Maxwell – a 64 FPGA Supercomputer

Rob Baxter, 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*

***Abstract*—We describe the FPGA-based supercomputer *Maxwell* built by the FPGA High-Performance Computing Alliance at the University of Edinburgh. Winner of the silver medal in the BT Flagship Award for Innovation at the 2007 British Computer Society Awards, *Maxwell* is a general-purpose 64 FPGA computer designed for high-performance parallel computing. This paper describes the machine itself, its hardware and software environment and presents benchmark results from runs of three commercial demonstration applications from the oil, medical and finance sectors.**

***Index Terms*—FPGAs, High-Performance Reconfigurable Computing**

I. Introduction

Field-programmable gate arrays (FPGAs) are not new, at least not by today's measure of technology novelty. Ross Freeman, co-founder of Xilinx Inc., invented this new form of programmable logic device in 1984. While rapidly becoming a key piece of technology in the embedded systems area it took really until the beginning of the 21st century for FPGAs to be viewed as something more – a potential contender for the place of numerical workhorse in high-performance computing (HPC) systems. In 2002 Compton and Hauck concluded that "Reconfigurable computing is becoming an important part of research in computer architectures and software systems" [1].

Since then high-performance reconfigurable computing (HPRC) has developed both a name and an ecosystem of researchers and computer vendors pushing the envelope of what it is possible to make programmable logic do in the context of HPC applications. Baxter *et al* [2] and Craven & Athanas [3] provide good analyses of the current and potential state of FPGA-based supercomputing.

Most recently a number of groups have begun to examine the potential of large-scale FPGA-based systems where the FPGAs are used as main processors rather than the more traditional co-processor. Cathey *et al* [4] describe in excellent detail the design choices and tradeoffs involved in building such a large-scale system – a system with many similarities to the one described here – while Sass *et al* take the possibilities a step further and consider a purely FPGA design involving no host CPUs at all [5].

Manuscript received February 15, 2008. This work was supported in part by Scottish Enterprise.

R.M. Baxter is with EPCC at the University of Edinburgh, Edinburgh, UK (phone: +44 131 651 3579; fax: +44 131 650 6555; e-mail: r.baxter@epcc.ed.ac.uk).

Against this background of potential in the emerging area of high-performance reconfigurable computing the FPGA High Performance Computing Alliance (FHPCA [6]) 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 HPC applications.

The machine itself – Maxwell – was completed in the first part of 2007 and subsequently went on to win the silver medal in the BT Flagship Award for Innovation at the 2007 British Computer Society Project Excellence Awards [7].

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

II. 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 [8] 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.

III. 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.

A. 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.

B. 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 [9]. 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.

C. Nallatech H101

The cards in one half of Maxwell are a slight variant on Nallatech's off-the-shelf H101-PCIXM [10]. 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 RocketIO 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.

D. Alpha Data ADM-XRC-4FX

The other 16 blades in Maxwell host 32 Alpha Data ADM-XRC-4FX cards [11]. 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.

E. 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.

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

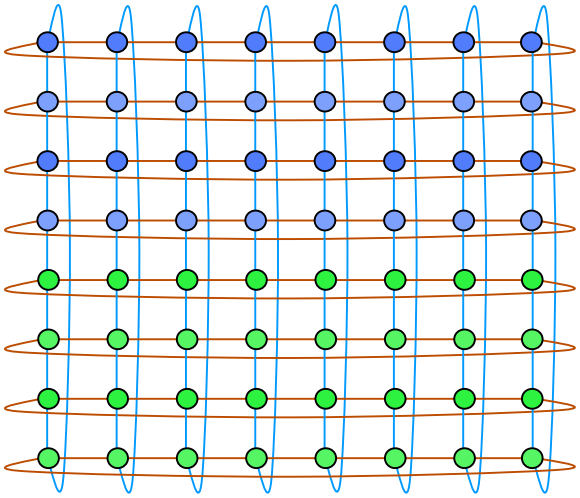

Figure 1. FPGA connectivity in Maxwell

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 IV below).

IV. 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 find it difficult to gain footholds in the HPC market.

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.

A. A Parallel Toolkit for HPRC

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.

Maintaining the portability and maintainability of HPC codes on new architectures is these days essential. Many years of effort have been invested since the early 1990s on standardizing HPC codes to run cleanly and portably across a wide range of parallel hardware. Standards such as the Message Passing Interface [12] and OpenMP [13] are used almost exclusively; gone are the days of vendor-specific codes and proprietary languages.

High-level FPGA programming faces the same challenges now as parallel computing did 15 years ago. Addressing these challenges requires a consensus among tool providers and a standardization activity by the users of high-performance reconfigurable computing – a process organizations like OpenFPGA [14] hope to catalyse. Indeed the PTK was developed contemporaneously with the OpenFPGAs GenAPI [15] and shares many of the same design goals.

The PTK is a potential step along the road towards standardizing both high-level APIs and job and machine configuration methods for HPRC. It is a set of practices and infrastructure intended to address key HPRC acceleration issues such as: how to associate processes with FPGA resources; how to associate FPGAs with bitstreams; how to manage contention for FPGA resources within a process; and how to managing code dependencies to facilitate re-use. To these ends the PTK comprises a library of C++ classes providing abstract interfaces to FPGA hardware components; classes providing standard ways to configure arbitrary FPGA hardware; and a standard way of launching parallel FPGA jobs.

The PTK is described in more detail in [16].

B. 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.

V. 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.

A. 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-Scholes model of prices [17], 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 application 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.

B. 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 [18].

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.

C. 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 [19]. 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 [20].

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.

VI. PERFORMANCE RESULTS

This section presents our final results from benchmark tests of the three demonstrator applications on Maxwell. 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.

A. 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.

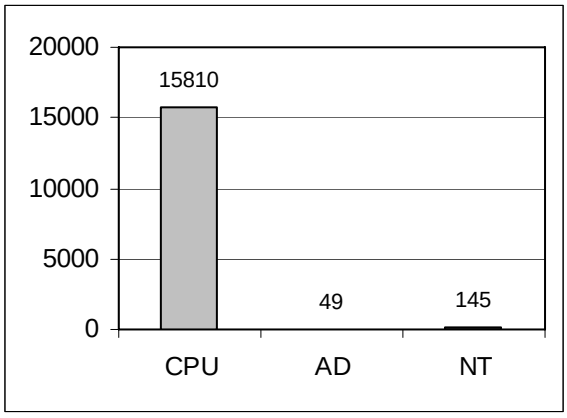

Figure 2. Single-node MCopt performance (s).

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.

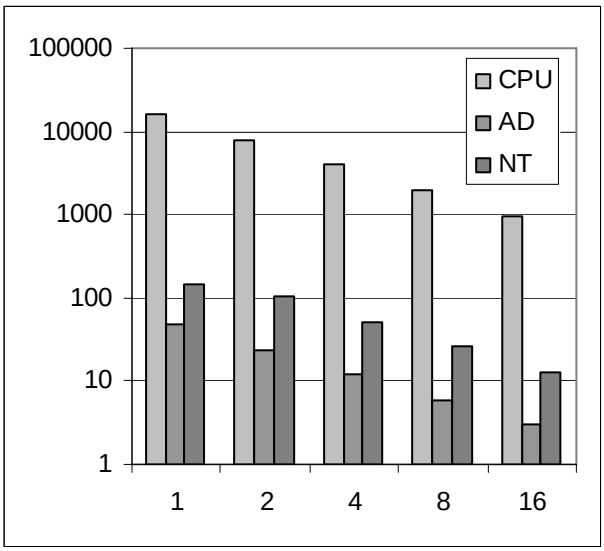

Figure 3. MCopt scaling performance on Maxwell (times in s). Note the scale is logarithmic.

B. 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 a networked disk, so this is also an interesting test of the network and i/o overheads of the parallel system.

Figure 4 shows the performance for these tests running on the Nallatech side of the machine. The tests were run on two nodes each (two CPUs versus two FPGAs) due to the memory requirements of the initialization phase

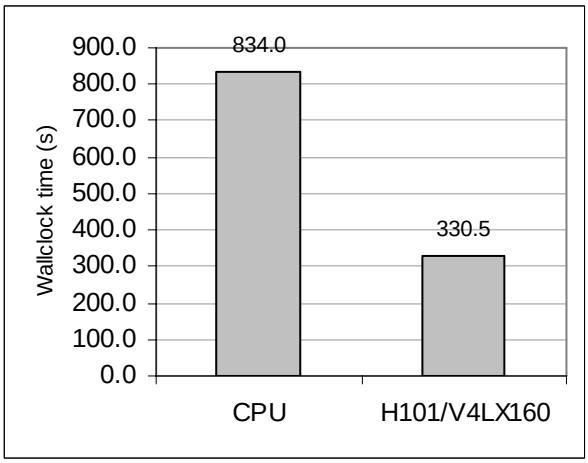

Figure 4. Two-node DI3D facial imaging performance (times in s), comparison between software and Nallatech hardware versions.

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() [21] across the full batch-processing version of the application. This includes software-only components

not accelerated in hardware.

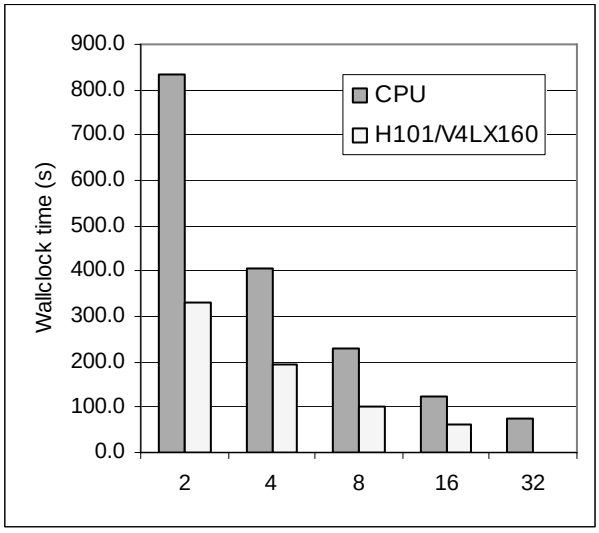

Figure 5. Facial imaging scaling performance on Maxwell (times in s), comparison of software and Nallatech H101 hardware versions.

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 software and hardware versions – we suspect that this is a feature of i/o bandwidth saturation across the machine although we have not investigated in detail. This may also be the root cause of the seeming superlinear speedup between 2 and 4 CPUs – recall that Maxwell's host blade cluster relies on diskless blades and all disk i/o must share bandwidth with any inter-process communication, thus potentially adding complexity to the comms patterns used in the parallel computing model of "master processor reads file and distributes." We have not been able to verify this conjecture, though.

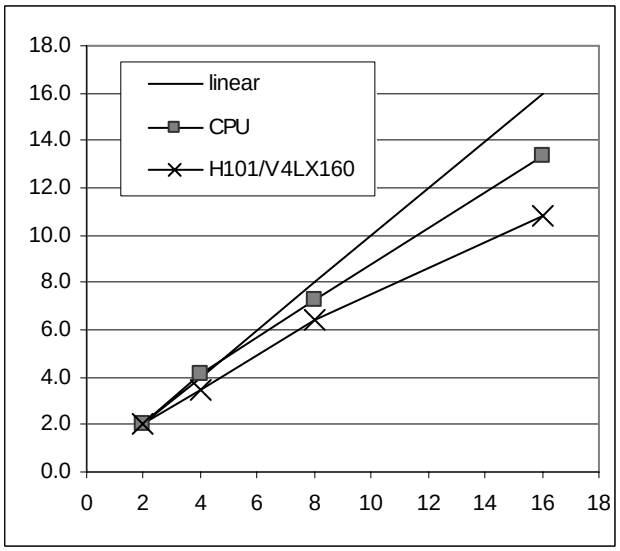

Figure 6. Parallel speedup of the DI3D facial imaging code on Maxwell, comparison of software and Nallatech H101 hardware versions.

C. 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).

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. These results suggests the FPGA version runs some 4.8 times faster per node than the pure software version.

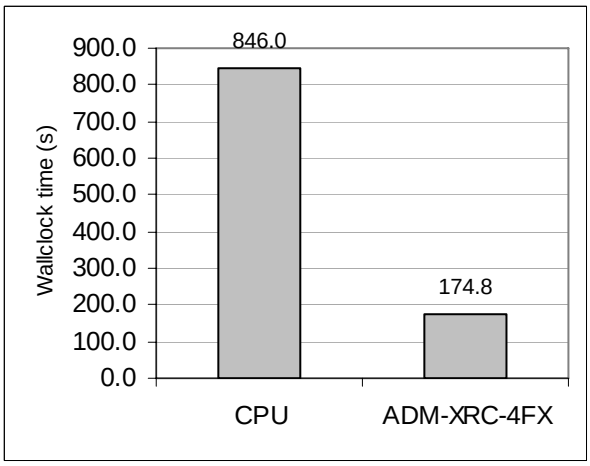

Figure 7. OHM3D 8-node performance in seconds, comparison of software and Alpha Data ADM-XRC-4FX hardware versions.

The key test of any current or future HPRC system will be its parallel scalability for non-trivially parallel codes such as OHM3D. Figures 8 and 9 plot the scaling characteristics of the OHM3D code running in pure software and on Alpha Data hardware.

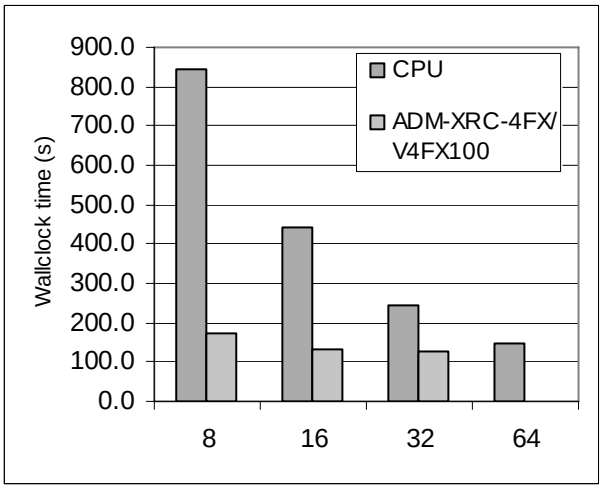

Figure 8. OHM3D scaling performance on Maxwell (times in s), comparison of software and Alpha Data ADM-XRC-4FX hardware versions

The scaling results for the FPGA version are disappointing

but are understood as a feature of the way boundary data exchange between domains had to be implemented. Timing constraints across the Virtex-4 proved extremely difficult to resolve for the RocketIO data exchange protocol; eventually a solution was implemented that uses MPI_Barrier synchronization calls at the software level between different dimensions of the boundary data exchange. Thus, in effect, the FPGA implementation can only scale in one dimension of the problem size, while the original software version scales in both physical dimensions.

This difficulty in achieving full two-dimensional scaling demonstrates clearly the complexities and challenges faced by full parallel HPRC systems.

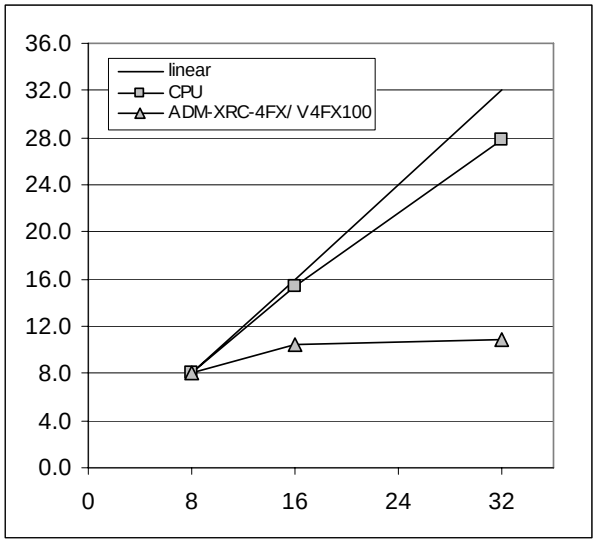

Figure 9. Parallel speedup of the OHM3D CSEM code on Maxwell, comparison of software and Alpha Data ADM-XRC-4FX hardware versions. This graph is rebased to linear at 8 nodes.

D. *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 14 staff-months to conclude only a partially-parallelised version.

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.

VII. The Future

Maxwell now exists as a world-leading HPRC research paltform, 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.

VIII. References

[1] K. Compton and S. Hauck, "Reconfigurable computing: a survey of systems and software," ACM Computing Surveys, vol. 34, no. 2, pp. 171–210, 2002.

[2] 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.

[3] S. Craven, P. Athanas, "Examining the viability of FPGA supercomputing", *EURASIP Journal on Embedded Systems*, v.2007 n.1, p.13-13, January 2007 .

[4] C. L. Cathey, J. D. Bakos, and D. A. Buell. "A reconfigurable distributed computing fabric exploiting multilevel parallelism". In J. Arnold and D. A. Buell, editors, *Proceedings of 14th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM'06)*, pages 121–130, Napa, CA, Apr. 2006.

[5] R. Sass 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)*

[6] The FHPCA, www.fhpca.org

[7] See http://www.bcs.org/server.php?show=conWebDoc.16275

[8] D. Bennett et al, "An FPGA-oriented target language for HLL compilation", RSSI 2006.

[9] Virtex-4 datasheets, http://www.xilinx.com/products/silicon_solutions/fpgas/virtex/virtex4/index.htm, Xilinx Inc, May 2007.

[10] H100 Series datasheet, http://www.nallatech.com/mediaLibrary/images/english/5595.pdf, Nallatech ltd, May 2007.

[11] ADM-XRC-4FX datasheet, http://www.alphadata.co.uk/adm-xrc-4fx.html, Alpha Data ltd, May 2007.
[12] The Message Passing Interface forum, www.mpi-forum.org
[13] The OpenMP Architecture Review Board, www.openmp.org
[14] OpenFPGA, www.openfpga.org.
[15] Current OpenFPGA GenAPI work-in-progress, https://isl.ncsa.uiuc.edu/twiki/bin/view/OpenFPGA/GenAPI
[16] R. Baxter et al, "The FPGA HPC Alliance Parallel Toolkit", *Proc. AHS2007 Conf., Second NASA/ESA Conference on Adaptive Hardware and Systems*, Edinburgh, 2007.
[17] F. Black & M. Scholes, "The Pricing of Options and Corporate Liabilities", *Journal of Political Economy*, Vol. 81, pp. 637-654.
[18] Dimensional Imaging, http://www.di3d.com/
[19] OHM plc, http://www.ohmsurveys.com/
[20] 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.
[21] MPI manual, Argonne National Lab, http://www-unix.mcs.anl.gov/mpi/www/www3/MPI_Wtime.html