Maxwell – a 64 FPGA Supercomputer

Rob Baxter[1], Stephen Booth, Mark Bull, Geoff Cawood, James Perry, Mark Parsons, Alan Simpson, Arthur Trew
EPCC and FHPCA

Andrew McCormick, Graham Smart, Ronnie Smart
Alpha Data ltd and FPHCA

Allan Cantle, Richard Chamberlain, Gildas Genest
Nallatech ltd and FHPCA

[1] communicating author: r.baxter@epcc.ed.ac.uk; 0131 651 3579;
University of Edinburgh, James Clerk Maxwell Building, King's Buildings, Edinburgh EH9 3JZ

Abstract

We present the initial results from the FHPCA Supercomputer project at the University of Edinburgh. The project has successfully built a general-purpose 64 FPGA computer and ported to it three demonstration applications from the oil, medical and finance sectors. This paper describes the machine itself – Maxwell – its hardware and software environment and presents very early benchmark results from runs of the demonstrators.

1. Introduction

Against the background of possibilities in the emerging area of high-performance reconfigurable computing [1] the FPGA High Performance Computing Alliance (FHPCA [2]) was founded in early 2005 to take forward the ideas of an FPGA-based supercomputer. The alliance partners are Algotronix, Alpha Data, EPCC at the University of Edinburgh, the Institute for System Level Integration, Nallatech and Xilinx. The project was facilitated and part funded by the Scottish Enterprise Industries team and had two main goals:

- design and build a 64-FPGA supercomputer from commodity parts and "plug-in" FPGA cards;
- demonstrate its effectiveness (or otherwise) for real-world high-performance computing (HPC) applications.

We describe here the results of the first of these goals and report some early results of the second.

The machine itself – Maxwell – was completed in the first part of this year. We describe its architecture in Section 3; interestingly it shares a number of similarities with the proposed petascale Reconfigurable Computing Cluster described by Sass et al in [3].

This paper is structured as follows. Section 2 describes the motivation behind Maxwell. Section 3 delves into the details of the machine's hardware while Section 4 describes the software environment and programming methodology used in porting a number of demonstration applications. Section 5 discusses three key demonstration applications from the fields of financial services, medical imaging and oil and gas exploration., and Section 6 presents early performance results from these applications on Maxwell. Finally Section 7 offers thoughts for the future.

2. Motivation

Maxwell is designed as a proof-of-concept general-purpose FPGA supercomputer. Given the specialized nature of hardware acceleration the very concept of 'general-purpose' for high-performance reconfigurable computing (HPRC) is worth investigating in its own right. Can a machine built to be as broadly applicable as possible deliver enough FPGA performance to be worth the cost?

Our real interest in building Maxwell was not to test whether FPGA hardware can be used to accelerate segments of a standard HPC application, but to explore whether standard HPC applications can be run almost entirely on FPGA hardware, parallel communications and all. We take the same view as Bennett et al [4] in regarding the FPGAs as the primary compute platform rather than co-processors to a CPU. To this end we constructed a machine capable in principle of parallel operation across a network of large FPGAs linked directly together.

3. Hardware

Maxwell is essentially an IBM BladeCentre Cluster with FPGA acceleration. Altogether it comprises 32 blade servers each with one Intel Xeon CPU and two Xilinx Virtex-4 FPGAs. The CPUs are connected to the FPGAs with a standard IBM PCI-X Expansion Module.

3.1 BladeCentre chassis

Physically Maxwell comprises two 19-inch racks and five IBM BladeCentres. Four of the BladeCentres have seven IBM Intel Xeon blades and the fifth has four. Each blade is a diskless 2.8 GHz Intel Xeon with 1 GB main memory. The blades are connected over gigabit Ethernet through a single 48-way Netgear switch with 40 Gb/s throughput. The blades are booted over the network from the headnode, a plain old Dell Precision 670 with 4 GB main memory and over 1 TB of local SATA disk.

The chassis is thus a fairly standard commodity setup – deliberately so since our intention was to investigate the viability of building a high-performance cluster from standard parts and plug-in FPGA cards from two independent vendors – Nallatech and Alpha Data.

Logically we regard Maxwell as a collection of 64 nodes, where a node is defined as a software process running on a host CPU, together with some FPGA acceleration hardware. In full operation each blade CPU thus hosts two software processes, each of which 'manages' one FPGA during runtime.

One obvious drawback to Maxwell's architecture is the diskless nature of the blades; all disk i/o traffic routes through the Ethernet switch, offering an instant performance bottleneck for data intensive applications. However, Maxwell is intended as a demonstration and exploration platform for FPGA-to-FPGA computing rather than a production system, so this is a design compromise we can live with just now.

3.2 FPGAs

The FPGAs in Maxwell are Xilinx Virtex-4 devices in two flavours. Those on the Alpha Data cards are XC4VFX100 parts, while those on the Nallatech cards are XC4VLX160. The reasons for this are more logistical and political than technical.

The LX flavours of Xilinx's Virtex range offer the greatest number of logic cells, while the FX flavours include embedded PowerPC cores and multigigabit serial transceivers (MGTs) ("RocketIO") for off-chip communications [5]. The V4LX160s each have 152,064 logic cells against the V4FX100s' 94,896. This makes them pretty large by current FPGA device standards, allowing room to implement significant pieces of HPC code on them. Given the mixed nature of devices in the machine we have so far not used the PowerPC cores on the FX100s.

These two flavours of Virtex-4 are built into two flavours of plug-in PCI card: the Nallatech H101 and the Alpha Data ADM-XRC-4FX.

Both types of card connect using a PCI/PCI-X bridge, capable of 64 bit, 133MHz operation in PCI-X mode – a peak bandwidth of 600 MB/s. PCI-X is by no means an ideal connection technology to use – PCI Express is capable of 8 GB/s on a 32-lane connection – and is another potential performance bottleneck on Maxwell – indeed on many PCI/PCI-X based FPGA cards. However, our approach to programming Maxwell aims to remove the CPU-FPGA connection from critical code paths by performing the bulk of calculations purely on the FPGA, so again we regard this as an acceptable design tradeoff.

3.3 Nallatech H101

The cards in one half of Maxwell are a slight variant on Nallatech's off-the-shelf H101-PCIXM [6]. The standard card uses V4100LX devices; Maxwell uses the 60% bigger V4160LX versions, with a peak clock speed of 200 MHz. There are 32 of these cards, occupying PCI slots in 16 of the blades.

Along with the V4LX160 the H101 has 16 MB of DDR-II SRAM in four banks, and one 512 MB bank of DDR-II SDRAM. The four SRAM banks deliver a peak bandwidth of 6.4 GB/s, while the SDRAM delivers 3.2 GB/s.

Observant readers will have noticed that the V4LX devices in the H101s have no serial MGTs and thus no means of accessing the outside world. Quite so. Communication links from the Nallatech cards are achieved through a separate comms chip, a small Virtex-II Pro FX device with an embedded router core. Each H101 card thus has four MGT links capable of running at 2.5 Gb/s.

3.4 Alpha Data ADM-XRC-4FX

The other 16 blades in Maxwell host 32 Alpha Data ADM-XRC-4FX cards [7]. The ADM-XRC-4FX is a high performance reconfigurable PMC/PMC-X/XMC (PCI Mezzanine Card) based on the Xilinx Virtex-4-FX range. The Maxwell cards are the V4FX100 variant.

As with the Nallatech cards the ADM-XRC-4FX have 16 MB of SRAM but double the SDRAM at 1,024 MB of DDR-II in four banks. This gives a peak memory bandwidth on 8.4 GB/s to the SDRAM.

Off-chip communication is direct from the V4FX devices – again, four RocketIO MGTs each with a maximum possible bandwidth of 3.125 GB/s.

3.5 Communications networks

As suggested thus far, Maxwell has two independent communications networks. The blade CPUs are networked over standard gigabit Ethernet via a single switch; the CPUs thus have an all-to-all connectivity. This is contrasted with the FPGA network which consists

of point-to-point links between the MGT connectors of adjacent FPGAs, as illustrated in Figure 1. The FPGA pairs hosted on a single CPU form "east-west" pairs in the network.

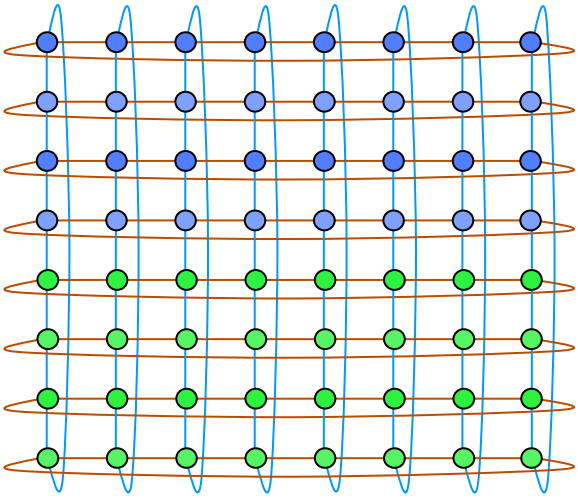

Figure 1. FPGA connectivity in Maxwell

The MGTs are connected with standard HSSDC2 1x-1x Infiband cables of 50cm and 100cm lengths, kept as short as possible.

The FPGA connections are purely point-to-point – we do not implement routing logic in the FPGA devices. Maxwell's FPGAs thus form a two-dimensional torus, making the RocketIO network highly suitable for nearest-neighbour communication patterns but less than ideal for reduction operations such as global sums. For these, applications call back to the host CPUs for MPI reduction operations to be performed over Ethernet.

Our general approach is to use the Ethernet network purely as a control network and to perform parallel communications over RocketIO. The Ethernet is also used for any explicit MPI calls that remain in the application – for instance for start-up data distribution or finalizing data marshalling on completion.

The hard-wired nature of the FPGA communications network contrasted with the implicit all-to-all CPU network means that care is required in constructing application configuration and acceleration components in software. This is one thing we have tried to address in the Parallel Toolkit software environment (Section 4 below).

4. Software environment

Our aim with the environment on Maxwell was to make it as 'HPC-system-like' as possible. Uptake of FPGA and related technologies in HPC will be hindered if the machines require a whole different approach to that of 'mainstream' HPC.

In the early days of parallel computing every vendor had their own way of doing things and progress for application developers was slow. One vendor's communication protocols did not map onto another's and machine environments were very different. Today, parallel environments are much more standardized and there is a lot of 'legacy' software built using libraries like MPI, BLAS and SCALAPACK. Machines requiring significantly different approaches will not find much traction.

Thus Maxwell looks very much like any other parallel cluster. It runs the Linux variant CentOS and all standard Gnu/Linux tools. It offers Sun Grid Engine as a batch scheduling system and MPI for inter-process communication.

The one novel innovation is the Parallel Toolkit (PTK). The PTK has been developed as part of the overall design of Maxwell and provides an attempt to enforce top-down standardization on application codes across the different 'flavours' of hardware. Unfortunately we do not have space here to go into detail, but the PTK is described in [8]. Suffice to say that it is a set of practices and infrastructure intended to address key issues in hardware acceleration such as associating processes with FPGA resources, associating FPGAs with bitstreams, managing contention for FPGA resources within a process and managing code dependencies to facilitate re-use.

4.1 FPGA programming

While providing a useful, common way of configuring FPGAs from software the PTK does not address the deeper portability issues for running applications on different flavours of FPGA hardware. Maxwell still requires developers to build their FPGA bitstreams against either Nallatech H101 or Alpha Data ADM-XRC-4FX cards offline and copy the bitfiles across to the system. We do not mandate any particular tool approach for FPGA developers – any tool capable of targeting the two card types will produce a bitstream that can be run on Maxwell.

5. Demonstration applications

As part of the overall project we have ported three demonstration applications to run on Maxwell. Each of these has been produced in two hardware 'flavours' for the two halves of the machine, with a common high-level interface to software captured in the Parallel Toolkit.

Two criteria were used to select these applications. Firstly they were chosen from the application areas of financial engineering, medical imaging and oil and gas, three areas judged generally to have most to gain from hardware acceleration solutions of one form or another.

Secondly they were chosen to illustrate progressively more complex parallel application features, from trivial parallelism and simple data requirements to full-scale distributed-memory parallelism.

In all three cases we adopted the methodology of the PTK: identify the application hotspot; define an abstract object-oriented interface that encapsulates the hotspot; refactor the code against this interface; generate accelerated versions of the hotspot code underneath the interface. In each case we also produced a 'pure software' version of the hotspot against the same PTK interface, providing a direct point of comparison for both testing and benchmarking purposes.

5.1 MCopt – Monte Carlo option pricing

Financial engineering is a mathematical branch of economics that deals with the modeling of asset prices and their associated derivatives. One of the cornerstones of financial engineering is the Black-Schole model of prices [9], essentially a recasting of the equations of physical heat diffusion and Brownian motion.

The assumptions of the Black-Scholes model imply that for a given stock price at time t, simulated changes in the stock price at a future time t + dt can be generated by the following formula:

$$dS = S\, r_c\, dt + S\, \sigma\, \varepsilon\, \sqrt{dt}$$

where S is the current stock price, dS is the change in the stock price, r_c is the continuously compounded risk-free interest rate, σ is the volatility of the stock, dt is the length of the time interval over which the stock price change occurs and ε is a random number generated from a standard Gaussian probability distribution.

The pricing of stock options follows the Black-Scholes model, and simple stock options (so-called 'European' options) can be priced with a simple closed-form formula called, unsurprisingly, the Black-Scholes formula. More complex options such as those whose final price depends on a time-average or other path-dependent price calculations have no closed form and are typically priced using stochastic or Monte Carlo modeling.

Our first demonstration applications supposes you wanted to price an 'Asian' option, an option in which the final stock price is replaced with the average price of the asset over a period of time, computed by collecting the daily closing price over the life of the option. The price can be modeled as a series of dSs over the option's lifetime (say $N_{timesteps}$). The formula for each dS is based on the previous day's closing price, and the average of the $N_{timesteps}$ stock prices would determine the value of the option at expiration.

The above gives you one possible future for the stock price; repeating the model N_{runs} times allows the process to converge on the 'right' option price. N_{runs} here is of order 10,000 – 50,000.

Based on the above model, a serial code would be as follows:

```
for i = 1, N_runs
  for n = 1, N_timesteps
    ε = gaussianRandomNumber()
    S[n][i] = S[n-1][i] (1 + r_c dt + σ ε √dt)
  endfor n
  S_av[i] = 1/N_timesteps Σ_n S[n][i]
  c[i] = max(S_av[i] - K, 0)
  p[i] = max(K - S_av[i], 0)
endfor i
S_bar = 1/N_runs Σ_i S_av[i]
c_final = 1/N_runs Σ_i c[i]
p_final = 1/N_runs Σ_i p[i]
```

K here is the strike price of the option, the price defined in the option contract; r_c and σ are as defined above.

Pricing a single Asian option thus requires $N_{runs} \times N_{timesteps}$ Gaussian random numbers (plus four multiplies and three adds each).

Our demonstration captures this whole core, parameterized, on FPGA, and batches similar pricing calculations for different stocks/assets across the whole 64 FPGAs. In fact the demonstrator core is so compact that it can be replicated 10 times or more across a single FPGA device, providing an additional order of magnitude in possible speedup.

This demonstrator – MCopt – is the simplest of the three applications, having a simple, compact computational core and very limited data requirements.

5.2 DI3D – three and four-D facial imaging

The second demonstration application was produced in collaboration with Dimensional Imaging 3D ltd, a firm specializing in three and four-dimensional facial imaging for medical applications such as maxilo-facial surgery [10].

The principle here is that a digital camera rig is used to capture pairs of still (for 3D) or video (for 4D) images. Each stereo pair is then combined to produce a depth map which contains full three-dimensional information and is used to create a 3D software model, or a 3D video in the case of 4D capture.

Image combination is an expensive business and is an ideal application for FPGA acceleration, playing well to the devices' strengths in image processing. Our demonstrator thus takes a key part of Dimensional Imaging's own software and accelerates it using FPGA hardware. Two versions of the demonstrator were produced – an embedded version which can connect to a live camera rig and provide on-the-fly image combination,

and a batch version designed to process large numbers of image pairs from video frames.

This latter version is designed to run across all 64 FPGAs of Maxwell and provides the next step-up in complexity. While as trivially parallel as the MCopt application this demonstrator has real data requirements – large digital images must be managed and streamed through the FPGAs in an efficient manner to ensure overall performance.

5.3 OHM3D – CSEM modeling

Our final demonstrator is another commercial code, this time in the area of oil and gas exploration. OHM plc are an Aberdeen-based consultancy offering services to the oil and gas industry [11]. OHM specializes in a form of simulation called controlled source electromagnetic modeling, a technique which uses the conductive properties of materials to analyse pieces of the seabed in the search for oil or gas reserves [12].

OHM's three-dimensional CSEM code provides the basis for our final demonstrator. Already parallelized using MPI, this is a 'classic' HPC application involving large data sets representing physical spaces and fields, double-precision arithmetic and iterative numerical methods for performing linear algebra operations on large matrices and vectors.

6. Initial performance results

This section presents our *preliminary* results from early tests of the three demonstrator applications on Maxwell. At the time of writing this section is only partially complete since the final versions of the demonstrators are still undergoing test.

All results quoted in this section were run on Maxwell. The quoted CPU results are thus for the 2.8 GHz Intel Xeon processors in the IBM blades. In caption legends we use the label 'AD' to refer to the Alpha Data hosted FPGAs, and 'NT' to refer to the Nallatech hosted devices.

6.1 MCopt

MCopt is the simplest of the demonstrators, a trivially-parallel engine to explore the parameter space of a typical option pricing calculation.

Our tests for MCopt aim to explore the five-dimensional parameter space defined by the variable input parameters to the Monte Carlo version of the Black-Scholes model (S, K, r_c, σ, $N_{timesteps}$). Our test draws 100,000 data samples from this parameter space; we fix the number of Monte Carlo iterations, N_{runs}, to 10,000 for each sample.

The single-node performance is shown in Figure 2, and Figure 3 shows MCopt run across 1 to 16 nodes, both CPU and FPGA

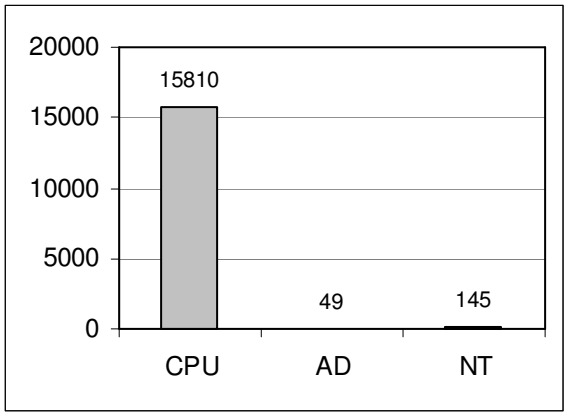

Figure 2. Single-node MCopt performance (s).

In Figure 3 the logarithmic scale belies the extreme scalability of the FPGA versions here: a batch of 100,000 parameters (prices, rates or some combination of these) that would take over 4 ½ hours on a Xeon blade runs in less than a minute on one FPGA, or less than 3 seconds on 16. As might be expected with such a simple calculation the FPGAs outperform the CPU by over two orders of magnitude – a factor of 300 in the Alpha Data case.

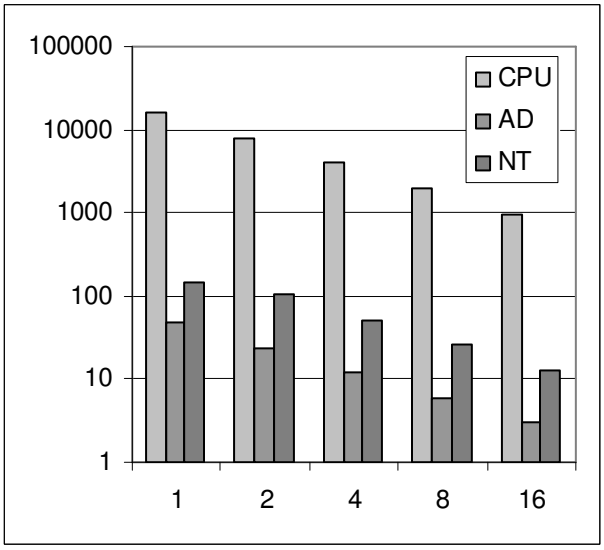

Figure 3. MCopt scaling performance on Maxwell (times in s). Note the scale is logarithmic.

6.2 DI3D – facial imaging

The facial imaging demonstrator, while still trivially parallel in execution, is more challenging than the MCopt application because of its data requirements.

Our tests here involve the batch processing of 32 pairs of video still images – 64 in all – each around 150 kB in size. This represents a little over a second of three-dimensional video. The images are read in from networked disk, so this is also an interesting test of the network and i/o overheads of the parallel system.

Figure 4 shows the single node performance for these tests. Note that at the time of writing only times from the Nallatech side of the machine have been generated.

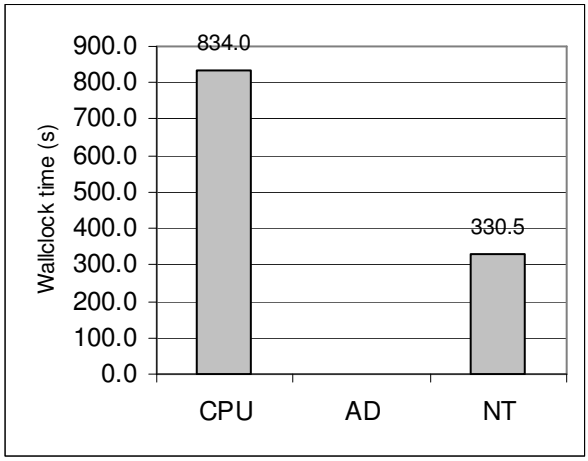

Figure 4. Single node DI3D facial imaging performance (times in s). Only the Nallatech results are available.

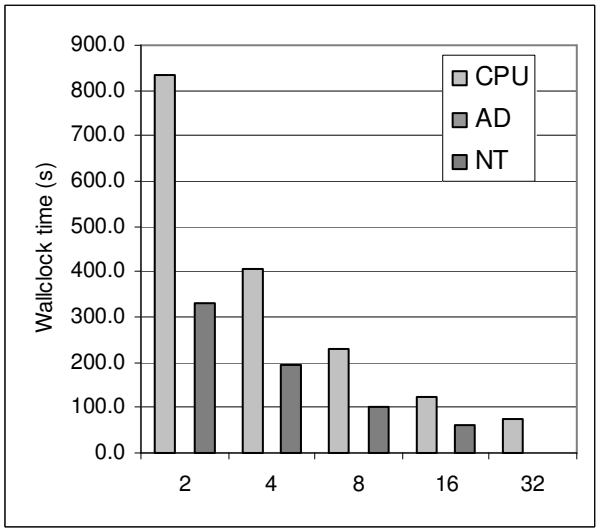

Figure 5. Facial imaging scaling performance on Maxwell (times in s). Again, only the Nallatech results are available.

Figure 5 plots the runtime for the same test against increasing numbers of nodes. In both Figures 4 and 5 timings were made using the standard MPI timer function MPI_Wtime() [13] across the full batch-processing version of the application. This includes software-only components not accelerated in hardware.

The overall application speedup is around a factor of 2.5. When we restrict timings to the accelerated kernel against its software-only equivalent the speedup of the FPGA version is around 3.6. While respectable in terms of general application optimization it is hard to make a case for the cost-effectiveness of the FPGA solution over a faster CPU and motherboard here.

The final plot in Figure 6 shows the speedup curve from the scaling results. Note the falloff towards 16 processors for both versions – we suspect that this is a feature of i/o bandwidth saturation across the machine although we have not investigated in detail.

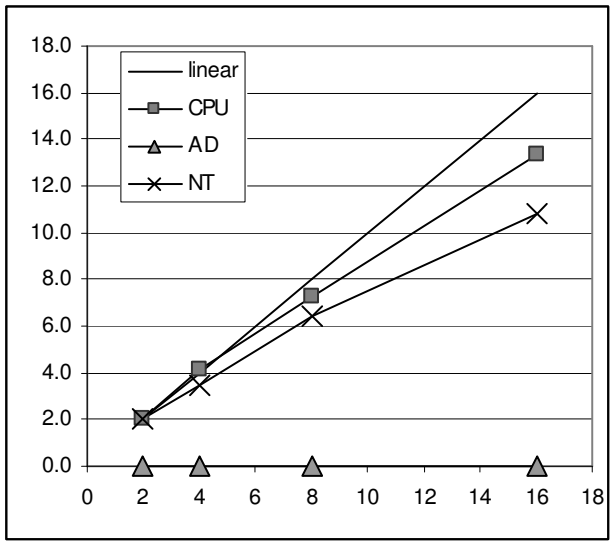

Figure 6. Parallel speedup of the DI3D facial imaging code on Maxwell. AD results are not yet available.

6.3 OHM3D – CSEM modeling

The most challenging of the demonstrators is the OHM3D CSEM modeling application, a fully-parallelised classic HPC simulation. Here we used a sample dataset of size 50×45×200 (450,000 points) and ran the solver until it converged on a solution (c. 1000 iterations in both CPU and FPGA cases).

Due to the preliminary nature of this work we have at the moment only one datapoint for the OHM3D code – Figure 7 shows this performance across eight nodes of Maxwell, comparing CPU-only runs with FPGA runs on the Alpha Data side of the machine. Such are the memory requirements of this code that it is unable to run on fewer than 4 CPUs or 8 FPGAs.

This preliminary result suggests the FPGA version runs some 4.8 times faster per node than the pure software version. Since we are currently unable to report on the scaling properties of the FPGA versions of this code it is

difficult to draw strong conclusions about the effectiveness of the accelerated versions over pure software. Component test results do suggest that the RocketIO implementations of the boundary swap at the heart of the parallel kernel are at least as efficient as the MPI-based versions over gigabit Ethernet. Thus we can expect that the FPGA versions will scale approximately as effectively as the CPU versions, and can conjecture that the factor of 4.8 performance improvement seen in the eight node version will persist on larger numbers of FPGAs.

This ability to scale with increasing numbers of nodes is critical if FPGA solutions are to be competitive with parallelized software; the test of this for OHM3D is left for further work.

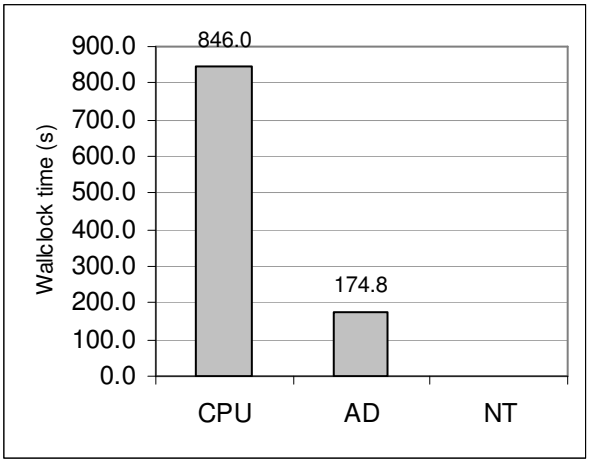

Figure 7. OHM3D 8-node performance in seconds. Nallatech figures are unavailable at time of writing.

6.4 A note on development costs

The potential performance gains from FPGA codes must be weighed against the non-recurrent software engineering costs of porting them to a particular hardware platform in the first place. For the work presented here, unravelling detailed engineering costs is a little complex, but approximately: MCopt took a few staff-weeks to develop both software harness and FPGA hardware; DI3D took six staff-months; and OHM3D took 12 staff-months and is as yet untested on more than eight FPGAs.

In software engineering terms these costs are high. In each case the software portions of these codes were ported by software engineers from EPCC and the FPGA portions by hardware engineers at Alpha Data and Nallatech. Even playing to the strengths of the teams in this way the larger applications proved challenging to port across. We have identified a number of reasons for this:

- early development was done on Virtex-II Pro platforms; the transition from this to Virtex-4 was by no means straightforward;
- the high-level "C-to-gates" tools used are still maturing. Difficulty with tools, again partly in changing hardware versions, slowed development;
- the size of the FPGAs brings challenges in two areas:
  - synchronizing timing across such large devices proves more difficult than with the earlier, smaller generations;
  - compilation (place-and-route) times for these codes is of the order of 6-8 hours;
- memory management for data-bound HPC applications is the key to performance, and a difficult thing to get right.

In our opinion addressing these issues must be a priority if FPGA computing is to gain more traction in the HPC or general-purpose computing fields.

7. The future

Maxwell now exists, a 32-way IBM Bladecentre containing 64 Xilinx Virtex-4s configured in two flavours. Initial performance results show that it has not suffered too badly from its 'general purpose' nature, and indeed suggest that 'general purpose FPGA computer' is at least not an oxymoron!

We intend to explore ways to improve the programmability of Maxwell and FPGA-based systems in general. While the three demonstrators here show that FPGAs *can* deliver genuine performance benefits even for memory-bound HPC simulation codes, the hardware accelerations did not write themselves, and the costs were high.

Realising significant benefit does still require collaboration between software and hardware engineers. The PTK has been a useful development in providing high-level vendor-neutral application interfaces and common methods of configuration and job launching but its standardizing approach does not go deep enough into the software stack.

The HPRC community needs now to turn its attention to standards at all levels to help cement FPGAs into the new fabric of high-performance computing.

8. References

[1] R. Baxter et al, "High-Performance Reconfigurable Computing – the View from Edinburgh", *Proc. AHS2007 Conf., Second NASA/ESA Conference on Adaptive Hardware and Systems*, Edinburgh, 2007.

[2] The FHPCA, www.fhpca.org

[3] R. Saas et al, "Reconfigurable Computing Cluster (RCC) Project: Investigating the Feasibility of FPGA-Based Petascale Computing", *15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'07)*
[4] D. Bennett et al, "An FPGA-oriented target language for HLL compilation", RSSI 2006.
[5] Virtex-4 datasheets, http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex4/index.htm, Xilinx Inc, May 2007.
[6] H100 Series datasheet, http://www.nallatech.com/mediaLibrary/images/english/5595.pdf, Nallatech ltd, May 2007.
[7] ADM-XRC-4FX datasheet, http://www.alphadata.co.uk/adm-xrc-4fx.html, Alpha Data ltd, May 2007.
[8] R. Baxter et al, "The FPGA HPC Alliance Parallel Toolkit", *Proc. AHS2007 Conf., Second NASA/ESA Conference on Adaptive Hardware and Systems*, Edinburgh, 2007.
[9] F. Black & M. Scholes, "The Pricing of Options and Corporate Liabilities", *Journal of Political Economy*, Vol. 81, pp. 637-654.
[10] Dimensional Imaging, http://www.di3d.com/
[11] OHM plc, http://www.ohmsurveys.com/
[12] L. MacGregor, D. Andreis, J. Tomlinson & N. Barker, "Controlled-source electromagnetic imaging on the Nuggets-1 reservoir", *The Leading Edge*; August 2006; v. 25; no. 8; pp. 984-992.
[13] MPI manual, Argonne National Lab, http://www-unix.mcs.anl.gov/mpi/www/www3/MPI_Wtime.html