Accelerating Scientific Computations using FPGAs

Oliver Pell, Lee W. Howes, Kubilay Atasu, Olav Beckmann, Oskar Mencer
Imperial College London, UK
{op, lwh01, atasu, ob3, oskar}@doc.ic.ac.uk

Abstract

Conventional processors are hitting the limits of attainable clock frequencies and thus, future significant increases in performance must come from exploiting parallelism. FPGA Accelerators have been shown to have the potential to speedup highly parallel applications by an order of magnitude and recent FPGA technology provides sufficient resources to tackle scientific applications on large-scale parallel systems. As a case study we implement the Fast Fourier Transform on FPGA and show that significant speed-ups are possible over general purpose processors.

Motivation

As microprocessors are hitting the limits of attainable clock frequencies and acceptable power consumption, we see a significant inflection point in the high performance computing environment. Intel and AMD are scaling up the number of cores per chip and processors per node in order to enable higher degrees of parallelism. Existing software has to be modified to take advantage of potentially only modest speed improvements. The change in software presents an opportunity to move beyond conventional processors to custom accelerators with the potential of much higher performance. Field-Programmable Gate Array (FPGA) accelerators can speedup highly parallel applications by an order of magnitude.

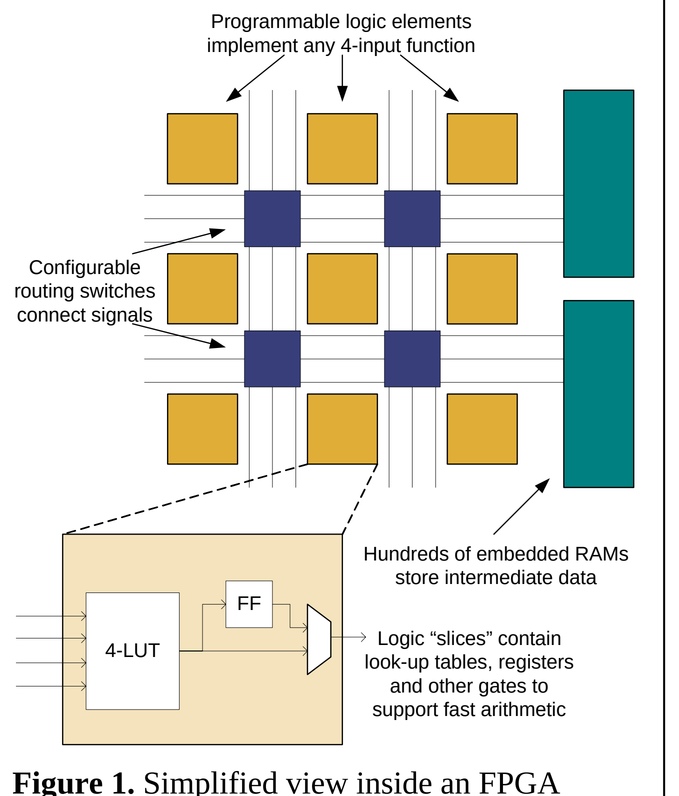
Programmable logic elements implement any 4-input function
Configurable routing switches connect signals
Hundreds of embedded RAMs store intermediate data
FF
4-LUT
Logic "slices" contain look-up tables, registers and other gates to support fast arithmetic

Figure 1. Simplified view inside an FPGA

FPGAs are semiconductor devices that contain a grid of programmable cells as shown in Figure 1, which the user configures to implement any digital circuit of up to a few million gates. Modern FPGAs allow the user to reconfigure these circuits many times each second, making FPGAs fully programmable and general purpose. The price of reconfigurability is a 10x slower dynamic clock frequency [1] compared to today’s state-of-the-art Pentium and Opteron processors. This slower clock frequency is compensated for by support for massive fine-grained parallelism.

Building Hardware Accelerators with FPGAs

Programming FPGAs remains essentially a hardware design task and is generally much harder than writing software. High level synthesis tools attempt to bridge the gap between algorithm and implementation, however the circuits that are obtained usually compare poorly with hand-crafted implementations.

ASC [2], *A Stream Compiler*, is designed to enable rapid development of hardware accelerators [3] while still producing results that match hand-crafted equivalents. ASC was developed following research at Stanford University and Bell Labs, and is now commercialized by Maxeler Technologies [4].

An ASC program represents a dataflow system which can be seen as a stream processor. Stream architectures are constructed using a C++ based object oriented approach, using an input description in C++ with ASC semantics.

In contrast to other methodologies, ASC provides the productivity of high-level design tools and the performance of low-level optimized hardware by permitting optimization at the algorithm, architecture and arithmetic levels. To aid optimization of implementations ASC provides a variety of objects implementing common design patterns. These object orientated abstractions provide for a high level of code reuse and allow highly optimized accelerators to be developed with minimal effort.

For implementing floating point arithmetic ASC provides a ‘HWfloat’ data-type that supports flexible floating point formats allowing mantissa and exponent bitwidths to be selected depending on the precision requirements of the application.

Fast Fourier Transform

As a case study, we implement the Fast Fourier Transform [5] in a flexible floating point implementation. The FFT circuitry described in ASC can be targeted to a variety of FPGA platforms in FFTW-style, though not yet completely automatically, allowing it to be adapted to the particular resources available on the system. The optimal implementation of an FFT accelerator depends on the length and dimensionality of the FFT, the available FPGA area, the available hard DSP blocks, the FPGA board architecture, and the precision and range of the application [6]. ASC allows a few core hardware descriptions to generate hundreds of different circuit variants to meet particular speed, area and precision goals.

The key to achieving maximum acceleration of FFT computation is to match memory and compute bandwidths so that maximum use is made of computational resources. Modern FPGAs contain up to hundreds of independent SRAM banks to store intermediate results, providing ample scope for optimizing memory parallelism.

At 175Mhz, one of Maxeler’s Radix-4 FFT cores computes 4x as many 1024pt FFTs per second as a Pentium-IV Xeon machine running FFTW (Figure 2). Eight such parallel cores fit onto the largest FPGA in the Xilinx Virtex-4 family, providing a possible 32x speed-up over performing the calculation in software.

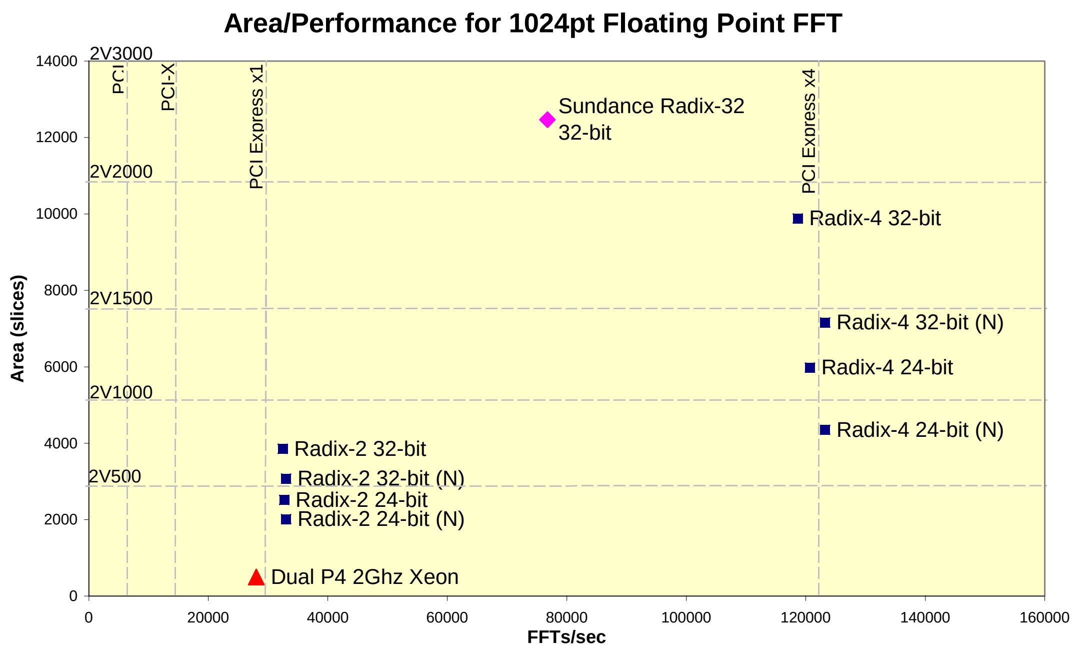

Figure 2. Area/performance trade-offs for eight FFT accelerators produced by the ASC system versus an alternative commercial Radix-32 core and a Pentium-IV software implementation. Horizontal dotted lines indicate the sizes of different chips in the Xilinx Virtex-II family, vertical dotted lines indicate the bus speed required to support that level of performance. Variants marked "(N)" contain fewer floating point normalization stages, trading some precision for decreased area and increased speed.

Our work at Imperial combines the Maxeler cores with a high performance FPGA computing platform from HP to demonstrate the potential of FPGAs for scientific computing applications. Clearly, performance depends on the communication bandwidth to the FPGAs and we can clearly see in Figure 2 that PCI Express x4 is well matched with current FPGAs. Further activities focus on investigation of larger radices and optimizations for longer transform lengths. Figure 3 below shows that FPGAs with a single core are competitive for transforms of more than 64 points by utilizing greater parallelism and reduced memory hierarchy overhead compared to CPUs.

Future Trends for FPGA Acceleration

The acceleration capabilities of FPGAs are dependent on the available silicon area to implement acceleration logic, on-chip memory capacity and the bandwidth of the bus which connects the accelerator to the host PC. In streaming applications, bus bandwidth is often the main bottleneck since FPGAs already contain ample compute resources, even for floating point applications. Industry benchmarks for the next versions of PCI Express and AMD Hypertransport indicate that bus capacity is likely to increase dramatically over the next few years.

By 2007 processing cards with the latest Xilinx Virtex-4 chips and PCI Express interfaces are expected to be available, offering significant improvements over currently available platforms. A

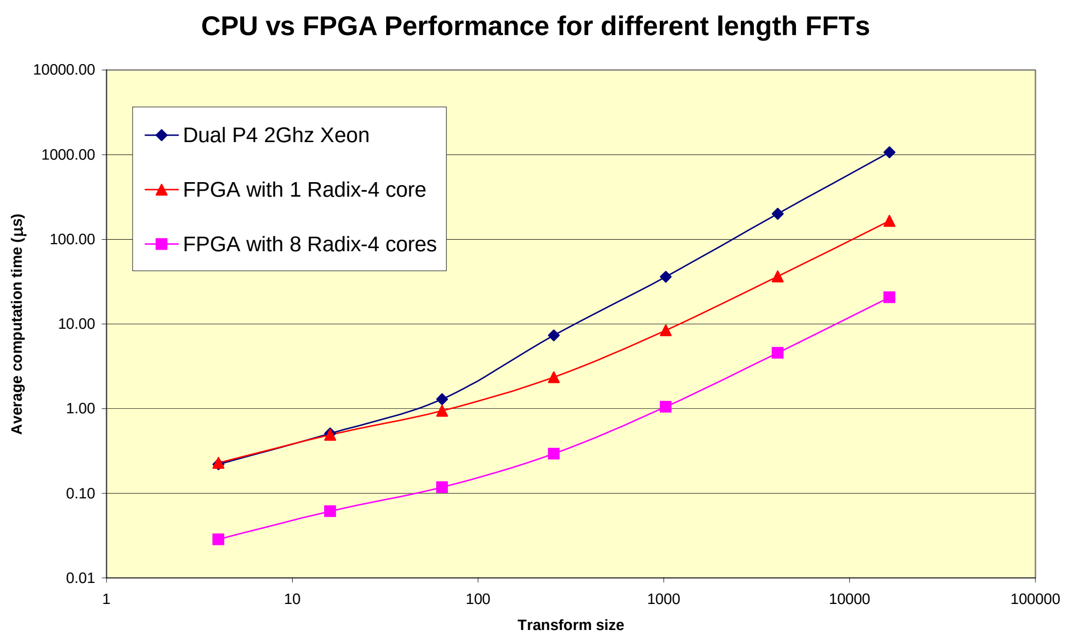

Figure 3. Comparison of FPGA and CPU floating point performance for transform sizes from 4 to 16,384 points.

further analysis of trends in FPGA capabilities and bus bandwidths shows that the advantages of using FPGAs are likely to increase over time, compared to single and multi-core processor implementations (Figure 4).

Conclusion & Future Work

FPGAs already offer significant capabilities to accelerate scientifically-useful computations, such as the Fast Fourier Transform. Improvements in FPGA and bus technology indicate that FPGAs will increase their lead over general purpose processors over the next few years.

Our future work on Fast Fourier Transform acceleration involves extending our work to 2 and 3-dimensional FFTs. Preliminary results using a relatively naïve hardware implementation show speed-ups of 3x over a 2Ghz AMD Operton running FFTW, with projected speed-ups of 23x for 2007 with an improved implementation.

References

1. J. Bower, O. Mencer, W. Luk, M.J. Flynn and M. Morf, *Dynamic Clock-Frequencies for FPGAs*. Elsevier Journal of Microprocessors and Microsystems, special issue, Sept 2006.

2. O. Mencer. *ASC, A Stream Compiler for Computing with FPGAs*. IEEE Trans. CAD, August 2006.

3. L. W. Howes, O. Pell, O. Mencer and O. Beckmann. *Accelerating the Development of Hardware Accelerators*. EDGE Computing Workshop, North Carolina, USA, May 2006.

4. Maxeler Technologies, *http://www.maxeler.com*

5. M. Frigo and S. G. Johnson. *The Design and Implementation of FFTW3*. Proc. IEEE, 93(2), 2005.

6. K. S. Hemmert, K. Underwood. *An Analysis of the Dobule-Precision Floating-Point FFT on FPGAs*. Proc. FCCM'05, IEEE Computer Society Press, 2005

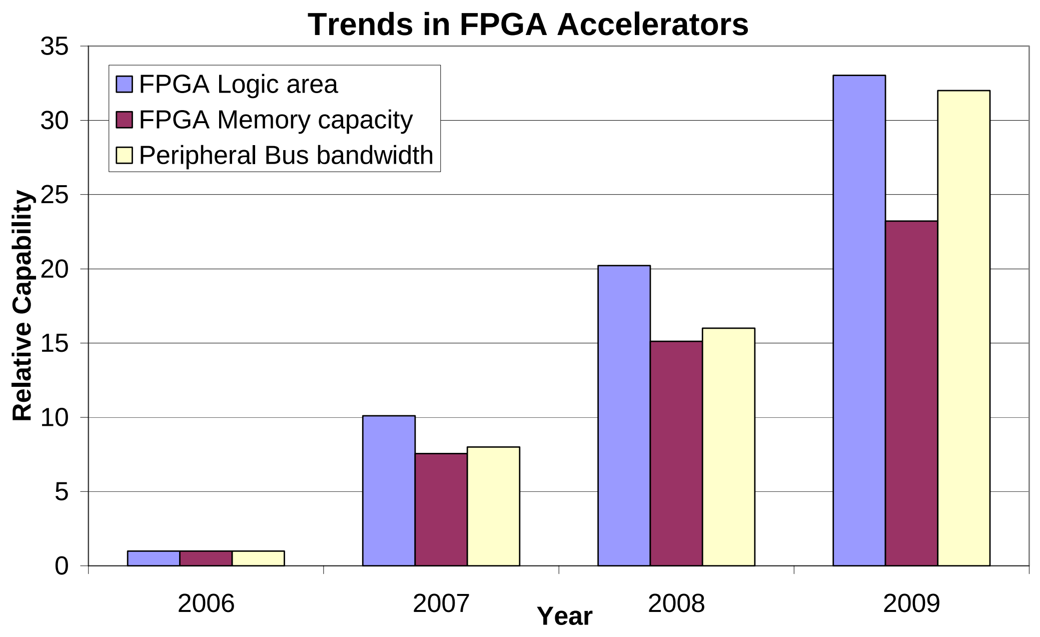

Figure 4. Projected growth in the capabilities of FPGA accelerators based on industry roadmaps, relative to a Xilinx Virtex-II based HP Sepia processing platform in 2006.