughly 22% of the cycle count of the app. This high percentage of cycles devoted to non-computation actions is indicative of the memory-intensive nature of the CAF app.

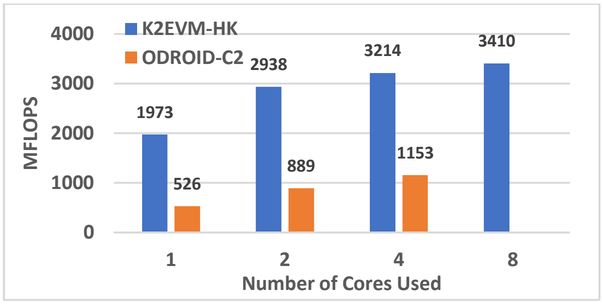


Fig 9. Complex Ambiguity Function on K2EVM-HKand ODROID-C2

Figure 9 shows CAF performance on both the K2EVM-HK and ODROID-C2. K2EVM-HK outperforms the ODROID-C2 here for a variety of reasons. The first is that the K2EVM-HK has a much larger IMB and EMB than the ODROID-C2 as seen in Table I. Since the FFT is a memory bound function having a larger memory bandwidth will lead to greater performance. Also, the KS2 architecture allows for more flexibility in its memory transfer scheme and has DSP units to handle FFTs efficiently. A summary of all benchmarks considered in this paper is presented in Figure 10. For the reasons stated above, the K2EVM-HK shows superior performance. Even though the ODROID-C2 consumed less power the lack of performance led to it still having poorer performance per Watt. It is notable that, overall, the ODROID-C2 does use less board power than the K2EVM-HK and that it was significantly easier to program the device for these functions. The difference between performances for CAF is likely smaller than the other benchmarks because the KS2 architecture was unable to use the ping-pong scheme for CAF as previously discussed in Section IV.

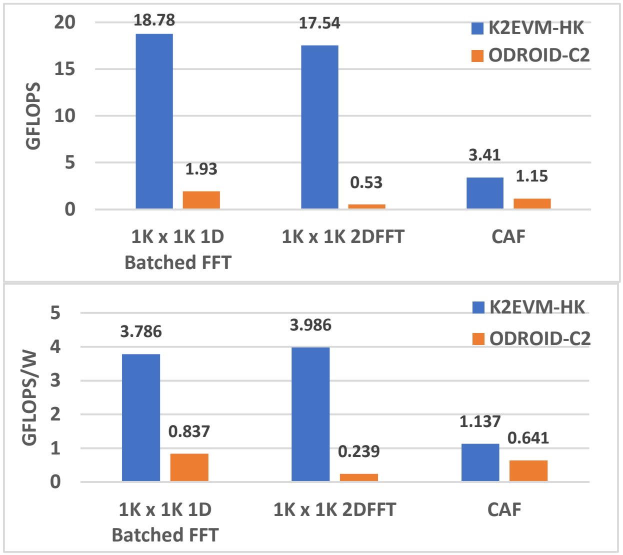


Fig 10. Maximum performance (a) and performance per Watt (b) of different benchmarks on K2EVM-HK and ODROID-C2

VI. Conclusions

In this paper, we examined the TI K2EVM-HK, which features the KS2 processor and DSP, and the ODROID-C2, a quad-core ARM Cortex-A53 CPU, to examine how well these architectures might serve future space missions. We created a custom DMA transfer scheme for the KS2 that accelerated batched FFTs. It performed particularly well when using a single core with 44% speedup over the base method for batched 1D-FFTs. This decreased to on average 6% at eight cores used. It is less successful for 2D-FFTs but once the problem size is large enough it gave positive speedup with 32% speedup for 1K × 1K 2D-FFTs. We observed that memory bandwidth plays a large part in the efficacy of this function with performance occasionally dipping from four to eight cores. This technique could be applied to other batched computations with the caveat that some manual tuning is needed with transfer sizes to attain the best performance. The FFT performance on the ODROID-C2 is always less than on the K2EVM-HKwith 9.8× worse performance for 1K × 1K batched 1D-FFTs. It also fell behind in performance per Watt by 4.5×.

Additionally, CAF was benchmarked on each device. The K2EVM-HK had a 3× faster performance and 1.7× better performance per Watt than the ODROID-C2. The performance margin between processors seen in FFT studies was smaller with CAF. There are several possible reasons for this outcome. While the K2EVM-HK has a much better EMB than the ODROID-C2, the IMB of the two devices is more comparable, with the K2EVM-HK only having 2.6× the IMB of the ODROID-C2. The CAF might require more IMB than EMB as compared to the FFTs. Additionally, the CAF input size was smaller than the 1K × 1K FFT which might have been a factor in saturating the memory of the devices. Nevertheless, the K2EVM-HK proved to be a more powerful device, but the ODROID-C2 had lower total power usage, is cheaper, and is easier to program. Additionally, the ODROID-C2 had better parallelization and did not experience the drastic reduction in performance gained as more cores were used. These factors, along with NASA and DOD goals to use multiple ARM processors together, suggest ARM processor performance can scale well in the future.

Acknowledgments

This research was supported by CHREC members and the I/UCRC Program of the National Science Foundation under Grant No. IIP-1161022. We wish to acknowledge Asheesh Bhardwaj from Texas Instruments, Josh Bruckmeyer from Harris Corporation, and Ricky Tran, from the University of Florida for their help in conducting the research presented in this paper.

REFERENCES

[1] Next generation processor for on-board payload data processing application, [Online], Available: http://spacewire.esa.int/edp-page/documents/NGDSP%20Round%20Table%20Synthesis%20V1.0.pdf

[2] G. Mounce, J. Lyke, S. Horan, W. Powell, R. Doyke,and and R. Some, “Chiplet based approach for heterogenous processing and packaging architectures.” IEEE Aerospace Conference, 5-12 March 2016.

[3] Multicore dsp+arm keystone ii system-on-chip (soc), [Online], Available: http://www.ti.com/lit/ds/symlink/66ak2h12.pdf

[4] Odroid wiki, [Online], Available: http://odroid.com/dokuwiki/doku.php?id=en:odroid-c2

[5] T. M. Lovelly and A. D. George, "Comparative Analysis of Present and Future Space-Grade Processors with Device Metrics", AIAA Journal of Aerospace Information Systems, March 2017, Vol. 14, No. 3, pp. 184-197

[6] Amlogic S905 Datasheet, [Online], Available: http://dn.odroid.com/S905/DataSheet/S905_Public_Datasheet_V1.1.4.pdf

[7] K. Singh, J.P. Walters, J. Hestness, J. Suh, C. Rogers, and S. Crago, “FFTW and complex ambiguity function performance on the maestro processor.” IEEE Aerospace Conference, 5-12 March 2011.

[8] J. Marshall, R. Berger, M. Bear, L. Hollinden, J. Robertson, and D. Rickard, “Applying a high performance tiled rad-hard digital signal processor to spaceborne applications.” IEEE Aerospace Conference, 3-10 March 2012.

[9] Y. Gao, F. Zhang, and J. Bakos, “Sparse matrix-vector multiply on the keystone II digital signal processor.” IEEE high Performance Extreme Computing Conference (HPEC). 9-11 September 2014.

[10] M. Bahtat, S. Belkouch, P. Ellaume, and P. Le Gall, “Efficient implementation of a complete multi-beam radar coherent-processing on a telecom soc.” IEEE International Radar Conference. 13-17 October 2014.

[11] M. Frigo and S. Johnson, “FFTW: an adaptive software architecture for the FFT.” IEEE International Conference on Acoustics, Speech, and Signal Processing. 15 May 1998.

[12] M. Frigo and S. Johnson, “The design and implementation of FFTW3” Proceedings of the IEEE, February 2005, Vol 93, No.2, pp.216-231.

[13] J. J. Johnson, “Implementing the cross ambiguity function and generating geometry-specific signals.” M.S thesis, Naval Postgraduate School, Monterey, California, 2001.