Examining the Viability of FPGA Supercomputing

Stephen D. Craven and Peter Athanas
Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, VA 24061 USA
email: {scraven,athanas}@vt.edu

Abstract—For certain applications, custom computational hardware created using field programmable gate arrays (FPGAs) produces significant performance improvements over processors, leading some in academia and industry to call for the inclusion of FPGAs in supercomputing clusters. This paper presents a comparative analysis of FPGAs and traditional processors, focusing on floating-point performance and procurement costs, revealing economic hurdles in the adoption of FPGAs for general High-Performance Computing (HPC).

Index Terms— computational accelerator, digital arithmetic, Field programmable gate arrays, high-performance computing, supercomputers.

I. INTRODUCTION

Supercomputers have experienced a recent resurgence, fueled by government research dollars and the development of low-cost supercomputing clusters. Unlike the Massively Parallel Processor (MPP) designs found in Cray and CDC machines of the 70s and 80s, featuring proprietary processor architectures, many modern supercomputing clusters are constructed from commodity PC processors, significantly reducing procurement costs.

In an effort to improve performance, several companies offer machines that place one or more FPGAs in each node of the cluster. Configurable logic devices, of which FPGAs are one example, permit the device's hardware to be programmed multiple times after manufacture. A wide body of research over two decades has repeatedly demonstrated significant performance improvements for certain classes of applications when implemented within an FPGA's configurable logic [1].

Applications well suited to speed-up by FPGAs typically exhibit massive parallelism and small integer or fixed-point data types. When implemented inside an FPGA, the Smith-Waterman gene sequencing algorithm can be accelerated two orders of magnitude over software implementations [2, 3]. Although an FPGA's clock rate rarely exceeds one-tenth that of a PC, hardware implemented digital filters can process data at many times that of software implementations [4]. Additional performance gains have been described for cryptography [5], network packet filtering [6], target recognition [7] and pattern matching [8], among other applications.

Because of these successes, FPGAs have begun to appear in some supercomputing clusters. SRC Computers [9], DRC Computer Corp. [10], Cray [11], Starbridge Systems [12], and SGI [13] all offer

Dagstuhl Seminar Proceedings 06141
Dynamically Reconfigurable Architectures
http://drops.dagstuhl.de/opus/volltexte/2006/732

clusters featuring programmable logic. Cray's XD1 architecture, characteristic of many of these systems, integrates 12 AMD Opteron processors in a chassis with six large Virtex 4 FPGAs. The smallest configurable logic device found in these systems is a XC4VLX60, with many systems featuring some of the largest parts including the XC2VP100 and XC4VLX160. The architecture of the Cray and the SGI machines are similar in that they feature a high-performance channel connecting the FPGA fabric with the core processors.

The addition of these high-end FPGAs to commodity processor-based clusters can significantly increase costs. Ideally, this expense would be more than offset by the increased computational throughput provided by the FPGA. High-Performance Computing (HPC) applications, however, traditionally utilize double-precision floating-point arithmetic, a domain for which FPGAs traditionally are not well suited. Efforts have been made to automate the conversion of floating-point arithmetic to the fixed-point arithmetic more amenable to hardware implementations [14]. Even with an automated flow, however, user intervention is required to specify an error bound and cost function.

Modern FPGAs are heterogeneous in nature, mixing configurable logic cells with dedicated memories and fast, fixed logic for performing integer arithmetic such as multiplication and Multiply Accumulate (MAC). The fine-grained programmability of the configurable fabric permits bit-wise hardware customization, enhancing performance for applications involving small data widths such as gene sequencing. The dedicated fixed-width multipliers improve performance for standard Digital Signal Processing (DSP) applications while improving computational density.

Dedicated floating-point arithmetic logic, however, does not currently exist in FPGAs, requiring that floating-point units be constructed out of the dedicated integer multipliers and configurable fabric. The intent of this paper is to present a thorough comparison of FPGAs to commodity processors for traditional HPC tasks. Three separate analyses are performed. In Section II, a comparison of IEEE-standard floating-point performance of FPGAs to processors is presented. Section III considers performance enhancements gained by using custom floating-point formats in hardware. For certain applications the flexibility of custom hardware permits alternative algorithms to be employed, better leveraging the strengths of FPGAs through all-integer implementations. The performance for one representative application, Mersenne prime identification, is discussed in Section IV. Based on the results of these analyses, Section V discusses the conclusions drawn.

II. IEEE STANDARD FLOATING-POINT COMPARISON

Many common HPC tasks utilize double precision floating-point arithmetic for applications such as weather modeling, weapons simulation, and computational fluid dynamics. Traditional processors, where the architecture has fixed the data path's width at 32 or 64-bits, provide no incentive to explore reduced precision formats.

While FPGAs permit data path width customization, some in the HPC community are loath to utilize a

nonstandard format owing to verification and portability difficulties. By standardizing data formats across different platforms, applications can be verified against known, golden results by a simple comparison of the numerical results. Many applications also require the full dynamic range of the double precision format to ensure numeric stability. Furthermore, standard formats ease portability of software across different architectures. This principle is at the heart of the Top500 List of fastest supercomputers [15], where ranked machines must exactly reproduce valid results when running the LINPACK benchmarks.

A. Present Day Cost-Performance Comparison

Owing to the prevalence of IEEE standard floating-point in a wide range of applications, several researchers have designed IEEE 754 compliant floating-point accelerator cores constructed out of the Xilinx Virtex-II Pro FPGA's configurable logic and dedicated integer multipliers [16-18]. Dou et al published one of the highest performance benchmarks of 15.6 GFLOPS by placing 39 floating-point processing elements on a theoretical Xilinx XC2VP125 FPGA [19]. Interpolating their results for the largest production Xilinx Virtex-II Pro device, the XC2VP100, produces 12.4 GFLOPS, compared to the peak 6.4 GFLOPS achievable for a 3.2 GHz Intel Pentium processor. Assuming that the Pentium can sustain 50% of its peak, the FPGA outperforms the processor by a factor of four for matrix multiplication.

Dou et al's design is comprised of a linear array of MAC elements, linked to a host processor providing memory access. This architecture enables high computational density by simplifying routing and control, at the requirement of a host controller. Since the results of Dou et al are superior to other published results, and even Xilinx's floating-point cores, they are taken as a conservative upper limit on FPGA's double precision floating-point performance. Performance in any deployed system would be lower owing the addition of interface logic and the reduced frequency caused by routing densely packed designs.

Table I below extrapolates Dou et al's performance results for other FPGA device families. Given the similar configurable logic architectures between the different Xilinx families, it has been assumed that Dou et al's requirements of 1,419 logic slices and nine dedicated multipliers holds for all families. While the slice requirements may be less for the Virtex 4 family, owing to the inclusion of a MAC function with the dedicated multipliers, as all considered Virtex 4 implementations were multiplier limited the over estimate in required slices do not affect the results. The clock frequency has been scaled by a factor obtained by averaging the performance differential of Xilinx's double precision floating-point multiplier and adder cores [20] across the different families.

For comparison purposes several commercial processors have been included in the list. Sony's Cell processor powers the PlayStation 3 but was designed as a broadband media engine. The Cell features a PPC processing core along with eight independent processing cores sharing the same die. Intel's Pentium D 920 is a high-performance dual-core design. As the presented prices are merely for device cost, ignoring all required peripherals, data for the System X supercomputing cluster [21] has been included. The peak performance for each processor was reduced by 50%, taking into account compiler and system inefficiencies,

permitting a more fair comparison as FPGAs designs typically sustain a much higher percentage of their peak performance than processors. This 50% performance penalty is in line with the sustained performance seen in the Top500 List's LINPACK benchmark. In the table FPGAs are assumed to sustain their peak performance.

Table I. Double Precision Floating-Point Multiply Accumulate Cost-Performance

Device	Speed (MHz)	GFLOPS	Device Cost	$ / GFLOPS
Virtex 4 FPGA				
XC4VLX200-11	280	5.6	$7,010	$1,250
XC4VSX35-11	280	5.6	$542	$97
Virtex-II Pro FPGA				
XC2VP100-7	200	12.4	$9,610	$775
XC2VP100-6	180	11.2	$6,860	$613
XC2VP70-6	180	8.3	$2,780	$334
XC2VP30-6	180	3.2	$781	$244
Spartan 3 FPGA				
XC3S5000-5	140	3.1	$242	$78
XC3S4000-5	140	2.8	$164	$59
Processors		50% peak		
Pentium 630	3000	3	$167	$56
Pentium D 920	2800 x 2	5.6	$203	$36
Cell Processor	3200 x 9	10 [22]	$230 [23]	$23
HPC		Rmax	System Cost	
System X	2300 x 2200	12250	$5.8 M [24]	$473

As can be seen from the table, FPGA double precision floating-point performance is noticeably higher than for traditional Intel processors; however, when considering the cost of this performance processors fair better, with the worst processor beating the best FPGA. In particular, Sony's Cell processor is more than two times cheaper per GFLOPS than the best FPGA. The results indicate that the current generation of larger FPGAs, such the XC2VP100 and XC4VLX200 found on many FPGA-augmented HPC clusters, are far from cost competitive with the current generation of processors for double precision floating-point tasks typical of supercomputing applications.

With one exception, all costs in Table I only cover the price of the device and do not include other components (motherboard, memory, network, etc.) that are necessary to produce a functioning supercomputer. These additional costs are non-negligible and, while the FPGA accelerators would also incur additional costs for circuit board and components, it is likely that the cost of components to create a functioning HPC node from a processor, even factoring in economies of scale, would be larger than for creating an accelerator plug-in from an FPGA. To place these additional costs in perspective the cost-performance for Virginia Tech's System X supercomputing cluster has been included. Constructed from 1,100 dual core Apple XServe nodes, the supercomputer, including the cost of all components, cost $473 per GFLOPS. Several of the larger FPGAs cost more per GFLOPS even without the memory, boards, and

assembly required to create a functional accelerator.

As the dedicated integer multipliers included by Xilinx, the largest configurable logic manufacturer, are only 18-bits wide, several multipliers must be combined to produce the 52-bit multiplication needed for double-precision floating-point multiplication. For Xilinx's double precision floating-point core 16 of these 18-bit multipliers are required [20] for each multiplier, while for the Dou et al design only nine are needed. For many FPGA device families the high multiplier requirement limits the number of floating-point multipliers that may be placed on the device. For example, while 31 of Dou's MAC units may be placed on an XC2VP100, the largest Virtex-II Pro device, the lack of sufficient dedicated multipliers permits only 10 to be placed on the largest Xilinx FPGA, an XC4VLX200. If this device was solely used as a matrix multiplication accelerator, as in Dou's work, over 80% of the device would be unused. Of course this idle configurable logic could be used to implement additional multipliers, at a significant performance penalty. And while the device could implement many more floating-point adders, most HPC-related tasks, such as matrix multiplication, dot product, and Fast Fourier Transforms (FFTs), make heavy use of multiplication and would not benefit from a sea of adders.

B. Extrapolated Cost-Performance Comparison

While the larger FPGA devices that are prevalent in computational accelerators do not provide a cost benefit for the double precision floating-point calculations required by the HPC community, historical trends [25] suggest that FPGA performance is improving at a rate faster than that of processors. The question is then asked, when, if ever, will FPGAs overtake processors in cost-performance?

Several assumptions are made in the following analysis. As has been noted by some, the cost of the largest cutting-edge FPGA remains roughly constant over time, while performance and size improves. A first-order estimate of $8,000 has been made for the cost of the largest and newest FPGA – an estimate supported by the cost of the largest Virtex-II Pro and Virtex 4 devices. Furthermore, it is assumed that the cost of a processor remains constant at $500 over time as well. While these estimates are somewhat misleading, as these costs certainly do vary over time, the variability in the cost of computing devices between generations is much less than the increase in performance. The comparison further assumes, as before, that processors can sustain 50% of their peak floating-point performance while FPGAs sustain 100%. Whenever possible, estimates were rounded to favor FPGAs.

Two sources of data were used for performance extrapolation to increase the validity of the results. The work of Dou et al [19], representing the fastest double precision floating-point MAC design, was extrapolated to the largest parts in several Xilinx device families. Additional data was obtained by extrapolating the results of Underwood's historical analysis [25] to include the Virtex 4 family. The initial results are shown below in Figure 1(a) for the Underwood data and Figure 1(b) for Dou et al.

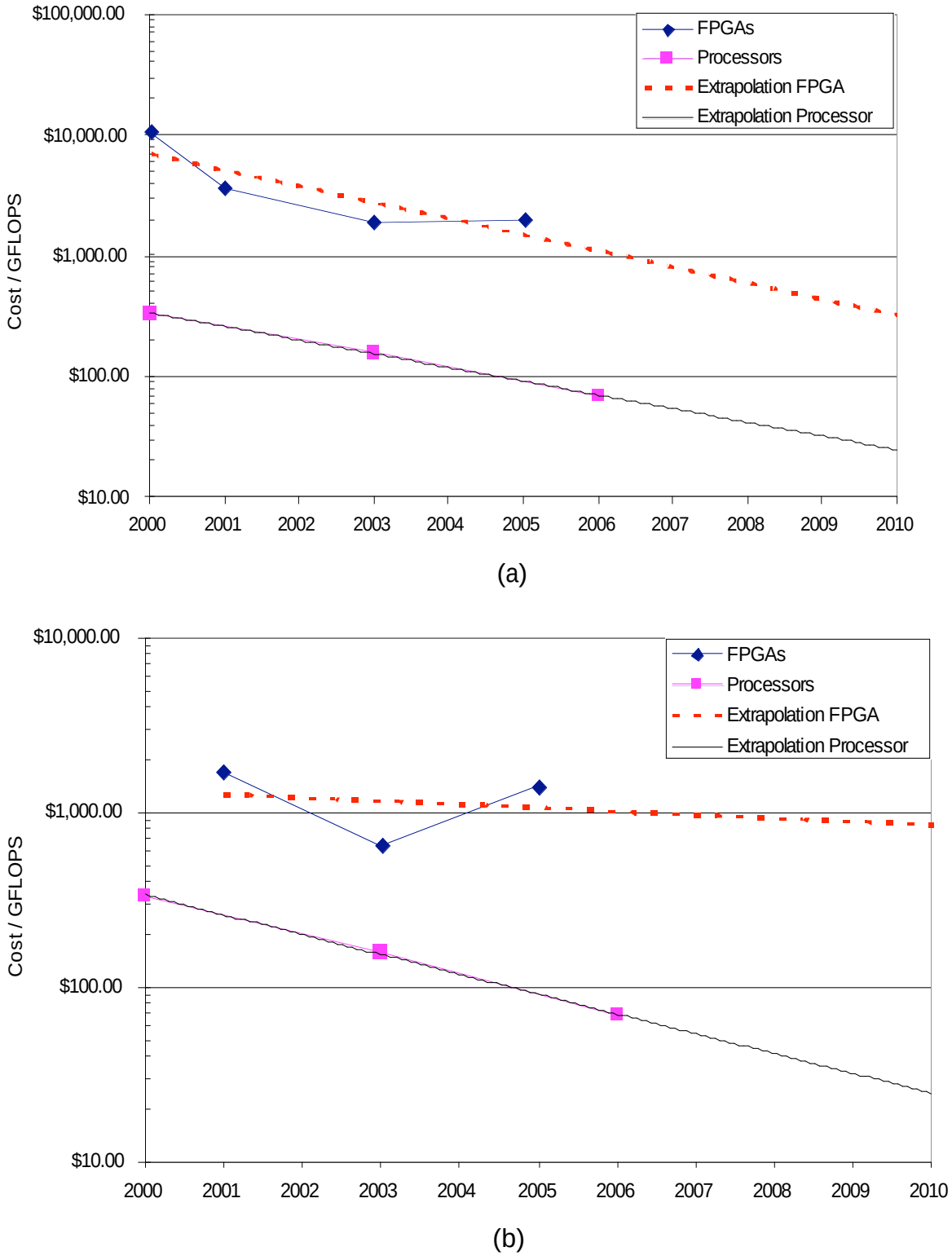


Figure 1: Extrapolated double precision floating-point MAC cost-performance for: (a) Underwood design and (b) Dou et al design.

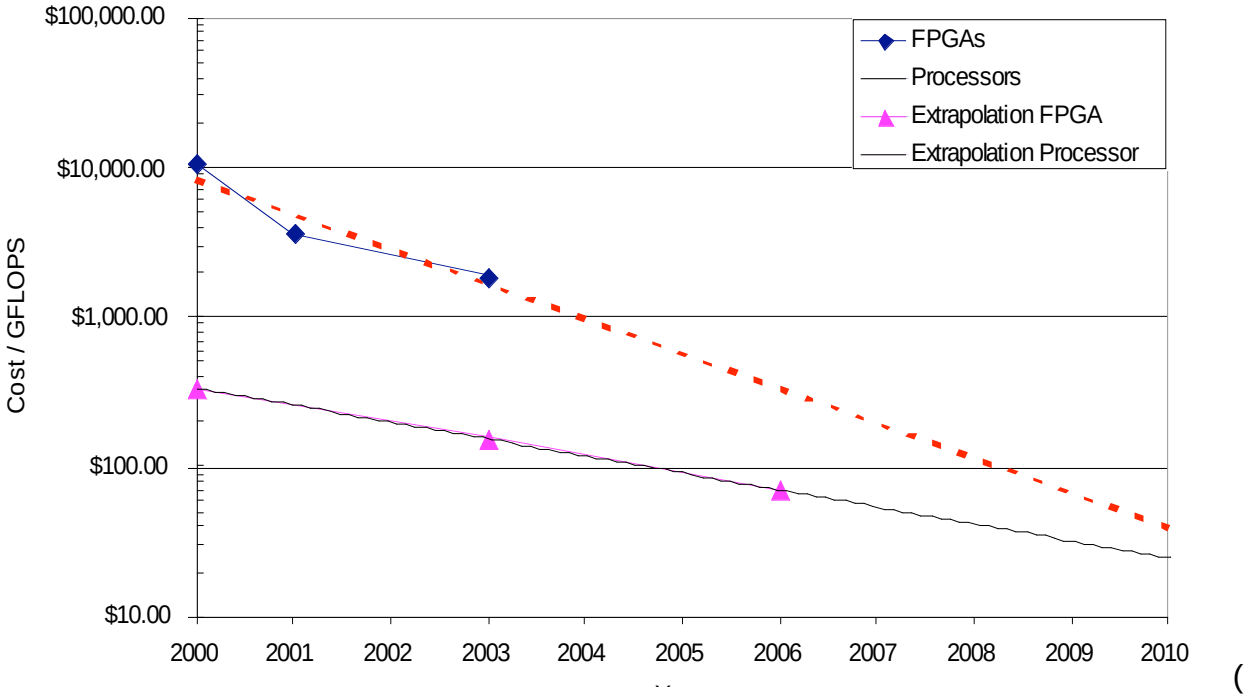
As apparent from the graphs, there are significant differences between the results of Dou and Underwood. An additional data point exists for the Underwood graph as his work included results for the Virtex E FPGAs. The design of Dou et al could not be extrapolated to this device as the Virtex E has no dedicated

multipliers and their design requires nine. The Dou et al design is higher performance and smaller, in terms of slices, than Underwood's design. In both graphs the latest data point, representing the largest Virtex 4 device, displays worse cost-performance than the previous generation of devices. This is due to the shortage of dedicated multipliers on the larger Virtex 4 devices. The Virtex 4 architecture is comprised of three sub-families: the LX, SX, and FX. The Virtex 4 sub-family with the largest devices, by far, is the LX and it is these devices that are found in FPGA-augmented HPC nodes. However, the LX sub-family is focused on logic density, trading most of the dedicated multipliers found in the smaller SX sub-family for configurable logic. This significantly reduces the floating-point multiplication performance of the larger Virtex 4 devices.

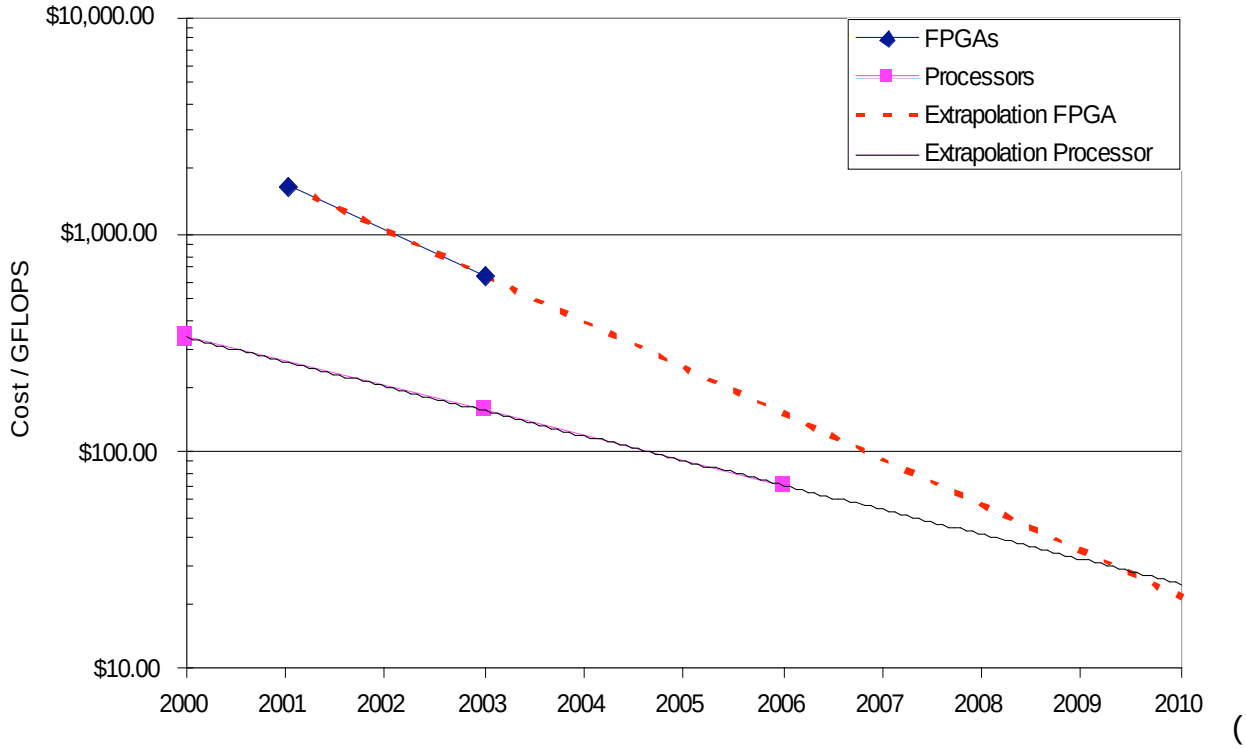
As the graphs demonstrate, if this trend towards logic-centric large FPGAs continues, it is unlikely that the largest FPGAs will be cost effective compared to processors anytime soon, if ever. The largest announced next-generation Virtex 5 device, the XC5VLX330, includes 192 dedicated multipliers, compared with only 96 that are present in the largest Virtex 4 device. Furthermore, the Virtex 5 multipliers are 25-bits by 18-bits wide, better supporting floating-point calculations. This multiplier design should reduce the number of dedicated multipliers required to construct a double precision floating-point multiplier from the present nine multipliers to four, eliminating the multiplier bottleneck that harms performance in the Virtex 4s.

As preliminary Virtex 5 data suggests that the relatively poor floating-point performance of the Virtex 4 is an aberration and not indicative of a trend in FPGA architectures, it seems reasonable to reconsider the results excluding the Virtex 4 data points. Figure 2, below, excludes these potential misleading data points and reconstructs the trend lines.

When the Virtex 4 data is ignored, the cost-performance of FPGAs for double precision floating-point matrix multiplication improves at a rate greater than that for processors. While there is always a danger from drawing conclusions from a small data set, both the Dou et al and Underwood design results point to a crossover point sometime around 2009 to 2012. At this point in time the largest FPGA devices, like those typically found in commercial FPGA-augmented HPC clusters, will be cost effective, compared to processors, for double precision floating-point calculations.



a)



b)

Figure 2: Extrapolated double precision floating-point MAC cost-performance excluding Virtex 4 data for: (a) Underwood design and (b) Dou et al design.

III. NONSTANDARD FLOATING-POINT COMPARISON

The use of IEEE standard floating-point data formats in hardware implementations prevents the user from leveraging an FPGA's fine-grained configurability, effectively reducing an FPGA to a collection of floating-point units with configurable interconnect. Seeing the advantages of customizing the data format to fit the problem, several authors have constructed nonstandard floating-point units.

One of the earlier projects demonstrated a 23x speedup on a 2-D FFT through the use of a custom 18-bit floating-point format [26]. More recent work has focused on parameterizible libraries of floating-point units that can be tailored to the task at hand [27-29]. By using a custom floating-point format sized to match the width's of the FPGA's internal integer multipliers, a speedup of 44 was achieved for a hydrodynamics simulation [30] using four large FPGAs.

Nakasato and Hamada's 38 GFLOPS of performance is impressive, even from a cost-performance standpoint. For the cost of their PROGRAPE-3 board, estimated at $15,000, it is likely that a 15-node processor cluster could be constructed producing 196 single precision peak GFLOPS. Even in the unlikely scenario that this cluster could sustain the same 10% of peak performance obtained by Nakasato and Hamada's for their software implementation, the PROGRAPE-3 design would still achieve a 2x speedup.

As in many FPGA to CPU comparisons, it is likely that the analysis unfairly favors the FPGA solution. Hardware implementations require specialized skills in digital design and vendor-specific tool flows. Development time and costs are significantly higher than for software. Many comparisons in literature spend significantly more time optimizing the hardware implementations than they do optimizing their software implementations. Previous research has demonstrated significant compiler inefficiency for common HPC functions [31]. For the DGEMM matrix multiplication function, a hand-coded version outperformed the compiler by greater than eight times. It appears that likely that Nakasato and Hamada's speedup would be significantly reduced, and perhaps eliminated on a cost-performance basis, if equal effort was applied to optimizing the software at the assembly level. To permit their design to be more cost-competitive, even against efficient software implementations, smaller FPGAs could be used. As evident from Table I, FPGA cost declines sharply as the size of the device decreases.

IV. ALL-INTEGER IMPLEMENTATIONS

The strength of configurable logic stems from the ability to customize a hardware solution to a specific problem at the bit level. The previously presented works all implemented coarse-grained floating-point units inside an FPGA for a wide range of HPC applications. For certain applications the full flexibility of configurable logic can be leveraged to create a custom solution to a specific problem, utilizing data types that play to the FPGA's strengths – integer arithmetic.

One such application can be found in the Great Internet Mersenne Prime Search (GIMPS) [32]. The software used by GIMPS relies heavily on double precision floating-point FFTs. Through a careful analysis of the problem, an all-integer solution is possible that improves FPGA performance by a factor of two.

The largest known prime numbers are Mersenne primes – prime numbers of the form 2^q-1, where q is also prime. The distributed computing project GIMPS has been created to identify large Mersenne primes and a reward of $100,000 has been issued for the first person to identify a prime number with greater than 10 million digits. The algorithm used by GIMPS, the Lucas-Lehmer test, is iterative, repeatedly performing modular squaring. As the numbers being squared are quite large (tens of millions of bits), specialized multiplication techniques are used.

One of the most efficient multiplication algorithms for large integers utilizes the FFT, treating the number being squared as a long sequence of smaller numbers. Instead of one, 24 million bit number, the algorithm may, for example, create a 3 million element sequence of 8-bit numbers. The linear convolution of this sequence with itself performs the squaring. As linear convolution in the time domain is equivalent to multiplication in the frequency domain, the FFT of the sequence is taken and the resulting frequency domain sequence is squared element-wise before being brought back into the time domain. Floating-point arithmetic is used to meet the strict precision requirements across the time and frequency domains. The software used by GIMPS has been optimized at the assembly level for maximum performance on Pentium processors, making this application an effective benchmark of relative processor floating-point performance.

Previous work focused on an FPGA hardware implementation of the GIMPS algorithm to compare FPGA and processor floating-point performance [33]. To leverage the advantages of a configurable architecture, multiple FFT implementations were considered, including floating-point and all-integer designs. The results of this comparison indicated that an all-integer FFT, specifically an irrational base discrete weighted transform, could outperform a floating-point implementation by a factor of two.

The final GIMPS accelerator, designed for the largest Virtex-II Pro FPGA, consisted of two 8-point FFT units fed by reorder caches constructed from the internal memories. To prevent a memory bottleneck, the design assumed four independent banks of DDR SDRAM. The final design could be clocked at 80 MHz and used 86% of the dedicated multipliers and 70% of the configurable logic.

In spite of the unique all-integer algorithmic approach, the customized FPGA implementation only achieved a speed-up of 1.76 compared to a 3.4 GHz Pentium 4 processor. Amdahl's Law limited the FPGA's performance due to the serial nature of certain steps in the algorithm. A slightly reworked implementation, designed as an FFT accelerator with all serial functions implemented on an attached processor, could achieve a speed-up of 2.6 compared to a processor alone under the reasonable assumption that communication could be overlapped with computation on the FPGA.

V. CONCLUSION

When comparing HPC architectures, many factors must be weighed, including memory and I/O bandwidth, communication latencies, and peak and sustained performance. However, as the recent focus on power consumption and commodity processor clusters demonstrates, cost-performance is of paramount importance. In order for FPGAs to gain acceptance within the general HPC community they

must be cost-competitive with traditional processors for the floating-point arithmetic typical in supercomputing applications. The analysis of the cost-performance of various current generation FPGAs revealed that only the lower end devices were cost-competitive with processors for double precision floating-point matrix multiplications. The limited capabilities of these lower end parts make them unattractive for inclusion in HPC nodes, with industry favoring the largest FPGA devices. An extrapolation of the double precision floating-point cost-performance of these larger devices using two different floating-point designs suggests that the largest FPGAs will not be cost-competitive with processors until the 2009-2012 timeframe. However, FPGA floating-point performance is very sensitive to the mix of dedicated arithmetic units in the architecture and for this cost-performance cross point to be achieved in the suggested timeframe requires architectures with significant dedicated multipliers. For non-IEEE standard floating-point formats, current generation FPGAs fair much better, making FPGAs cost-competitive with processors. While completely integer implementations of floating-point applications permit the FPGA to fully leverage its strengths, for at least one such application the cost-performance of an all-integer implementation was significantly worse than a processor. This suggests that, as with traditional integer applications, only certain classes of problems will experience significant performance improvements with hardware implementation.

REFERENCES

1. Compton, K. and S. Hauck, *Reconfigurable computing: a survey of systems and software.* ACM Comput. Surv., 2002. 34(2): p. 171-210.

2. Puttegowda, K., et al. *A Run-Time Reconfigurable System for Gene-Sequence Searching*. in *International VLSI Design Conference*. 2003. New Delhi, India.

3. TimeLogic, *DeCypher Engine G4*. 2006.

4. Tessier, R. and W. Burleson, *Reconfigurable Computing for Digital Signal Processing: A Survey.* Journal of VLSI Signal Processing, 2001. 28: p. 7-27.

5. Patterson, C. *High performance DES encryption in Virtex FPGAs using JBits*. 2000.

6. Sinnappan, R. and S. Hazelhurst, *A Reconfigurable Approach to Packet Filtering*, in *Lecture Notes in Computer Science*. 2001. p. 638.

7. Jean, J., et al. *Accelerating an IR automatic target recognition application with FPGAs*. 1999.

8. Zachary, K.B. and K.P. Viktor, *Time and area efficient pattern matching on FPGAs*, in *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*. 2004, ACM Press: Monterey, California, USA.

9. SCR Inc.., *SRC-7 Product Sheet*. 2006.

10. Vance, A. *Start-up could kick Opteron into overdrive*. The Register 2006 April 21, 2006

11. Woods, G. *Cray ARSC Presentation FPGA,*. in *ARSC High-Performance Reconfigurable Computing Workshop*. 2005. Fairbanks, AL.

12. Collins, J., G. Kent, and J. Yardley. *Using the Starbridge Systems FPGA-based Hypercomputer for Cancer Research*. in *International Conference on Military and Aerospace Programmable Logic Devices*. 2004. Washington, D.C.

13. SGI, *Extraordinary Acceleration of Workflows with Reconfigurable Application-specific Computing from SGI*. 2004.

14. Leong, M.P., et al. *Automatic floating to fixed point translation and its application to post-rendering 3D warping*. 1999.

15. Meuer , H., J. Dongarra, and E. Strohmaier. *Top 500 List*. 2005 [cited; Available from: http://www.top500.org.

16. Underwood, K.D. and K.S. Hemmert. *Closing the gap: CPU and FPGA trends in sustainable floating-point BLAS performance*. 2004.

17. Zhuo, L. and V.K. Prasanna. *Design tradeoffs for BLAS operations on reconfigurable hardware*. 2005.

18. Ho, C.H., et al. *Rapid prototyping of FPGA based floating point DSP systems*. in *13th IEEE International Workshop on Rapid System Prototyping, Proceedings*. 2002.

19. Dou, Y., et al., *64-bit floating-point FPGA matrix multiplication*, in *Proceedings of the 2005 ACM/SIGDA 13th international symposium on Field-programmable gate arrays*. 2005, ACM Press: Monterey, California, USA.

20. Xilinx, *Floating-point Operator v2.0 Datasheet.* 2006.

21. Ribbens, C., et al. *Balancing Computational Science and Computer Science Research on a Terascale Computing Facility*. in *ICCS 2005: 5th International Conference*. 2005. Atlanta, GA.

22. Chen, T., et al. *Cell Broadband Engine Architecture and its first implementation*. IBM developWorks 2005 Nov 29, 2005 [cited; Available from: http://www-128.ibm.com/developerworks/power/library/pa-cellperf/.

23. Lynch, M. *Playstation 3 slippage looking more likely - implications*. Technology Strategy report Feb 17, 2006.

24. Kahney, L., *System X Faster, but Falls Behind*, in *Wired News*. 2004.

25. Keith, U., *FPGAs vs. CPUs: trends in peak floating-point performance*, in *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*. 2004, ACM Press: Monterey, California, USA.

26. Shirazi, N., P. Athanas, and A. Abbott. *Implementation of a 2-D Fast Fourier Transform on a FPGA-Based Custom Computing Machine*. in *International Workshop on Field Programmable Logic and Applications*. 1995. Oxford, UK.

27. Liang, J., R. Tessier, and O. Mencer. *Floating point unit generation and evaluation for FPGAs*. 2003.

28. Belanovic, P. and M. Leeser. *A Library of Parameterized Floating Point Modules and Their Use*. in *International Conference on Field Programmable Logic and Applications*. 2002. Montpelier, France.

29. Dido, J., et al., *A flexible floating-point format for optimizing data-paths and operators in FPGA based DSPs*, in *Proceedings of the 2002 ACM/SIGDA tenth international symposium on Field-programmable gate arrays*. 2002, ACM Press: Monterey, California, USA.

30. Nakasato, N. and T. Hamada. *Astrophysical hydrodynamics simulations on a reconfigurable system*. 2005.

31. Gropp, W. *Closing the Performance Gap*. in *DOE SciDAC PI Meeting*. 2003. Napa, California.

32. GIMPS. *The Great Internet Mersenne Prime Search*. [cited; Available from: http://www.mersenne.org.

33. Craven, S., C. Patterson, and P. Athanas. *Super-sized Multiplies: How Do FPGAs Fare in Extended Digit Multipliers?* in *International Conference on Military and Aerospace Programmable Logic Devices*. 2004. Washington, D.C.