978-1-5090-3525-0/16/$31.00 ©2016 IEEE

# Computational and Memory Analysis of Tegra SoCs

Andrew Milluzzi, Alan George, Herman Lam
NSF Center for High-Performance Reconfigurable Computing (CHREC)
Department of Electrical and Computer Engineering
University of Florida
Gainesville, Florida 32611
Email: {milluzzi, george, hlam}@chrec.org

***Abstract*—Low-power, embedded, GPU System-on-Chip (SoC) devices provide outstanding computational performance, especially for compute-intensive tasks. While clusters of SoCs for High-Performance Embedded Computing (HPEC) are not new, the computational power of these supercomputers has long lacked the efficiency of their more traditional, High-Performance Computing (HPC) counterparts. With the advent of the Tegra K1 and X1, the efficiency debate is significantly more complex. These new devices can provide up to 500 GFLOPS of Float32 performance with a TDP of just 10 Watts. This paper investigates the current state of NVIDIA SoCs, comparing and contrasting their performance and power characteristics to high-end NVIDIA accelerators. In order to perform this analysis, we leverage two metrics, Computational Density (CD) and External Memory Bandwidth (EMB), to provide a first-order estimate and then normalize the results with Realizable Utilization (RU). RU is a measurement of device efficiency, comparing observed benchmarking results to the theoretical CD or EMB. From this analysis, we are able to uncover which computational kernels might show similar or improved performance on an HPEC cluster of NVIDIA Tegra SoCs. Based on our observations, an SoC cluster would be plausible for some power-constrained HPC applications. Furthermore, the observed performance improvement in Tegra devices suggests that a future GPU-SoC cluster could be an option for a wide range of applications.**

## I. Introduction

Embedded computing continues to see significant gains in performance. The current industry trends, focusing on low-power devices, have contributed significant advances to the field of High-Performance Embedded Computing (HPEC). Many of these advances are seen in low-power System-on-Chip (SoC) devices, running 64-bit ARM cores and a small embedded GPU. Low-power devices, such as the Raspberry Pi, Pine64, and ODRIOD, provide powerful CPU-focused platforms that can run off a USB port. Their high performance per Watt makes them an attractive compute platform. With the rise of mobile gaming and graphics processing, chip vendors have begun to include high-end graphics on their SoCs.

NVIDIA's introduction of the Tegra K1 (Kepler architecture) and Tegra X1 (Maxwell architecture) SoCs made a significant impact in the HPEC community. While the Tegra SoCs focus on mobile gaming and tablet processing, the inclusion of NVIDIA's CUDA development language made the devices accessible to developers. The latest generation Tegra X1 achieves over 500 GFLOPS of Float32 performance at just 10 Watts Thermal Design Power (TDP) [1]. Combining hundreds of CUDA cores, these HPEC SoCs have even caught the eye of traditional High-Performance Computing (HPC) developers. The low-power, high-performance characteristics of the Tegra SoCs suggest a compute efficiency similar to their HPC counterparts. Metrics suggest a cluster of Tegra SoCs could outperform NVIDIA's highest-end GPUs or the flexibility to scale to only what is needed for a given mission. In addition, the thermal and physical characteristics of many small devices could be more useful for HPEC applications.

In order for an SoC cluster to be practical, the individual devices combined must be capable of showing improved performance. First, we explore the theoretical maximum performance through metrics. This application-independent upper bound enables a first-order analysis to identify how computational or memory bandwidth of SoCs scale up to the TDP of an HPC device. Few applications will reach 100% efficiency; benchmarking a range of compute-intensive to memory-intensive kernels will provide insight into device utilization. Based on analysis of metrics, benchmarking, and utilization results, we will show that the Tegra SoCs have few architectural bottlenecks and exhibit similar performance trends to their HPC counterparts. After evaluating computational performance and memory bandwidth for several applications, Tegra SoCs show good potential for overcoming the overhead of clustering.

The following sections of this paper explore the comparison of an individual Tegra SoC versus a high-power GPU accelerator in terms of performance gains and power savings. Through metrics, benchmarking, and efficiency analysis of computational and memory-bandwidth characteristics of NVIDIA SoC and HPC devices, this paper seeks to explore the feasibility of an SoC cluster as a finer-grain power and performance solution for HPEC applications.

## II. Related Work

The complex architecture of an SoC makes analysis a difficult task. Existing work in device metrics presents a methodology of comparing theoretical device performance by computational units. Computational Density (CD), which is presented in giga-operations per second (GOPS), evaluates the maximum sustained amount of instructions issued by a device. The total device CD is the frequency multiplied by the sum of its processing cores, normalized for cycles per instruction (CPI) as seen in Eq. 1. External Memory Bandwidth (EMB) is presented in gigabytes per second (GB/s) and measures the total memory bandwidth of a device as seen in Eq. 2. [2], [3]

$$CD = Frequency \times \sum_i \frac{Core_i}{CPI_i} \tag{1}$$

$$EMB = \sum_{i} Frequency_i \times Bus\ Width_i \times Port_i \tag{2}$$

In order to determine the efficiency of a given application, we leverage and expand upon Realizable Utilization (RU), a process established in [4] and [5]. RU leverages benchmarking to observe application performance and relate those results to metrics. RU seeks to establish a spectrum of utilization, enabling a finer-grain understanding of the device. The work presented in [6] seeks to investigate real-world performance in a similar manner. However, [6] focuses on traditional GFLOPS and memory bandwidth with a binary selection of kernels: memory-bound and compute-bound. Their achievement in binary classification is impressive, but the labor involved in this approach makes it unrealistic on a wider selection of devices.

The high computational efficiency of SoCs presents a unique opportunity for cluster computing. SoC clusters will suffer from individual compute limitations, but can show overall performance gains as seen in [7]–[9]. Connectivity is an issue for SoCs, as it is often limited to an Ethernet connection. The authors in [7] saw networking issues for larger applications, but still achieved 75 MFLOPS for their cluster of dual-core ARM Cortex-A7 CPUs. The work presented in [8] examines an SoC cluster with Zynq CPU-FPGA SoCs. These devices show excellent performance due the flexibility of the FPGA fabric. Furthermore, this paper dives into the memory and communication overhead of their cluster, noting that MPI communication and DRAM random access showed comparable performance of approximately 30 MB/s. This performance suggests good potential for SoC clustering and the authors note there is potential for improvement with new SoCs. Lastly, the work presented in [9] showcases the Tegra K1 for two specific applications. From their conclusions, the observed GPU performance lags that of its HPC counterparts in terms of energy efficiency. This result is complicated by the impressive efficiency seen in the Tegra K1's ARM cores, indicating the potential utility for a select set of applications that can use both sets of cores.

## III. Approach

Metrics or benchmarking alone present an incomplete approach to this problem. Metrics provide a comparable maximum, but lack the application-specific details to understand real-world performance. Conversely, kernel benchmarking is a direct observation of real-world conditions, but is difficult to scale as an architecture changes. However, both are important tools in architecture analysis. Metrics provide the first-order analysis, determining if it is even theoretically possible to achieve comparable performance. The next step is to benchmark the individual devices with some common HPC kernels. Ranging from computationally intensive matrix multiplication to memory-intensive matrix transposition, benchmarking stresses the devices with real-world use. More computationally balanced kernels, including 1D and 2D Fast Fourier Transforms, provide insight to how observed performance changes with the workload. The benchmarking results presented in this paper represent the average of 1000 trials, reducing the effect of interrupt handling or timing inaccuracies.

Relating metrics to real-world benchmarking can be a complex challenge. There is some existing work with the CD metric. Both [4] and [5] leverage Eqs. 3 and 4 for calculating CD-RU. One common theme of both papers is that computational operations only considers productive work done on the data. Thus, there is some overhead in loop counters and memory management not included. Furthermore, in keeping with the CD metric, results consider operations performed by the device, not mathematical operations. This focus on device operations creates a level playing field when comparing devices.

$$Benchmark_{comp} = \frac{Computational\ Ops}{Execution\ Time} \tag{3}$$

$$RU_{CD} = 100\% \times \frac{Benchmark_{comp}}{CD} \tag{4}$$

While peak memory performance is important, real applications have memory overhead as well. In order to best evaluate the potential of an SoC cluster, we must extend the RU work in [4] to include EMB. Memory-usage patterns for an application can introduce overhead. For instance, some applications work well with line-based computation; while others, such as matrix multiplication, can take advantage of a block-based design. Given these considerations, the proposed method for EMB-RU leverages Eqs. 5 and 6. The variable $i$ in Eq. 5 accounts for each instance of data movement in an application. For simplicity, EMB-RU does not consider individual variables as they typically reside on-chip in registers or cache.

$$Benchmark_{mem} = \frac{\sum_i Memory\ Ops_i \times Data\ Size_i}{Execution\ Time} \tag{5}$$

$$RU_{EMB} = 100\% \times \frac{Benchmark_{mem}}{EMB} \tag{6}$$

The goal of RU is to aid in device comparisons by linking device metrics and real-world results. By extending this approach to consider memory systems, developers gain new insight into potential optimizations and can better identify device limitations. RU, however, is not a fixed result for a given device or even a given kernel. RU varies on kernel, implementation, dataset size, and Computational Intensity (CI). In order to remove the implementation variation on each device and focus on the kernel and problem size, each experiment leverages NVIDIA-optimized libraries: CUBLAS and CUFFT. CI is a metric to relate computation to data movement in an application, defined by Eq. 7 for the scope of our work.

$$CI = \frac{CD\ Ops}{CD\ Ops + EMB\ Ops} \tag{7}$$

For some kernels, CI can also vary with dataset size. This variance is due to leveraging device caching and data reuse. It is an important consideration for GPU and FPGA devices, where extensive parallelism can lead to data starvation. Most GPU developers design around on-device memory. A common example of this method is a block-based matrix multiplication

978-1-5090-3525-0/16/$31.00 ©2016 IEEE

TABLE I. METRICS OF DEVICES STUDIED [1], [10]–[12]

| Device | CD Float32 *(GOPS)* | CD Float64 *(GOPS)* | EMB *(GB/s)* | Power *(Watts)* |
|---|---|---|---|---|
| NVIDIA Tegra K1 (GPU only, Kepler) | 182.40 | 7.60 | 14.93 | 8 |
| NVIDIA Tegra X1 (GPU only, Maxwell) | 256.00 | 8.00 | 25.60 | 10 |
| NVIDIA K20X (Kepler) | 1967.60 | 655.87 | 250.00 | 235 |
| NVIDIA K40 (Kepler) | 2145.60 | 715.20 | 288.40 | 235 |
| NVIDIA Titan X SC (Maxwell) | 3462.14 | 108.19 | 336.48 | 250 |

implementation. The block size is typically determined by number of threads and on-chip memory size. This approach enables fewer external memory transfers while computing each block, resulting in a high CI and better performance.

In summary, RU analysis requires some knowledge about the device and the benchmark. As shown in subsequent sections, this approach begins with computing the device metrics to quickly gain insight into the devices under study. To complete RU analysis of a device, the only additional information required is execution time for each benchmark under study on the device at a certain dataset size. The following sections showcase how this approach can be used to evaluate multiple GPUs and identify efficiency trends in determining the potential of an SoC cluster. Considering common HPC kernels that range from compute to memory bound, this approach investigates if the sum of the individual Tegra SoCs can achieve comparable performance to their HPC counterparts, when also considering application efficiency. These HPC kernels include matrix multiplication, matrix transposition and both 1D and 2D Fast Fourier Transforms (FFT). Since computational kernel and dataset size are important factors, CI will also be explored.

## IV. DEVICE METRICS AND COMPUTATIONAL INTENSITY ANALYSIS

Device metrics and CI establish the scope for the RU analysis of GPUs. By examining the maximum theoretical computational power and memory bandwidth for a device. This first-order analysis, combined with TDP, reveals extremely high performance per Watt for both SoCs. CI analysis for the highly-optimized implementations of the kernels studied, reveals clear resource trends. Both metrics and CI provide the basis for further RU analysis and confirm the potential for scalable SoC cluster computing.

### A. Device Metrics

In order to determine if a cluster of SoCs can achieve similar performance in the same power envelope, we first must have the CD and EMB metrics for each device. Table I presents the metrics results and TDP for each GPU under study according to their published datasheets. CD is computed according to Eq. 1 for Float32 and Float64 precisions, and EMB is calculated according to Eq. 2. For a simpler and direct comparison, the ARM cores are ignored for the NVIDIA Tegra K1 and NVIDIA Tegra X1.

The metrics show some initially interesting results for the Kepler family of devices. Despite the 15 SMX units in the K40, we only see a 11.76× performance gain over the Tegra K1. The disproportionate difference is due to variations in the clock frequency between the two devices. We also see nearly a 29.38× power difference between the Tegra K1 and K40. This lower-power consumption leads to a significant difference in performance per Watt, but it is not proportional to the performance gain. Furthermore, we see a 20× difference in memory bandwidth between the Tegra K1 and K40 devices. These initial metrics would lend credence to the idea of combining multiple low-power chips to achieve greater performance on the order of the supercomputer-grade K20X or K40 due to the significant power differential. Also note, we are not considering any contribution from the ARM cores (although their power consumption is considered in the 5 Watt TDP).

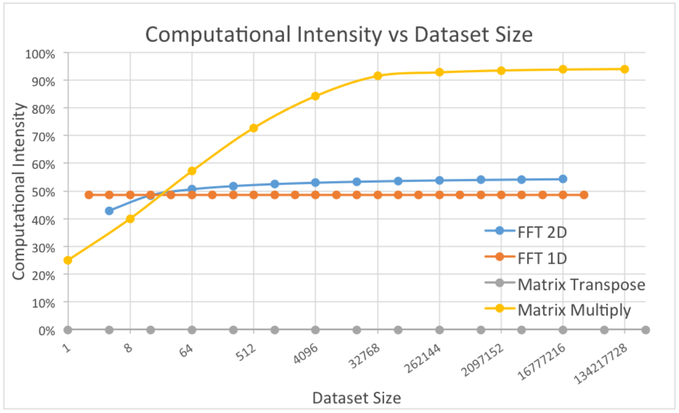

Fig. 1. Comparison of CI for multiple kernels under study.

The Maxwell architecture also shows interesting results. The Titan X SC is 25× the power of the Tegra X1; however, it is only 13.52× the GPU performance of the Tegra X1. This suggests that a cluster of Tegra X1s could provide enough performance to compete with the Titan X SC, with less power. Furthermore, the EMB metrics mirror the performance metrics: the Titan X SC is 13.14× the bandwidth of the Tegra X1. The metrics also show that the Tegra K1 and Tegra X1 have similar Float64 performance. The low ratio of Float64 to Float32 cores is unusually low for the Kepler architecture, but it is a documented design choice in the Maxwell architecture. The similar Float64 CD scores for the SoCs suggest that both will be compute bound and have similar benchmarking results for double precision datasets.

### B. Computational Intensity

Understanding the computational load of the kernels under study is an important factor in determining the impact of the RU results. Some kernels are obvious, such as matrix transposition. In this case the only work done on the data is reordering its layout in memory, thus it has a CI of 0%. Fig. 1 shows the CI of our kernels under study at various dataset sizes according to Eq. 7. Matrix multiplication is initially dominated by memory operations. Taking advantage of a block-based implementation, it quickly becomes almost exclusively computational.

One of the more interesting kernels is that of the FFT. The vector 1D FFT shows constant intensity around 49% computational due to the data-usage patterns between transforms. The

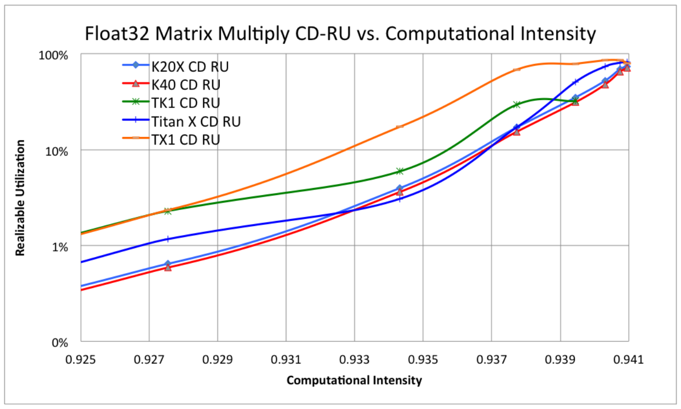

Fig. 2. Comparison of CD-RU of GPUs for Float32 Matrix Multiplication showing peak efficiency as CI increases.

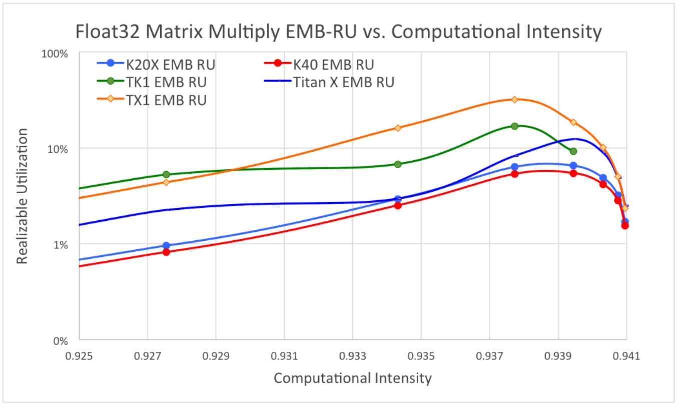

Fig. 3. Comparison of EMB-RU of GPUs for Float32 Matrix Multiplication showing an increase, then decrease as CI increases.

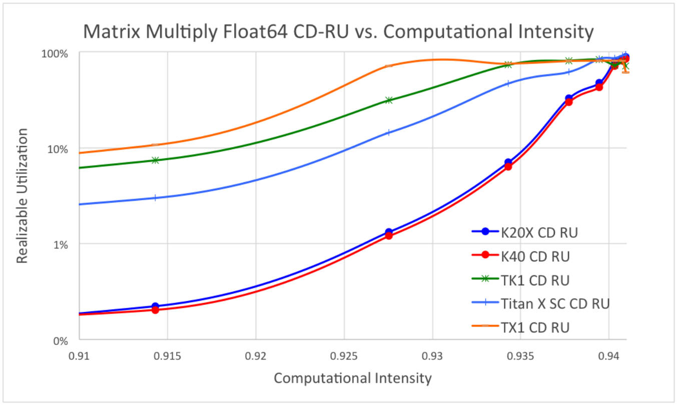

Fig. 4. Comparison of CD-RU of GPUs for Float64 Matrix Multiplication showing saturation for embedded GPUs at high CI.

2D FFT shows a similar trend to the matrix multiplication, albeit with much less variance. The range of 2D FFT is between roughly 45% and 55% as seen in Fig. 1.

## V. Realizable Utilization Analysis

With metrics and CI well defined for the scope of our work, RU analysis provides new insight into device characteristics. Variations in implementation details can often rob even the most powerful devices of computational performance. RU analysis relates the impact of the optimizations to the device architecture. This measure also enables new comparisons beyond the traditional bounding limitations. By leveraging RU analysis on a spectrum of benchmarks, performance trends and tradeoffs can quickly be identified. In the end, this combination of performance trends and potential for optimization can aid in determining how to best leverage a device. In some cases these observations might result in batch operations or multiple kernels executing at once. In the device selection phase of a project, understanding device utilization can be invaluable and often steer the decision-making process.

### A. Matrix Multiplication

Matrix Multiplication (MM) quickly becomes a computational bound kernel, pushing each device to their maximum performance. While small dataset sizes are dominated by memory transfers, the reuse of data in the algorithm enables a on-device caching and fewer external memory operations. These two distinct trends can be seen in Figs. 2 and 3. As CD-RU quickly rises for large sizes and high CI in Fig. 2, Fig. 3 shows a dramatic decline in EMB-RU due to data reuse and the algorithm taking advantage of the GPU's shared memories.

Close analysis of the Tegra K1 and Tegra X1 CD-RU results in Fig. 2 show that both devices do begin to saturate. The Tegra X1 constantly has the highest RU score, indicating that the 256 Float32 cores are well utilized. The Tegra K1 has a lower CD-RU than the Tegra X1, due to the slower DDR3 memory. As a result, the Tegra K1 does not demonstrate the same computational efficiency of the K40, consistent with the observations in [9]. The Tegra K1 does see CD-RU increase for larger CI, where memory use is less of the overall instruction count. From this observation, while the Tegra K1 has impressive compute power, the only applications that capitalize on the available resources are kernels with low memory requirements.

From the results in Fig. 2, it is clear that the SoCs are tracking similar CD-RU MM scores to their HPC counterparts. While the Tegra devices show a higher CD-RU score, this efficiency is due to limited computational resources. Analysis of the observed efficiency suggests that scaling would be feasible for compute bound applications on the Tegra X1 and to a lesser extent, the Tegra K1. This result is further supported by the EMB-RU MM scores in Fig. 3. Each devices shows a similar tend of tapering off near larger dataset sizes and higher CI, suggesting that the performance limitations in the Tegra K1 are mostly computational. The characteristics for the Tegra SoCs in terms of both CD and EMB are similar to the Titan X SC, K20X, and K40, showing that all GPUs considered are compute bound for MM.

Neither the Tegra K1 nor the Tegra X1 are designed for Float64 acceleration. Fig. 4 confirms this tradeoff for all devices in the Maxwell architecture family and Tegra K1. These devices reach nearly 100% CD-RU well before their Kepler HPC counterparts, a clear sign of computational resource limitation. The extremely limited double-precision resources would suggest for applications that rely on the Float64 data type, neither SoC in a cluster is a good candidate to scale in a cluster. This trend holds true for all kernels studied due to

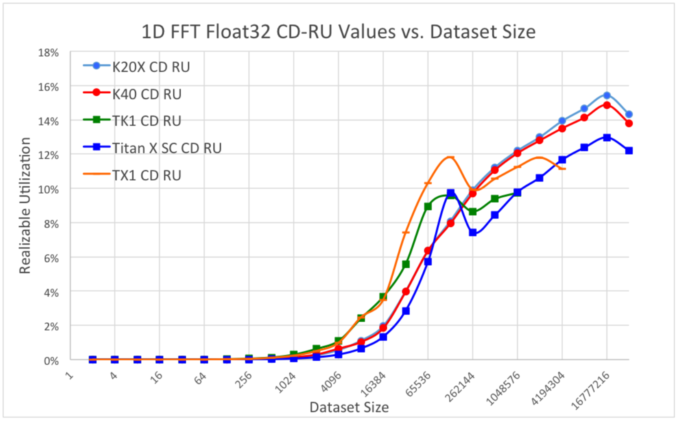

Fig. 5. Increasing CD-RU of GPUs for Float32 1D FFT against dataset size.

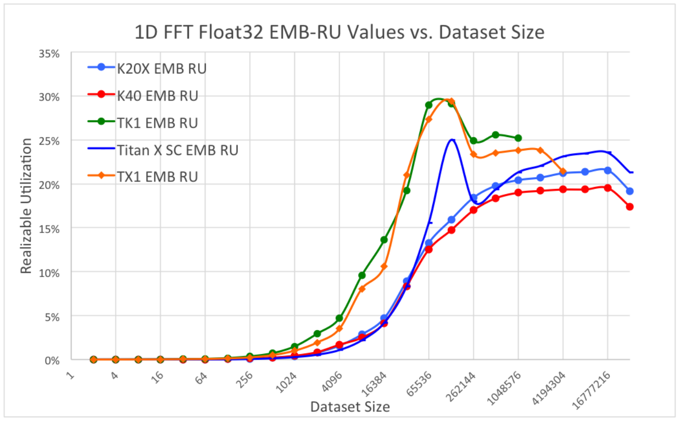

Fig. 6. Saturating EMB-RU of GPUs for Float32 1D FFT.

the extreme lack of Float64 resources.

## B. 1D Fast Fourier Transform

The 1D FFT has roughly equal memory and computational requirements. This trend is seen in its CI score in Fig. 1. Furthermore, since this kernel is a vector operation, its CI is very consistent among dataset sizes. The structure of the FFT forces an implicit compute barrier halfway through the computation. The barrier causes performance to be skewed as it is nearly impossible to hide memory accesses with computation. This algorithmic structure poses a large problem for a highly-parallel GPU, as is clearly visible in Fig. 5.

The results in Fig. 5 show that for small size FFTs, GPUs are inefficient with a CD-RU score very close to 0%. While peak CD-RU of any device may only make it to 15%, it does show a positive trend, which suggests that smaller dataset sizes are difficult to spread over the GPU cores. This limited utilization is also mirrored in the EMB-RU scores presented in Fig. 6. EMB-RU peaks out around 30% on the HPEC devices and between 20% and 25% for the HPC devices. The EMB-RU scores do suggest that both the Tegra K1 and Tegra X1 could be efficiently combined into a cluster. From the RU scores in Fig. 5, cluster of at least 16 Tegra X1s could match the performance of Titan X SC, showing good potential for HPEC cluster applications. This is a larger margin than seen with the MM kernel, suggesting that as compute requirements decrease and memory requirements remain low, the clustering potential increases.

## C. 2D Fast Fourier Transform

The 2D FFT shares similar CI with the 1D FFT, but also has similar variance to MM as seen in seen in its CI score in Fig. 1. While the range of this variance is greatly reduced, it becomes a factor due to the corner turn in the middle of the kernel. Like the 1D FFT, the CD-RU scores in Fig. 7 are very low, but show a positive trend. The memory hiding issues are reduced as the corner turn requires all values to move through the GPU a second time.

The impact of the corner turn can be seen in Fig. 8. The EMB-RU is much higher for each device, with all GPUs reach at least 50% for large dataset sizes and high computational intensity. As seen with both the 1D FFT and MM kernels,

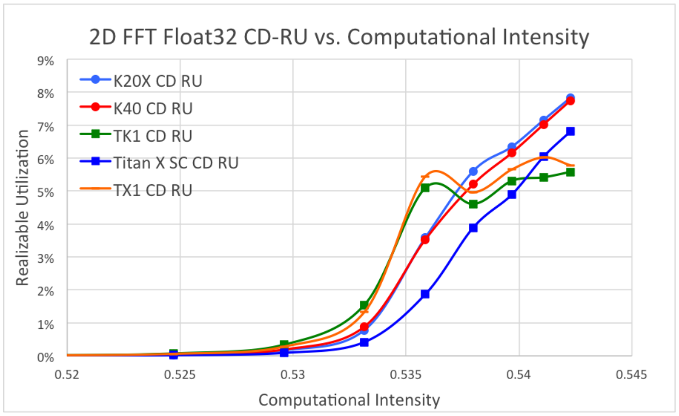

Fig. 7. Increasing CD-RU of GPUs for Float32 2D FFT against CI.

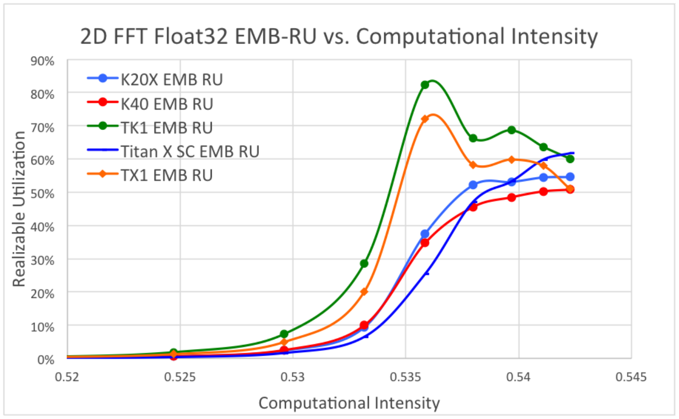

Fig. 8. Saturating EMB-RU of GPUs for Float32 2D FFT against CI.

the SoCs have higher utilization, due to the limited memory bandwidth.

Unlike MM and the 1D FFT, the CD-RU and EMB-RU results of the 2D FFT suggest that a cluster of Tegra K1s or Tegra X1s could scale very well, potentially exceeding the performance of their HPC counterparts. In order to match the performance of the K20X or K40, a cluster of 18 Tegra K1s would be required. This cluster would consume about 150 Watts, well below the TDP of either device. Likewise, at least 19 Tegra X1s would be required to match the performance

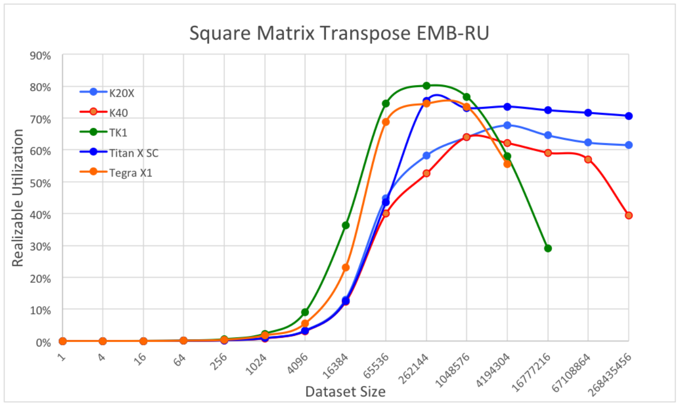

Fig. 9. Comparison of RU vs. dataset size for CUBLAS Matrix Transposition on Tegra K1, K20X, and K40 GPUs.

of the Titan X SC based on RU scores, resulting in potential power savings of almost 60 Watts. This low utilization is due to the structure of the FFT kernel, providing the GPU many idle cores to execute other applications and overhead.

## *D. Matrix Transposition*

Matrix transposition (MT) is at the opposite end of the CI spectrum from MM. The CD-RU for a transposition is 0%, which is in line with the 0% CI score. Fig. 9 shows the results of square-matrix transposition on Tegra K1, K20X, and K40 GPUs. The implementation leverages the CUBLAS library for maximum performance and productivity. Due to the smaller memory size and contiguous blocks of memory on the shared memory space of the Tegra K1 and Tegra X1, the maximum-sized matrix is only about 16 MB.

Tegra K1, Tegra X1, and Titan X SC show impressive EMB-RU. Peaking out at approximately 80% RU, the Tegra K1 bests the RU scores of both the Kepler and Maxwell architectures shown in Fig. 9. There are some significant differences in memories between GDDR5, DDR4, and DDR3 on the HPC GPUs, Tegra X1, and Tegra K1, respectively. Tegra K1 and Tegra X1 are limited to a 64-bit memory bus, compared with 384 on the GDDR5 memories. SoC memory is a shared space between the Linux kernel running on the ARM cores and GPU computation. This bottleneck was suspect for the limited MM performance of Tegra K1. However, the Tegra K1's high EMB-RU score in Fig. 9 suggests that limitations are not due to the Linux OS, but rather the speed of the DDR3 memories. The faster DDR4 memory of the Tegra X1 is reflected in the slightly lower EMB-RU score, of 70% of its 25.6 GB/s EMB. The high EMB-RU of the Tegra X1 may also be a factor of the Maxwell architecture as the Titan X has higher utilization than the K20X and K40 GPUs. With regards to potential scaling, due to the nature of the kernel and the limited EMB of the Tegra SoCs, neither is a good candidate. Since the GPU and CPUs in the Tegra SoCs share the memory controllers, MT would likely starve any other tasks or significantly degrade performance.

## VI. Conclusions and Future Work

The recent advances in SoC technology have resulted in large performance gains for HEPC. While many common SoC boards such as the Raspberry Pi, ODROID, Beagleboard, have been assembled into clusters, their performance per Watt has lagged behind that of most modern accelerators. However, the introduction of the NVIDIA Tegra K1 in 2014 and the NVIDIA Tegra X1 in late 2015 has enabled a well-supported ARM and CUDA ecosystem that is familiar to HPC and HPEC developers alike, while delivering unmatched performance. It is this theoretical and observed performance that enables a Tegra SoC cluster to be a plausible HPEC solution.

In examining the metrics for each device, both Tegra SoCs show high CD and EMB power-efficiency. While benchmarking performance lags that of other NVIDIA GPUs, the RU trends are only slightly elevated. Factoring in the difference in power consumption, both SoCs show excellent compute efficiency, in some cases better than their HPC counterparts. In order to evaluate real-world efficiency, we explore CD-RU and EMB-RU on each SoC with a set of HPC-oriented benchmarks. Analyzing the RU results on both embedded and high-power devices, we are able to gain insight into the potential for scaling SoC performance.

Not all applications show scalable performance on either Tegra SoC. Kernels that are extremely memory bound, such as Matrix Transposition, will suffer due to the slower memories. More computationally balanced kernels, such as a 2D FFT, could see speed up with a cluster of either SoC and offer more fine-grain system power and performance control. With compute-intensive kernels, such as Matrix Multiplication, a cluster of Tegra X1 could be a plausible competitor to more HPC-focused NVIDIA devices. The Tegra K1 does not scale as well with 64 fewer cores, limiting its utility to more power-limited applications. For both devices, potential clustering only shows benefits for Float32 operations due to the limited Float64 resources. These results show a clear trend that, with each new generation, a cluster of NVIDIA Tegra SoCs is becoming a viable finer-grain replacement for HPC computing.

Further work would investigate the impact of implementations of each kernel to better understand potential SoC-specific optimizations. While this initial analysis only considers external memory and compute performance, latency from communication networks would likely impact results. By understanding the impact of various implementation choices, communication latency could be hidden or leveraged to increase communication throughput. Building upon implementation optimizations, this information could be combined to understand how Tegra SoCs could be used in tandem with other HPC accelerators to better address the computational needs of large systems.

## Acknowledgments

This work was funded by the industry and government members of the NSF CHREC Center, as well as the I/UCRC Program of the National Science Foundation under Grant No. IIP-1161022. The authors would like to thank Aishwarya Dhandapani, Ishan Dalal, Athira Ajaykumar, Yathindra Kota, and Peter Harduvel of the NSF Center for High-Performance Reconfigurable Computing at the University of Florida for assisting with the calculation and confirmation of the metrics presented.

## References

[1] NVIDIA. (2015) Nvidia tegra x1: Nvidia's new mobile superchip, version 1.0. Apress. Accessed: 2016-4-20. [Online]. Available: http://international.download.nvidia.com/pdf/tegra/Tegra-X1-whitepaper-v1.0.pdf

[2] J. Williams, C. Massie, A. George, J. Richardson, K. Gosrani, and H. Lam, "Characterization of fixed and reconfigurable multi-core devices for application acceleration," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 3, no. 4, pp. 19:1–19:29, Nov. 2010.

[3] J. Richardson, S. Fingulin, D. Raghunathan, C. Massie, A. George, and H. Lam, "Comparative analysis of hpc and accelerator devices: Computation, memory, i/o, and power," in *Proceedings of High-Performance Reconfigurable Computing Technology and Applications Workshop (HPRCTA)*, ser. SC '10, New Orleans, LA, nov 2010.

[4] J. Richardson, A. George, and H. Lam, "Performance analysis of gpu accelerators with realizable utilization of computational density," in *Proceedings of Symposium on Application Accelerators in High-Performance Computing (SAAHPC)*, Chicago, IL, July 2012.

[5] A. Milluzzi, J. Richardson, A. George, and H. Lam, "A multi-tiered optimization framework for heterogeneous computing," in *High Performance Extreme Computing Conference (HPEC), 2014 IEEE*, Sept 2014, pp. 1–6.

[6] E. Konstantinidis and Y. Cotronis, "A practical performance model for compute and memory bound gpu kernels," in *Parallel, Distributed and Network-Based Processing (PDP), 2015 23rd Euromicro International Conference on*, March 2015, pp. 651–658.

[7] Z. Krpic, G. Horvat, D. agar, and G. Martinovi, "Towards an energy efficient soc computing cluster," in *Information and Communication Technology, Electronics and Microelectronics (MIPRO), 2014 37th International Convention on*, May 2014, pp. 178–182.

[8] P. Moorthy and N. Kapre, "Zedwulf: Power-performance tradeoffs of a 32-node zynq soc cluster," in *Field-Programmable Custom Computing Machines (FCCM), 2015 IEEE 23rd Annual International Symposium on*, May 2015, pp. 68–75.

[9] J. Zhang, S. You, and L. Gruenwald, "Tiny gpu cluster for big spatial data: A preliminary performance evaluation," in *2015 IEEE 35th International Conference on Distributed Computing Systems Workshops*, June 2015, pp. 142–147.

[10] NVIDIA. (2014) Nvidia's next generation cuda compute architecture: Kepler gk110/210, version 1.1. Apress. Accessed: 2016-4-20. [Online]. Available: http://images.nvidia.com/content/pdf/tesla/NVIDIA-Kepler-GK110-GK210-Architecture-Whitepaper.pdf

[11] NVIDIA. (2014) Nvidia tegra k1: A new era in mobile computing, version 1.0. Apress. Accessed: 2016-4-20. [Online]. Available: http://international.download.nvidia.com/pdf/tegra/Tegra-X1-whitepaper-v1.0.pdf

[12] M. Harris. (2014) Maxwell: The most advanced cuda gpu ever made. Accessed: 2016-4-20. [Online]. Available: https://devblogs.nvidia.com/parallelforall/maxwell-most-advanced-cuda-gpu-ever-made/