

# Accelerating the Development of Hardware Accelerators

Lee W. Howes, Oliver Pell, Oskar Mencer, Olav Beckmann  
Department of Computing, Imperial College London  
{lwh01, op, oskar, ob3}@doc.ic.ac.uk

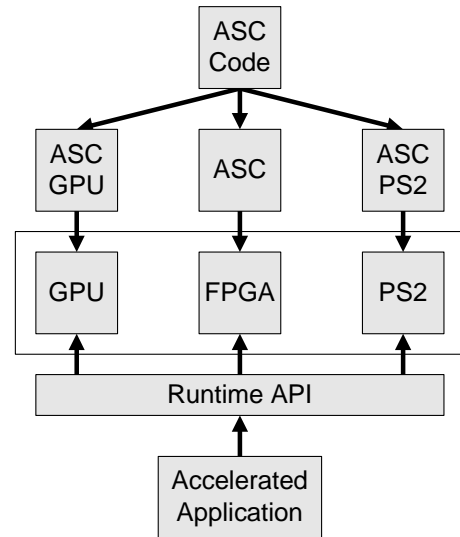
ASC [1], *A Stream Compiler*, is designed to enable rapid development of hardware accelerators while still producing results that match hand-crafted equivalents. An ASC program represents a dataflow system which can be seen as a stream; to avoid the difficulties often associated with behavioural synthesis, ASC allows direct implementation of a hardware design based on the programmer's input. Stream architectures are constructed using a C++ based object oriented approach, and as shown in Figure 1, a single ASC description can target Field Programmable Gate Arrays (FPGAs) and also other accelerator platforms, such as Graphics Processing Units (GPUs) and Sony Playstation 2 vector units. Architectural differences are transparent to the programmer in a basic implementation, requiring only the selection of an appropriate target architecture.

FPGAs show excellent potential as hardware accelerators for a wide class of applications. While running many times slower than modern Pentium-type processors, the massive fine-grained parallelism and many independent memory buses of FPGAs can permit order-of-magnitude speed-ups over software. However, programming FPGAs remains essentially a hardware design task and is 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.

While intended for graphics acceleration, modern GPUs are highly programmable parallel stream computation engines [4] and increasingly suitable for general computation [3]. The Sony Playstation 2 processor contains two programmable vector units that can execute independently, the ability of this system to accelerate applications is thus indicative of the potential of upcoming similar architectures such as Cell.

Through targeting multiple different acceleration architectures from a common description, we can compare the performance of each architecture for a particular algorithm. Once the best accelerator architecture has been selected, architecture specific extensions can be used to further optimise the implementation.

Developer access to multiple levels in the design hierarchy offers the ASC programmer great flexibility of implementation when required, permitting optimisation at the al-

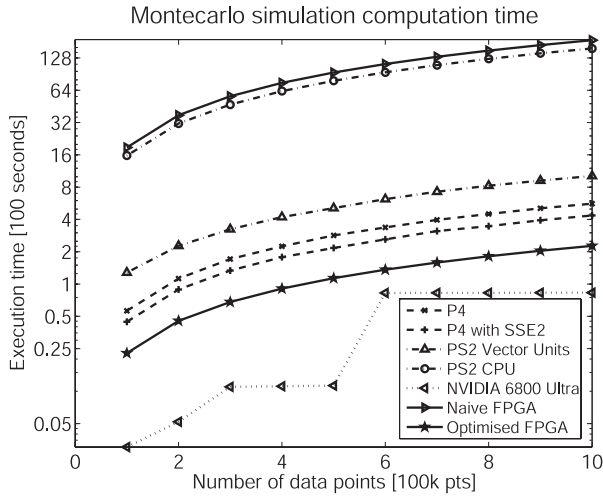


**Figure 1.** ASC can target multiple accelerator platforms from a single source description.

gorithm, architecture and arithmetic levels. To aid optimisation of implementations ASC provides a variety of objects implementing common optimisation design patterns. These object orientated abstractions provide for a high level of code reuse and allow highly optimised accelerators to be developed with minimal effort.

We illustrate the application of ASC to accelerate two sample applications and compare the performance of several different accelerator platforms.

Figure 2 shows the results obtained when comparing an accelerated implementation of a montecarlo simulation against software running on a Pentium 4. From the base implementation we can perform simple optimisations on each target to improve the performance. The “Optimised FPGA” line on the graph represents an optimisation of the FPGA implementation using on-chip data buffers, which are easy using ASC abstractions. In this case optimisation is necessary as the base Montecarlo implementation includes a loop which renders hardware pipelining impossible. The GPU is clearly the best accelerator in this application, particularly for small numbers of data points.

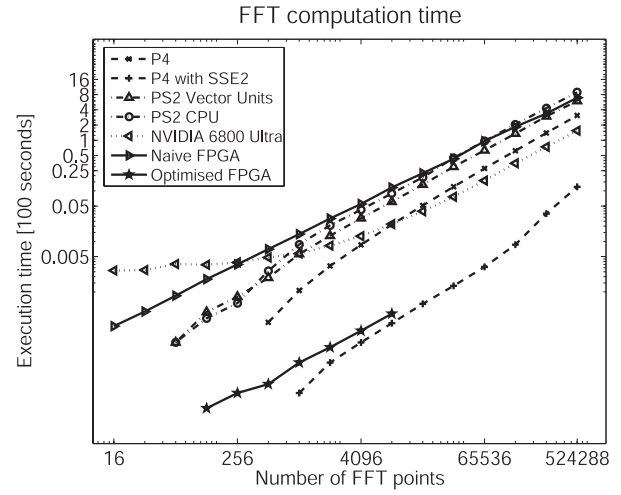


**Figure 2.** Montecarlo simulation on P4 3.2GHz with Intel C compiler, full optimisation. Compared with NVIDIA 6800 Ultra, PS2 vector units, PS2 CPU and Xilinx XC2V8000 FPGA.

Figure 3 shows the results for the implementation of a Radix-2 Fast Fourier Transform in single precision floating point. The FFT shows a wide performance range across the different platforms. In the simple implementation only the butterfly computation is accelerated, with memory rearrangement carried out by the CPU. This memory rearrangement scheme particularly penalises the GPU and unoptimised FPGA, however the FPGA performance is vastly improved by the same on-chip buffering optimisation as used for the montecarlo simulation. This suggests that FPGAs are the most promising platform for achieving significant acceleration of this application.

Maxeler Technologies is currently using ASC abstractions to develop a range of Fast Fourier Transform accelerator cores that substantially outperform reported implementations produced using high level tools [2]. 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 [5]. ASC facilitates this by allowing a few core hardware descriptions to generate hundreds of different circuit variants to meet particular speed, area and precision goals.

The key to maximising FFT acceleration is to balance memory and compute bandwidths so that maximum use is made of computational resources, while supporting computation with higher radices than Radix-2 to maximise parallelism. At 175Mhz, one of Maxeler’s Radix-4 FFT cores computes 4x as many 1024pt FFTs per second as a dual Pentium-IV Xeon machine running FFTW (Table 1). Eight such parallel cores fit onto the largest FPGA in the Xilinx Virtex-4 family, providing a 32x speed-up over performing the calculation in software.



**Figure 3.** FFT on P4 3.2GHz with Intel C compiler, full optimisation. Compared with NVIDIA 6800 Ultra, PS2 vector units, PS2 CPU and Xilinx XC2V6000 FPGA.

Points	CPU	1 FPGA core	8 FPGA cores
4	0.22	0.23	0.03
16	0.51	0.49	0.06
64	1.29	0.94	0.12
256	7.33	2.35	0.29
1024	36.24	8.43	1.05

**Table 1.** FFT computation time (in microseconds) for a range of transform sizes from 4–1024 points, comparing a dual Pentium-IV Xeon 2Ghz machine with a single, and multiple Maxeler Radix-4 FPGA cores running at 175Mhz.

## References

- [1] O. Mencer. *ASC, a stream compiler for computing with FPGAs*. IEEE Trans. CAD, 2006.
- [2] I. S. Uzun et al. *FPGA implementations of fast Fourier transforms for real-time signal and image processing* IEE Proc. Vis. Image Signal Process, 152 (3), 2005.
- [3] NVIDIA Corporation R. Fernando. *Trends in GPU evolution*. Eurographics, September 2004.
- [4] I. Buck et al. *Brook for GPUs: Stream computing on graphics hardware*. SIGGRAPH, 2004.
- [5] K. S. Hemmert and K. D. Underwood. *An analysis of the double-precision floating-point FFT on FPGAs* Proc. FCCM’05, IEEE Computer Society Press, 2005.