Hindawi Publishing Corporation
EURASIP Journal on Embedded Systems
Volume 2007, Article ID 93652, 8 pages
doi:10.1155/2007/93652

Research Article

Examining the Viability of FPGA Supercomputing

Stephen Craven and Peter Athanas

Bradley Department of Electrical and Computer Engineering, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061, USA

Received 16 May 2006; Revised 6 October 2006; Accepted 16 November 2006

Recommended by Marco Platzner

For certain applications, custom computational hardware created using field programmable gate arrays (FPGAs) can produce 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).

1. INTRODUCTION

Supercomputers have experienced a resurgence, fueled by government research dollars and the development of low-cost supercomputing clusters constructed from commodity PC processors. Recently, interest has arisen in augmenting these clusters with programmable logic devices, such as FPGAs. By tailoring an FPGA's hardware to the specific task at hand, a custom coprocessor can be created for each HPC application.

A wide body of research over two decades has repeatedly demonstrated significant performance improvements for certain classes of applications through hardware acceleration in an FPGA [1]. Applications well suited to acceleration by FPGAs typically exhibit massive parallelism and small integer or fixed-point data types. Significant performance gains have been described for gene sequencing [2, 3], digital filtering [4], cryptography [5], network packet filtering [6], target recognition [7], and pattern matching [8].

These successes have led SRC Computers [9], DRC Computer Corp. [10], Cray [11], Starbridge Systems [12], and SGI [13] to offer 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 Xilinx Virtex-4 FPGAs. Many systems feature some of the largest FPGAs in production.

Many HPC applications and benchmarks require double-precision floating-point arithmetic to support a large dynamic range and ensure numerical stability. Floating-point arithmetic is so prevalent that the benchmarking application ranking supercomputers, LINPACK, heavily utilizes double-precision floating-point math. Due to the prevalence of floating-point arithmetic in HPC applications, research in academia and industry has focused on floating-point hardware designs [14, 15], libraries [16, 17], and development tools [18] to effectively perform floating-point math on FPGAs. The strong suit of FPGAs, however, is low-precision fixed-point or integer arithmetic and no current device families contain dedicated floating-point operators though dedicated integer multipliers are prevalent. FPGA vendors tailor their products toward their dominant customers, driving development of architectures proficient at digital signal processing, network applications, and embedded computing. None of these domains demand floating-point performance.

Published reports comparing FPGA-augmented systems to software-only implementations generally focus solely on performance. As a key driver in the adoption of any new technology is cost, the exclusion of a cost-benefit analysis fails to capture the true viability of FPGA-based supercomputing. Of two previous works that do incorporate cost into the analysis, one [19] limits its scope to a single intelligent network interface design and, while the other [20] presents impressive cost-performance numbers, details and analysis are lacking. Furthermore, many comparisons in literature are ineffective, as they compare a highly optimized FPGA floating-point implementation to nonoptimized software. A much

Table 1: Published FPGA supercomputing application results.

Application	Platform	Format	Speedup
DGEMM [21]	SRC-6	DP	0.9x
Boltzmann [22]	XC2VP70	Float	1x
Dynamics [23]	SRC-6E	SP	2x
Dynamics [24]	SRC-6E	SP	3x
Dynamics [25]	SRC-6E	Float	3.8x
MATPHOT [26]	SRC	DP	8.5x
Filtering [27]	SRC-6E	Fixed	14x
Translation [28]	SRC-6	Integer	75x
Matching [29]	SRC-6/Cray XD1	Bit	256x/512x
Crypto [30]	SRC-6E	Bit	1700x

better benchmark would redesign the algorithm to play to the FPGA's strengths, comparing the design's performance to that of an optimized program.

The key contributions of this paper are the addition of an economic analysis to a discussion of FPGA supercomputing projects and the presentation of an effective benchmark for comparing FPGAs and processors on an equal footing. A survey of current research, along with a cost-performance analysis of FPGA floating-point implementations, is presented in Section 2. Section 3 describes alternatives to floating-point implementations in FPGAs, presenting a balanced benchmark for comparing FPGAs to processors. Finally, conclusions are presented in Section 4.

2. FPGA SUPERCOMPUTING TRENDS

This section presents an overview of the use of FPGAs in supercomputers, analyzing the reported performance enhancements from a cost perspective.

2.1. HPC implementations

The availability of high-performance clusters incorporating FPGAs has prompted efforts to explore acceleration of HPC applications. While not an exhaustive list, Table 1 provides a survey of recent representative applications. The SRC-6 and 6E combine two Xeon or Pentium processors with two large Virtex-II or Virtex-II Pro FPGAs. The Cray XD1 places a Virtex-4 FPGA on a special interconnect system for low-latency communication with the host Opteron processors.

In the table, the applications are listed by performance. The abbreviations SP and DP refer to single-precision and double-precision floating point, respectively. While the speedups provided in the table are not normalized to a common processor, a trend is clearly visible. The top six examples all incorporate floating-point arithmetic and fare worse than the applications that utilize small data widths.

With no cost information regarding the SRC-6 or Cray XD1 available to the authors a thorough cost-performance analysis is not possible. However, as the cost of the FPGA acceleration hardware in these machines alone likely is on the order of US$10 000 or more, it is likely that the floating-point examples may loose some of their appeal when compared to processors on a cost-effective basis. The observed speedups of 75–1700 for integer and bit-level operations, on the other hand, would likely be very beneficial from a cost perspective.

2.2. Theoretical floating-point performance

FPGA designs may suffer significant performance penalties due to memory and I/O bottlenecks. To understand the potential of FPGAs in the absence of bottlenecks, it is instructive to consider the theoretical maximum floating-point performance of an FPGA.

Traditional processors, with a fixed data path width of 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. This principle is at the heart of the Top500 List of fastest supercomputers [31], where ranked machines must exactly reproduce valid results when running the LINPACK benchmarks. Many applications also require the full dynamic range of the double-precision format to ensure numeric stability.

Due 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 [32–34]. 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 [14]. 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. The design is pipelined to a depth of 12, permitting operation at a frequency up to 200 MHz. 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 an absolute upper limit on FPGA's double-precision floating-point performance. Performance in any deployed system would be lower because of the addition of interface logic.

Table 2 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 1419 logic slices and nine dedicated multipliers hold for all families. While the slice requirements may be less for the Virtex-4 family, owing to the inclusion of an MAC function with the dedicated multipliers, as all considered Virtex-4 implementations were multiplier limited the overestimate in required slices does not affect the results. The clock frequency

Table 2: Double-precision floating-point multiply accumulate cost-performance in US dollars.

Device	Speed (MHz)	GFlops	Device cost	$/GFlops
xc4vlx200	280	5.6	$7010	$1,250
xc4vsx35	280	5.6	$542	$97
xc2vp100-7	200	12.4	$9610	$775
xc2vp100-6	180	11.2	$6860	$613
xc2vp70-6	180	8.3	$2780	$334
xc2vp30-6	180	3.2	$781	$244
xc3s5000-5	140	3.1	$242	$78
xc3s4000-5	140	2.8	$164	$59
ClearSpeed CSX 600	N/A	50 [36]	$7500 [37]	$150
Pentium 630	3000	3	$167	$56
Pentium D 920	2800 × 2	5.6	$203	$36
Cell processor	3200 × 9	10 [38]	$230 [39]	$23
System X	2300 × 2200	12 250 [31]	$5.8 M [40]	$473

has been scaled by a factor obtained by averaging the performance differential of Xilinx's double-precision floating-point multiplier and adder cores [35] across the different families.

For comparison purposes, several commercial processors have been included in the list. The peak performance for each processor was reduced by 50%, taking into account compiler and system inefficiencies, permitting a fairer 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 [31]. In the table, FPGAs are assumed to sustain their peak performance.

As can be seen from the table, FPGA double-precision floating-point performance is noticeably higher than for traditional Intel processors; however, considering the cost of this performance processors fare 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 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 two exceptions, ClearSpeed and System X, all costs in Table 2 only cover the price of the device not including other components (motherboard, memory, network, etc.) that are necessary to produce a functioning supercomputer. It is also assumed here that operational costs are equivalent. These additional costs are nonnegligible 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. However, as most clusters incorporating FPGAs also include a host processor to handle serial tasks and communication, it is reasonable to assume that the cost analysis in Table 2 favors FPGAs.

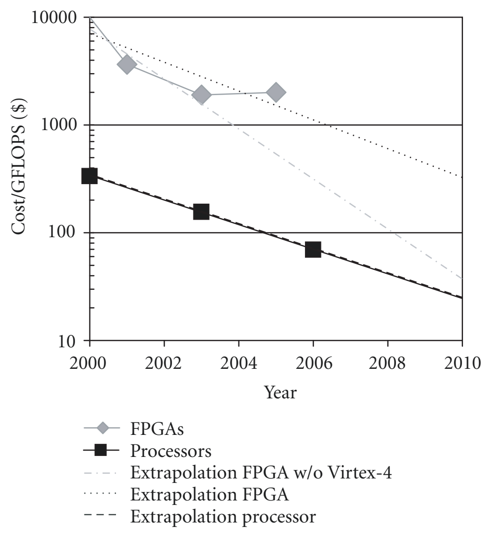
To place the additional component costs in perspective, the cost-performance for Virginia Tech's System X supercomputing cluster has been included [41]. Constructed from 1100 dual core Apple XServe nodes, the supercomputer, including the cost of all components, cost US$473 per GFLOPS. Several of the 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 [35] 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.

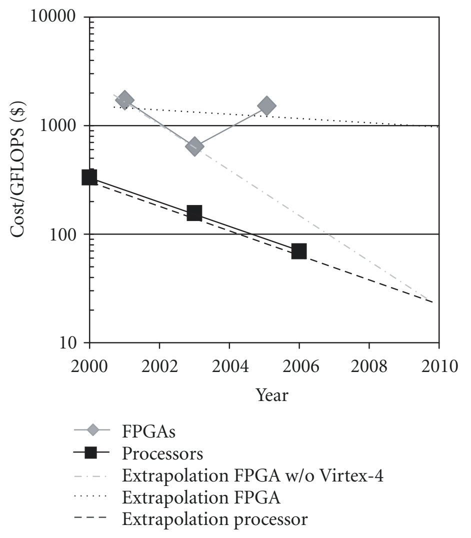
While the larger FPGA devices that are prevalent in computational accelerators do not provide a cost benefit for the double-precision floating-point calculations required by the HPC community, historical trends [42] suggest that FPGA performance is improving at a rate faster than that of processors. The question is then asked, when, if ever, will FPGAs overtake processors in cost performance?

As has been noted by some, the cost of the largest cutting-edge FPGA remains roughly constant over time, while performance and size improve. A first-order estimate of US$ 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 US$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. [14], 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 [42] to include the Virtex-4 family. Underwood's



(a)



(b)

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

data came from his IEEE standard floating-point designs pipelined, depending on the device, to a maximum depth of 34. The results are shown in Figure 1(a) for the Underwood data and Figure 1(b) for Dou et al.

An additional data point exists for the Underwood graph as his work included results for the Virtex-E FPGAs. 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 subfamilies: the LX, SX, and FX. The Virtex-4 subfamily 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 subfamily is focused on logic density, trading most of the dedicated multipliers found in the smaller SX subfamily for configurable logic. This significantly reduces the floating-point multiplication performance of the larger Virtex-4 devices.

As the graphs illustrate, 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. However, as preliminary data on the next-generation Virtex-5 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 1 trend lines labeled "FPGA extrapolation w/o Virtex-4" exclude these potential misleading data points.

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 when the largest FPGA devices, like those typically found in commercial FPGA-augmented HPC clusters, will be cost effectively compared to processors for double-precision floating-point calculations.

2.3. Tools

The typical HPC user is a scientist, researcher, or engineer desiring to accelerate some scientific application. These users are generally acquainted with a programming language appropriate to their fields (C, FORTAN, MATLAB, etc.) but have little, if any, hardware design knowledge. Many have noted the requirement of high-level development environments to speed acceptance of FPGA-augmented clusters. These development tools accept a description of the application written in a high level language (HLL) and automate the translation of appropriate sections of code into hardware. Several companies market HLL-to-gates synthesizers to the HPC community, including impulse accelerated technologies, Celoxica, and SRC.

The state of these tools, however, as noted by some [43], does not remove the need for dedicated hardware expertise. Hardware debugging and interfacing still must occur. The use of automatic translation also drives up development costs compared to software implementations. C compilers and debuggers are free. Electronic design automation tools, on the other hand, may require expensive yearly licenses. Furthermore, the added inefficiencies of translating an inherently sequential high-level description into a parallel hardware implementation eat into the performance of hardware accelerators.

3. FLOATING-POINT ALTERNATIVES

3.1. Nonstandard data formats

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 2D fast Fourier transform (FFT) through the use of a custom 18-bit floating-point format [44]. More recent work has focused on parameterizible libraries of floating-point units that can be tailored to the task at hand [45–47]. By using a custom floating-point format sized to match the width of the FPGA's internal integer multipliers, a speedup of 44 was achieved by Nakasato and Hamada for a hydrodynamics simulation [48] 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 US$ 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. Many comparisons spend significantly more time optimizing hardware implementations than is spent optimizing software. Significant compiler inefficiencies exist for common HPC functions [49], with some hand-coded functions outperforming the compiler by many times. It is possible 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 software at the assembly level. However, to permit their design to be more cost-competitive, even against efficient software implementations, smaller more cost-effective FPGAs could be used.

3.2. GIMPS benchmark

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 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) [50]. 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 and avoids the inaccuracies inherit in floating-point math.

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 was created to identify large Mersenne primes and a reward of US$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.

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. 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 elementwise 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 [51]. Performing a traditional port of the algorithm from software to hardware involves the creation of a floating-point FFT on the FPGA. On an XC2VP100, the largest Virtex-II Pro, 12 near-double-precision complex multipliers could be created from the 444 dedicated integer multipliers. Such a design with pipelining performs a single iteration of the Lucas-Lehmer test in 3.7 million clock cycles.

To leverage the advantages of a configurable architecture an all-integer number theoretical transform was considered. In particular, the irrational base discrete weighted transform (IBDWT) can be used to perform integer convolution, serving the exact same purpose as the floating-point FFT in the Lucas-Lehmer test. In the IBDWT, all arithmetic is performed modulo a special prime number. Normally modulo arithmetic is a demanding operation requiring many cycles of latency, but by careful selection of this prime number the reduction can be performed by simple additions and shifting [51]. The resulting all-integer implementation incorporates two 8-point butterfly structures constructed with 24-64-bit integer multipliers and pipelined to a depth of 10. A single iteration of Lucas-Lehmer requires 1.7 million clock cycles, a more than two-fold improvement over the floating-point design.

The final GIMPS accelerator, shown in Figure 2 implemented in the largest Virtex-II Pro FPGA, consisted of two butterflies fed by reorder caches constructed from the internal memories. To prevent a memory bottleneck, the design assumed four independent banks of double data rate (DDR) SDRAM. Three sets of reorder buffers were created out of the dedicated block memories on the device. These memories operated concurrently, two of the buffers feeding the butterfly units while the third exchanged data with the external SDRAM. The final design could be clocked at 80 MHz

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS

Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS

Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS

Sync your system to PACER to automate legal marketing.