# Spectral Method Characterization on FPGA and GPU Accelerators

Karl Pereira and Peter Athanas
Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg,USA
{karlvt08,athanas}@vt.edu

Heshan Lin and Wu Feng
Department of Computer Science
Virginia Polytechnic Institute and State University
Blacksburg,USA
{hlin2,feng}@cs.vt.edu

***Abstract*—As CPU clock frequencies plateau and the doubling of CPU cores per processor exacerbate the memory wall, hybrid core computing, utilizing CPUs augmented with FPGAs and/or GPUs holds the promise of addressing high-performance computing demands, particularly with respect to *performance*, *power* and *productivity*. This paper compares the sustained performance of a complex, single precision, floating-point, 1D, Fast Fourier Transform (FFT) implementation on state-of-the-art FPGA and GPU accelerators. As results show, FPGA floating-point performance is highly sensitive to a mix of dedicated FPGA resources; DSP48E slices, block RAMs and FPGA I/O banks in particular. Estimated results show that for the floating-point FFT benchmark on FPGAs, these resources are the performance limiting factor. For fixed-point FFTs, however, FPGAs exploit a flexible data path width to trade-off circuit cost with speed of computation in applications requiring smaller precision to improve performance, power and device utilization. GPUs cannot fully take advantage of this, having a fixed data-width architecture.**

***Keywords*-FFT, floating-point, integer-point, HPC, FPGA, GPU**

## I. Introduction

The increasing demand for *High Performance Computing* (HPC) across a myriad environments, ranging from traditional supercomputers to embedded devices has outpaced the conventional processor's ability to deliver performance. The technique of simply scaling a single-core processor's frequency according to Moore's law for increased performance has run its course due to power dissipation escalating to impractical levels. Both of these factors have given rise to the notion of co-processors that augment a CPUs capabilities by offloading data- and compute-intensive portions of an application to dedicated hardware ranging from GPUs [1], FPGAs [2] to special ASICs [3].

This paper focuses on FPGA and GPU accelerators, primarily because they have shown potential in meeting the demands of current and future HPC applications. GPUs, traditionally used for graphical operations, have become attractive to high-performance scientific applications due to a combination of high floating-point performance, low cost and ease of programming. FPGAs, on the other hand, provide massive parallelism and have the flexibility to be tuned to meet the specific needs of an application without the cost or delay of designing a custom co-processor. The Fast Fourier Transform (FFT) is an efficient algorithm to compute the discrete fourier transform and its inverse. This paper provides a multi-dimensional evaluation of the FFT on FPGA and GPU accelerators with respect to performance, power and productivity. Application performance on these heterogeneous architectures is highly dependent on a balance of memory volume and bandwidth, input/output bandwidth and the availability of specific compute resources.

Productivity is another major issue concerning HPC. The time to a solution can sometimes be as important as performance itself. Factors such as the level of programming abstraction and the availability of standardized interfaces directly translate to ease of programming, scalability, portability and re-usability. This paper gives some insights into FPGA and GPU productivity and efforts to close the productivity gap. The need for low power directly impacts both space availability and cost of maintenance. Despite the importance of price, the paper focuses on the technical aspects of this computational space; hence, economic factors are not considered in this work.

This paper is organized as follows: Section II provides relevant work and how this paper adds value. The architectural features of the accelerators used in the study, and some assumptions made are discussed in Section III. Section IV details the approach used for the analysis and the optimization on the accelerators. Section V gives initial results and discussions. Finally, Section VI summarizes the findings.

## II. Related Work

Many modern devices ranging from x86 processors to GPUs to FPGAs have been compared by their raw computation density, power consumption, I/O bandwidth and memory bandwidth [4]. While these peak numbers are good to understand qualitative trends, an actual running application gives better quantitative insight and permits better assessment of these advanced architectures. Govindaraju et al. have analyzed GPU memory system behavior by using an FFT as the algorithm for evaluation [5]. A GeForce 7900 GTX performed a 1 M sample FFT in 19 ms and provided a 2X improvement over an Intel processor. An equivalent Virtex-4 FPGA implementation with a Sundance floating-point FFT core, operating at 200 MHz, performed a 1 M sample FFT in 21 ms. The GPU was demonstrated in this case to be superior.

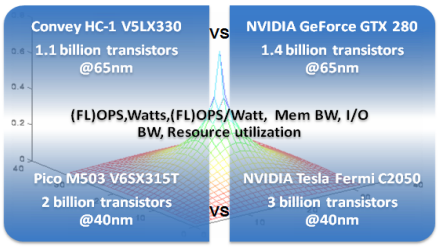



Figure 1: Mix of accelerators and corresponding metrics

Ben et al. compared GPUs and FPGAs using five different benchmark algorithms [6]. It was found that the FPGAs were superior over a GPUs for algorithms requiring a large number of regular memory accesses, while GPUs were better suited to algorithms with variable data reuse. It was concluded that FPGAs with a customized datapath outperform GPUs at the expense of a larger programming effort. Reduced bit-width floating-point units for FPGAs resulted in much higher speed-ups compared to a full floating-point implementation for a face detection algorithm studied by Yingsoon et al [7]. While still retaining 94% of face detection accuracy, a significant reduction in area utilization by 50% and power by 15%, was recorded.

Brahim et al. performed extensive comparison of the NVIDIA Tesla C1060 GPU and the Convey Hybrid Core (HC-1) FPGA system using four benchmarks having different computational densities and memory locality characteristics [8]. It was shown that the Convey HC-1 had superior performance and energy efficiency for the FFT and a Monte Carlo simulation. The GPU in [8] achieved a meagre 25 Giga Floating-point Operations Per Second (GFLOPS) for the 1D in-place FFT, possibly due to the unoptimized mapping of NVIDIA's CUDA FFT on the NVIDIA Tesla. The work done by Brahim et al. [8] concluded that for applications requiring random memory accesses, the Convey HC-1 with its optimized memory system for non-sequential memory accesses performs better than GPUs, which incur a higher performance penalty for non-contiguous memory block transfers.

In this paper, the authors demonstrate that GPUs fare much better for the floating-point 1D FFT compared to FPGAs due to immense floating-point capability and high memory bandwidth. A standalone FPGA has the advantage of a higher power efficiency and superior integer-point performance. The work reports higher floating-point performance on the Convey HC-1 over previous efforts. This paper supplements an existing base of knowledge by analyzing some of the reasons for the performance discrepancy between GPUs and FPGAs, that can help future design of FPGA accelerators for HPC.

## III. Architectural Considerations

As shown in Fig. 1, this study considers two generations of promising FPGA and GPU accelerators, based on technology size and a set of metrics. A Convey HC-1 node served as the Virtex-5 platform. A Pico EX-500 PCIe card (housing the M-503 module) was used as a representative Xilinx Virtex-6 platform. The architecture and programming model details of the Convey HC-1 and the Pico M-503 are available at [9], [10]. The Convey HC-1 is a high-performance, high-power, socket-based server system while the Pico M-503 is a PCIe-based system suited for low-power embedded applications. Similarly, a NVIDIA Tesla C2050 PCIe card based on the latest generation Fermi architecture with a large number of computational cores was used to supplement a previous generation NVIDIA GTX 280 PCIe card based on the GT200 architecture [11], [12].

Convey provides Math Libraries (CML), auto-loop unrolling techniques and a set of pre-defined personalities. A *personality* is a particular configuration (i.e. a bitstream) of the compute FPGAs, available as a custom instruction on the host-side application. For instance, Convey provides a single-precision vector personality with 32 function pipes. Each pipe provides four Fused-Multiply Add (FMA) operations. Hence, the performance peaks at 76.8 GFLOPS (32 pipes * 4 FMA/pipe * 2 FLOPS/FMA * 300 MHz). The Convey supplied math library for a 1D FFT also peaks around 65 GFLOPS [9]. A sustained performance of 76 GFLOPS was recorded for a floating-point matrix multiply [13]. These numbers are low considering the abundant compute resources available in these high-end architectures. None of the personalities, libraries or auto-code generation techniques were used for the implementation in this study. The design in this paper is custom developed using Convey's Personality Development Kit (PDK), Verilog and the FFT IP core from Xilinx [14]. The Pico M-503 system architecture with the FFT cores is shown in Fig 3. It has two 2GB DDR3 memory modules providing an aggregate memory bandwidth of roughly 17 GB/s.

Scientific computing typically requires both single precision and double precision floating-point operations. There is a clear advantage for GPUs, which already support IEEE-754 single precision and double precision built into the architecture for native floating-point graphic rendering operations. FPGAs, on the other hand, usually have no native support for floating-point arithmetic and many applications use fixed-point implementations for ease of development. FPGA developers use on-chip programmable resources to implement floating-point logic for higher precisions, but these implementations consume significant resources and tend to require deep pipelining to get acceptable performance [15]. Peak performance on CPUs and GPUs can be defined as the product of the number of floating-point operations per clock, number of floating-point cores and clock frequency. For FPGAs, calculating the peak performance is more complex considering the different combinations of functional units that can be generated and the level of maturity of the floating-point cores [16]. As seen in Table I, GPUs clearly benefit with roughly 3X-4X times higher floating-point performance compared to FPGAs. The peak predicted in the case of FPGAs is the estimate made in [16] after factoring in logic due to the I/O interface, place and route, and reduced clock frequency for timing considerations. The question remains as to how much of that peak can be sustained.

In the Convey HC-1, the host CPU communicates with the co-processor board via the Front Side Bus (FSB), that

| Device | Absolute Peak | Predicted Peak |
|---|---|---|
| Virtex-5 XC5VLX330 | 135 [16] | 110 [16] |
| NVIDIA Geforce GTX 280 | 933 [12] | N/A |
| Virtex-6 XC5VSX315T | 380 [16] | 355 [16] |
| NVIDIA TESLA FERMI C2050 | 1030 [11] | N/A |

Table I: Peak device performance in GFLOPS

becomes a bottleneck for streaming applications. Hence, for experiments in this study, data is appropriately allocated in the device memory and the communication overhead is ignored. The peak bandwidth of both PCIe (x16) and FSB are roughly 8.5 GB/s, so the assumption does not bias either platform. The performance of a streaming FFT illustrates how important inter-processor bandwidth is for accelerators. Consider a 1024-pt FFT, where the real and imaginary components are represented with 32-bit floating-point numbers. Assume the sustained FSB bandwidth is an optimistic 4.8 GB/s, and that the accelerator is infinitely fast, such that no time is spent performing the calculations. This operation then requires 8 KB (1024 X 8 Bytes) of data to be transferred 629 K times (4.8 GB/s / 8KB) per second over the I/O link. The resulting performance of the overall system is around 32 GFLOPS as illustrated in Equation 1.

$$(629\text{k} \times 1024 \times 5 \times \log_2 1024) \,/\, 1\,\text{s} = 32\ \text{GFLOPS} \quad (1)$$

As seen in Table I, FPGAs are capable of higher floating-point performance. However, the FSB in the Convey HC-1 avails in keeping application data coherent with the processor memory space compared to PCIe-based accelerators. Emerging heterogeneous computing architectures that *fuse* the CPU and GPU on the same die, for example, AMD's Fusion Accelerated Processing Unit (APU) and Intel's Knights Ferry, hold the promise of addressing the PCIe bottleneck in GPU accelerators. Clearly, higher I/O bandwidth is required if sustained performance is to be achieved for streaming applications in FPGA accelerators.

## IV. Approach

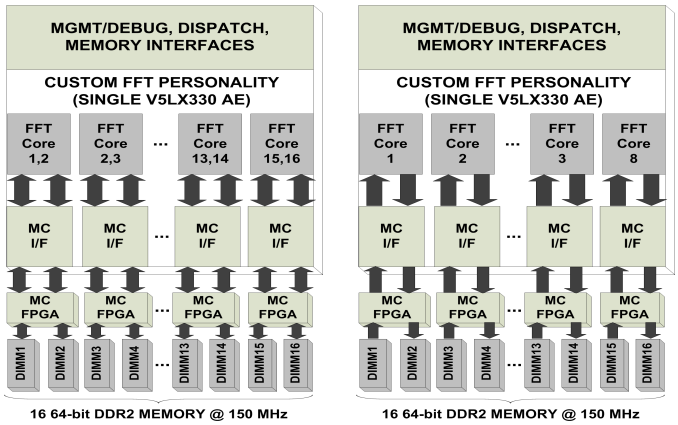
The FFT was selected to characterize the accelerators, being memory and computation intensive, and a recurrently seen kernel in myriad signal processing applications in both high performance servers and embedded computers. It is one of the thirteen computational idioms identified by Berkeley to benchmark modern parallel architectures [17]. The FFT was implemented using the Xilinx FFT IP core, that provides the ability to make all necessary algorithmic and implementation specific trade-offs to select the most resource-and performance-efficient solution for any transform size [14]. Both pipelined and burst implementations were considered to balance resource utilization, transform time and external memory bandwidth. The complex, floating-point data samples with real and imaginary parts, each of 32 bits are stored in external memory. The metric for performance used is FLOPS given by Equation 2.

$$FLOPS = \frac{N_c * 5\, N\, log_2(N)}{T_t} \quad (2)$$

where $N_c$ = Number of FFT cores, $N$ = FFT transform size,$T_t$ = Transform time. The "5" from the equation comes from the number of floating-point operations in a single butterfly stage of the FFT. There are a total of ten floating-point operations taking place. A $N$-point FFT using radix-2 decomposition has $\log_2(N)$ stages, with each stage containing $N/2$ radix-2 butterflies.

The FPGA implementation for the Convey HC-1 used the Verilog HDL, the Xilinx FFT IP and the Convey-supplied Personality Development Kit (PDK). The Pico implementation used the Verilog HDL, the Xilinx Memory Interface Generator [18], a Pico-designed PCIe endpoint module and driver API [10]. The GPU used the FFT that is a part of the Scalable Heterogeneous Computing (SHOC) benchmark suite written in OpenCL [19]. Both implementations use the well known Cooley-Tukey algorithm.

### A. Optimizing FFT on the Convey HC-1




(a) Un-optimized mapping (b) Optimized mapping

Figure 2: FFT mapping on the Convey HC-1

By utilizing all available DSP48E slices, sixteen radix-2 burst FFT cores were implemented on each compute FPGA. Fig. 2a shows the case for a single FPGA. The number of cores correspond to the sixteen 150 MHz memory ports available for reading and writing. The burst FFT cores use an iterative approach to save resources, thus, the core does not load the next data frame until it finishes processing the current data frame. An implementation with 16 pipelined, streaming cores would be limited by DSP48E slices. Fewer cores would reduce utilized memory ports, making the implementation unbalanced. To improve performance, eight streaming cores with lower DSP48E usage (and more Virtex-5 CLBs) were implemented. For a pair of memory ports, the first port was used for streaming data in and the second port was used to write-back results. As seen in Fig. 2b, dedicated read and write datapaths eliminate bus turn-around penalties and improve performance.

### B. Optimizing FFT on the Pico M-503

Custom DDR3 SDRAM memory controllers with a burst-length of eight were developed using the Xilinx memory

interface generator [18], running at 400 MHz, to ease timing closure. Based on the width of the user-interface provided by the memory controllers (i.e. 256 bits), four FFT cores were mapped, running at half the memory frequency. The SRAM banks were not useful since they provide only 18 bits at a time, while the FFT core requires 64 bits of real and imaginary data in a single clock cycle. As in the Convey HC-1, continuous data streaming was possible with the first memory controller performing reads from sequential locations and the second memory controller writing results as illustrated in Fig. 3. Since DDR3 memory peak specified frequency is 533 MHz, higher performance could only be achieved by increasing the frequency of the memory controllers and optimizing memory access patterns. Timing constraints were met with additional stages being added to the datapath. Finally, the memory controllers were fine-tuned for performance by increasing the number of bank machines, at the expense of slightly higher resource utilization.

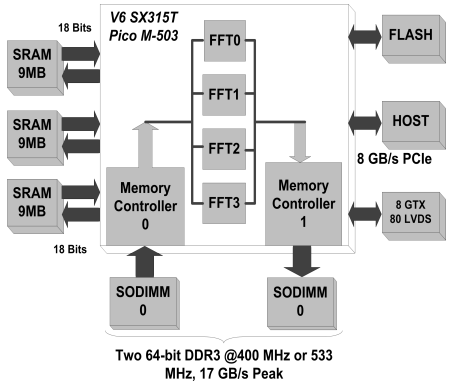



Figure 3: FFT mapping on the Pico M-503

### C. Power Measurement Setup

Power was measured by a Watts Up meter that connected the system or the device under test to the power supply. It is important to note the distinction between the system and device powers. System power (in Convey and NVIDIA GTX 280), is the power measured when both the host processor and the device share the same power supply. Device power (in Pico and NVIDIA Fermi), signifies the power consumed by the device alone, from a dedicated power supply. Idle power, in the case of the system power, is the difference in power when the card or co-processor is attached to the host and when it is disconnected. This is done because the GPU and the Convey HC-1 have different base CPU configurations and power supplies. For the device, the idle power is recorded when the GPU is not running any algorithm, and the Pico M-503 module does not have a bitstream loaded. The load in all the devices is a 1024-point FFT run long enough (around 10 seconds) to average out any fluctuations and start-up delays. Thus, load power indicates power level when system or device is running the FFT, in addition to the idle power.

## V. Results and Discussion

The performance across the platforms is shown in Fig. 4a, including integer-point numbers for the FPGA. GPUs inherently support floating-point computation and the width of the datapath is fixed, so a reduced precision is not efficient. For a long integer-point FFT (32-bits) implementation, the GPU performance would be similar to the floating-point performance and this analysis makes that assumption. For all practical purposes, the performance would degrade, due to the additional datapath scaling needed to accommodate bit-growth after each stage of the transform.

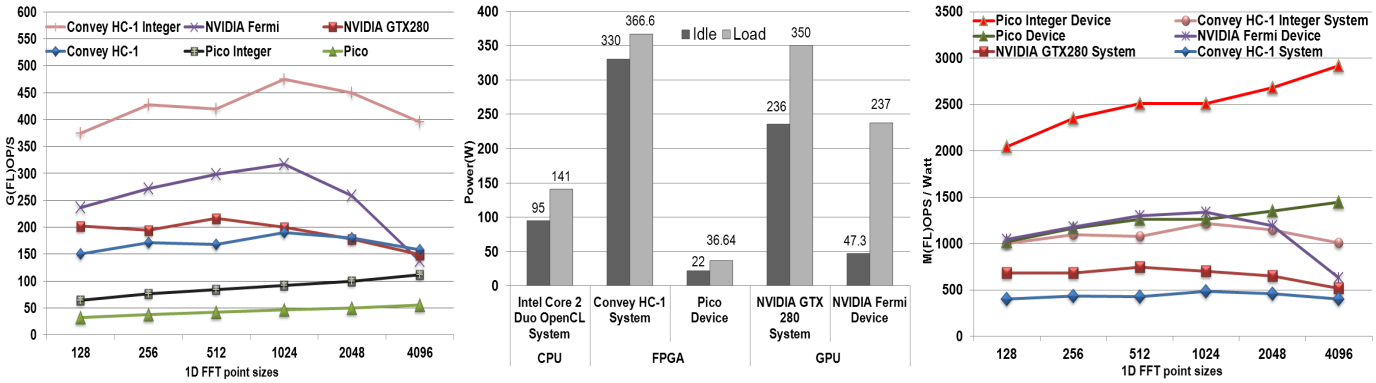
### A. Performance

For the Convey HC-1, performance of all four compute FPGAs is considered against a single GPU. This is done for three reasons:

1) All the FPGAs being confined to the same co-processor board, makes it hard to isolate a single FPGA, specially when measuring aggregate power of the co-processor,
2) Comparing external memory bandwidth in excess of 140 GB/s in GPUs, against a single FPGA in the Convey HC-1 having around 19.2 GB/s (8 B * 300 Mhz * 8 memory controllers) of memory bandwidth, makes the results biased for the GPU, and
3) Finally, utilizing all FPGAs can help to evaluate the performance degradation caused by multiple FPGAs accessing the same memory controller and link saturation.

The maximum performance attained using sixteen burst FFT cores was 75 GFLOPS, roughly the same as the Convey library. However, measuring memory bandwidth indicated that the memory controllers were not fully utilized. The Convey optimized implementation prior to 512-points used all memory ports and eight computational FFT cores per FPGA, implemented with all available DSP48E slices. Performance was limited by available memory ports. After 512-point, performance was bound by DSP48E slices. This reduced the number of FFT cores, and required a subset of the cores to use the Virtex-5 Look-Up Tables (LUTs) instead of the DSP48E slices to implement the complex multipliers. Performance peaks at 190 GFLOPS for a 1024-point transform. With six FFT cores synthesized using DSP48Es and one core compiled exclusively from the LUTs, one memory port was left idle. The performance curve after 1024-point keeps falling-off primarily because of DSP48E limitations and reduced memory bandwidth utilization. Performance beyond 4096-pt is bound by available block RAMs. Despite being designed for non-unity stride memory accesses, the Convey-designed scatter-gather memory modules gave approximately 6% superior performance [20]. The NVIDIA GTX 280 suffers after a 512-pt transform due to the capacity limits of the shared memory, resulting in a high latency to the global memory.

For the Pico M-503, performance is limited entirely by available memory ports. A linear increase in performance gives the Pico card an advantage for a large FFT transform, due to the ample DSP48E slices available in the Xilinx Virtex-6. The DDR3 memory clocked at 533 MHz improved performance linearly from 400 MHz. The maximum performance that can be achieved on the Virtex-6 SX315T was estimated to be around 98 GFLOPS, with a maximum of four DDR3 memory




(a) Performance (b) Power Consumption (c) Power Efficiency

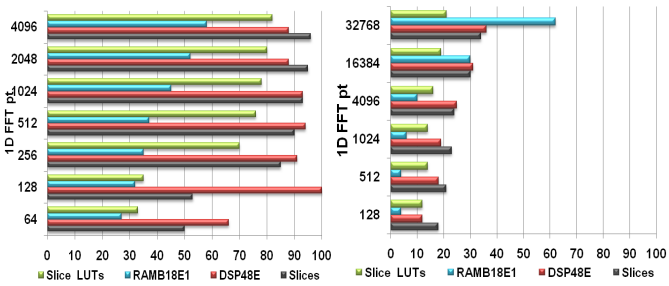
Figure 4: Performance, power consumption and power efficiency of architectures

controllers. The performance limitation is the Virtex-6 I/O banking architecture. The Virtex-6 requires a minimum of four I/O banks to implement a single memory controller. A different part of the Virtex-6 family, the XC6VLX760 with more I/O banks can support upto six memory controllers, resulting in a maximum performance of 192 GFLOPS, estimated from the current configuration. Another limitation is that only center I/O columns in the Virtex-6 can run at the fastest possible frequency (i.e. 533 MHz). In order to instantiate all six controllers, the outer columns (i.e. non-center columns) would have to be used, limiting the highest frequency to 400 MHz.

Unlike the Convey HC-1, FFT performance on the Pico M-503 module is not limited by local block RAMs, giving results upto a 32 K point transform. Fermi performs fairly well until 1024-pt, after which it falls sharply due to the thrashing of the L1 cache. In Fermi, the 64 KB L1 cache is configurable to support 48 KB of shared memory and 16 KB of L1 cache or 16 KB of shared memory and 48 KB of L1 cache. The benchmark used the former configuration. Reversing the configuration could improve performance, but is yet to be tested. The integer-point FFT implementation on the FPGAs enabled the number of FFT cores to be doubled, while permitting shorter pipelines. A 2.5-fold speed-up in the case of the Convey HC-1 resulted in a maximum of 470 Giga Operations Per Second (GOPS).

Despite having just two memory ports, the Pico M-503 edged the performance of a single Convey HC-1 FPGA after a 2048-pt transform, due to the abundant DSP48Es in the Virtex-6. This is also attributed to its low-latency, high-bandwidth DDR3 memory clocked at a higher rate. The NVIDIA GTX 280 has eight 64-bit memory channels to the GDDR3 memory having a clock rate of 1.1 GHz, almost 3.5X faster than Convey HC-1's memory frequency, and 2X faster than Pico's memory [12]. This clock rate and abundant floating-point multiply-accumulate units is what makes the FFT GPU performance impressive. The NVIDIA Fermi has almost double the number of compute cores compared to the GTX 280 and only six memory controllers. However, the reduced memory interface is compensated by the high-bandwidth GDDR5 memory, clocked at 1.5 GHz [11].

The Convey HC-1 and the Pico M-503 achieve 88% of the estimated performance without considering the effects of memory, demonstrating the efficient utilization of the memory subsystem by the FPGAs. As Fig. 5a illustrates, the Convey HC-1 utilizes nearly 90% of the slices and DSP48Es for large FFT transforms. The Convey-designed interfaces itself occupy 10% of the FPGA area and 25% of the block RAMs. The two DDR3 memory controllers, the PCIe interface, the FFT cores and custom interfacing logic occupy hardly 30% of the Pico M-503 module, as seen in Fig. 5b.




(a) A single Convey HC-1 FPGA resource utilization (b) Pico M-503 resource utilization

Figure 5: Device utilization on FPGA architectures

## B. Power

The higher core and memory frequencies increase the power consumption in GPUs, as seen in Fig. 4b. The large power consumption in the Convey HC-1 server system is attributed to the presence of nearly 16 FPGAs and supporting peripherals, resulting in the least power-efficient floating-point solution. The Pico M-503 draws the least power at idle and at load. As the graphs in Fig. 4c depict, this results in a higher power efficiency over the NVIDIA Fermi that performs almost 6X faster. Despite its low performance, Pico maintains a good power efficiency for large floating-point FFT transforms compared to GPUs, which suffer after a 2048-point FFT. The FPGA devices also exhibit a lower dynamic power consumption from idle to load. The integer-point FFT on the Pico M-503 is the most power-efficient solution, as apparent from its superior performance and lower power dissipation.

### C. Productivity

GPUs are easily programmable via high-level programming languages like CUDA and OpenCL. Programming FPGAs using HDLs is time consuming and requires low-level hardware design experience. The use of open-source and proprietary cores optimized for each generation of FPGA devices can reduce development time, unlike CUDA or OpenCL code which will run on a new architecture, but may not be optimal [21]. FPGA development involves a greater effort to design a custom datapath and optimize memory accesses. The Convey HC-1 helps speed-up development time by providing the necessary interfaces, optimized math libraries and abstracting away communication details from the developer. However, a major FPGA productivity bottleneck is the compilation time that can run into hours or even days for large implementations (i.e. many large-size FFT cores), aggravating debug time. There are on-going collaborative efforts to increase FPGA design productivity by using front-end high-level synthesis tools such as ImpulseC to develop Convey personalities.

## VI. Conclusions

The sustained FPGA performance is less than 50% of the estimated peak floating-point performance for the Virtex part. The FFT benchmark on FPGA accelerators reveals that higher sustained performance is limited by key device resources. For instance, the DSP48Es are pivotal to floating-point butterfly computations and attaining higher frequencies using fewer programmable device resources. In addition, block RAMs are needed to cache reused data (i.e phase factors) for large FFT transforms. Finally, I/O banks are required to achieve greater external memory bandwidth. GPUs attain the same percentage of the peak performance and need architecture-specific optimizations to get higher performance [21]. The Convey HC-1 performs worse than the NVIDIA GTX 280, despite having four FPGAs and the same number of memory ports. This is primarily due to the high latency to external memory, arising from a low clock frequency, that limits the operation of the FFT cores. As the Pico M-503 system demonstrates, this low frequency translates to a high power efficiency, that makes it suited for power-constrained environments.

FPGA floating-point FFT performance can be improved by having an appropriately balanced number of DSP48Es, BRAMs, I/O banks and integrating faster memory parts. Estimates show that the Convey HC-1$^{\text{ex}}$ [22], with a large Virtex-6 part can potentially deliver upto 350 GFLOPS for the 1D floating-point FFT, owing to the abundant DSP48Es. GPUs having the highest external memory bandwidth and abundant floating-point units excel at the floating-point 1D FFT, as seen in the Fermi. Integer-point FFT results for the FPGA show that reduced-precision can deliver almost 2-2.5X speed-up over the floating-point FFT, making it more power-efficient compared to GPUs.

## Acknowledgment

The authors would like to thank sponsoring members of the Center for High-Performance Reconfigurable Computing (CHREC), Glen Edwards of Convey Corporation, Pico Computing and Prasanna Sundararajan of Xilinx for supporting the work.

## References

[1] J. Owens, M. Houston, D. Luebke, S. Green, J. Stone, and J. Phillips, "GPU Computing," *Proceedings of the IEEE*, vol. 96, no. 5, pp. 879 –899, may 2008.

[2] P. Sundararajan, "High Performance Computing Using FPGAs," Xilinx, Tech. Rep., 2010.

[3] M. Gschwind, F. Gustavson, and J. F. Prins, "High Performance Computing with the Cell Broadband Engine," *Sci. Program.*, vol. 17, pp. 1–2, January 2009.

[4] 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 *High-Performance Reconfigurable Computing Technology and Applications (HPRCTA), 2010 Fourth International Workshop on*, nov. 2010, pp. 1 –10.

[5] N. K. Govindaraju, S. Larsen, J. Gray, and D. Manocha, "A Memory Model for Scientific Algorithms on Graphics Processors," in *Proceedings of the 2006 ACM/IEEE conference on Supercomputing*, ser. SC '06. New York, NY, USA: ACM, 2006.

[6] B. Cope, P. Cheung, W. Luk, and L. Howes, "Performance Comparison of Graphics Processors to Reconfigurable Logic: A Case Study," *Computers, IEEE Transactions on*, vol. 59, no. 4, pp. 433 –448, april 2010.

[7] Y. Lee, Y. Choi, S.-B. Ko, and M. H. Lee, "Performance Analysis of Bit-Width Reduced Floating-Point Arithmetic Units in FPGAs: A Case Study of Neural Network-Based Face Detector," *EURASIP Journal on Embedded Systems*, vol. 2009, pp. 4:1–4:11, January 2009. [Online]. Available: http://dx.doi.org/10.1155/2009/258921

[8] B. Betkaoui, D. Thomas, and W. Luk, "Comparing Performance and Energy Efficiency of FPGAs and GPUs for High Productivity Computing," in *2010 International Conference on Field-Programmable Technology (FPT)*, dec. 2010, pp. 94 –101.

[9] T. M. Brewer, "Instruction Set Innovations for the Convey HC-1 Computer," *IEEE Micro*, vol. 30, pp. 70–79, March 2010.

[10] *M-503 Hardware Technical Manual, Rev B*, Pico Computing Inc.

[11] *The Tesla C2050/C2070 GPU Computing Processor: Supercomputing at 1/10th the cost*, NVIDIA Corp. [Online]. Available: http://www.nvidia.com/docs/IO/43395/NV_DS_Tesla_C2050_C2070_jul10_lores.pdf

[12] *NVIDIA GeForce GTX 200 GPU datasheet*, NVIDIA Corp. [Online]. Available: http://www.nvidia.com/docs/IO/55506/GPU_Datasheet.pdf

[13] J. Bakos, "High-Performance Heterogeneous Computing with the Convey HC-1," *Computing in Science Engineering*, vol. 12, no. 6, pp. 80 –87, nov.-dec. 2010.

[14] Xilinx, *Xilinx LogicCore IP FFT*, 7th ed., March 2011. [Online]. Available: http://www.xilinx.com/support/documentation/ip_documentation/xfft_ds260.pdf

[15] R. Scrofano, G. Govindu, and V. K. Prasanna, "A Library of Parameterizable Floating-Point Cores for FPGAs and Their Application to Scientific Computing," in *Engineering of Reconfigurable Systems and Algorithms*, pp. 137–148.

[16] D. Strenski, P. Sundararajan, and R. Wittig, "The Expanding Floating-Point Performance Gap Between FPGAs and Microprocessors," HPCwire, November 2010.

[17] K. Asanovic, R. Bodik, J. Demmel, T. Keaveny, K. Keutzer, J. Kubiatowicz, N. Morgan, D. Patterson, K. Sen, J. Wawrzynek, D. Wessel, and K. Yelick, "A View of the Parallel Computing Landscape," *Commun. ACM*, vol. 52, pp. 56–67, October 2009.

[18] *Virtex-6 FPGA Memory Interface Solutions User Guide, UG406*, Xilinx, June 2011.

[19] A. Danalis and G. Marin, "The Scalable Heterogeneous Computing (SHOC) benchmark suite," in *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units*, ser. GPGPU '10. New York, NY, USA: ACM, 2010, pp. 63–74.

[20] K. Pereira, "Characterization of FPGA-based High Performance Computers," Master's thesis, Virginia Tech, 2011.

[21] M. Daga, "Architecture-Aware Mapping and Optimization on Heterogeneous Computing Systems," Master's thesis, Virginia Tech, 2011.

[22] *Convey's Hybrid-Core Technology: The HC-1 and the HC-1$^{ex}$*, Convey Corporation, November 2010.