

Introduction

Applications that push the limits of computer technology come from a variety of fields, including financial analytics, data mining, medical imaging, and scientific computation, each with widely diverse program requirements. What these high-performance computing applications all have in common, however, is a critical need for the fastest possible speeds of program execution.

The last few years have seen a move towards cluster computing as an answer to rising performance demands. Now, however, cluster systems based solely on conventional processors are running out of room to grow. Adding field-programmable gate array (FPGA) coprocessors to these systems can boost application performance while simultaneously reducing power consumption and the total cost of ownership.

Trends in High Performance Computing

The term High Performance Computing (HPC) was originally used to describe powerful, number-crunching supercomputers. As the range of applications for HPC has grown, however, the definition has evolved to include systems with any combination of accelerated computing capacity, superior data throughput, and the ability to aggregate substantial distributed computing power. The architectures of HPC systems have also evolved over time, as illustrated in Figure 1. Ten years ago, symmetric multiprocessing (SMP) and massively parallel processing (MPP) systems were the most common architectures for high performance computing. More recently, however, the popularity of these architectures has decreased with the emergence of a more cost-effective approach: cluster computing. According to the Top500[®] Supercomputer Sites project, the cluster architecture is now the most commonly used by the world's highest performing computer systems.

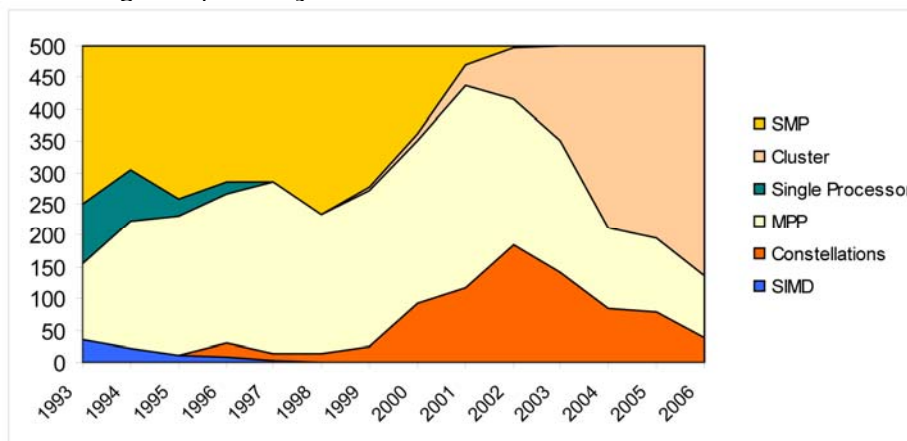


Figure 1. The history of HPC architectures shows a shift toward cluster computing.

Source: Top500.org

Cluster computing has achieved this prominence because it is extremely cost-effective. Rather than relying on the custom processor elements and proprietary data pathways of SMP and MPP architectures, cluster computing employs commodity standard

processors, such as those from Intel and AMD, and uses industry-standard interconnects such as Gigabit Ethernet and InfiniBand®.

Applications that are a good fit for the cluster architecture tend to be those that can be “parallelized,” or broken into sections that can be independently handled by one or more program threads running in parallel on multiple processors. Such applications are widespread, appearing in industries such as finance, engineering, bioinformatics, and oil and gas exploration. Application software for these industries is now routinely being delivered in “cluster aware” forms that can take advantage of HPC architectures.

Clusters are becoming the preferred HPC architecture because of their cost effectiveness; however, these systems are starting to face challenges. The single-core processors used in these systems are becoming denser and faster, but they are running into memory bottlenecks and dissipating ever-increasing amounts of power. The presence of memory bottlenecks has two components: a limited number of I/O pins on the processor package that can be dedicated for memory access and an increasing latency due to the use of multi-layered memory caches. Higher power dissipation is a direct result of increasing clock speeds, and is forcing a need for extensive cooling of the processors.

The increasing power demand of processors is of particular concern in data centers and many other applications where calculations per watt are increasingly important. System cooling requirements are limiting cluster sizes, and therefore limit performance levels achievable. Their existing cooling systems are simply running out of capacity and increasing cooling capacity carries a high price.

The industry’s current solution to these growing problems is to move towards multicore processors. Increasing the number of processor cores in a single package offers increased node performance at somewhat lower power dissipation than that of an equivalent number of single-core processors. But the multicore approach does not address the memory bottlenecks inherent in the packaging.

An alternative approach has arisen: using FPGA-based coprocessors to accelerate the execution of key steps in the application software. The approach is similar to the idea of coding an inner loop of a C++ application in assembly language to increase overall execution speed.

FPGAs typically run at much slower clock speeds than the latest CPUs, yet they can more than make up for this with their superior memory bandwidth, high degree of parallelization, and the customization that is possible. An FPGA coprocessor programmed to hardware-execute key application tasks can typically provide a 2X to 3X system performance boost while simultaneously reducing power requirements 40% as compared to adding a second processor or even a second core. Fine tuning of the FPGA to the application’s needs can achieve performance increases greater than 10X.

FPGA Acceleration Opportunities in HPC

Any solution to boosting HPC applications must cover a wide spectrum, with widely differing computational needs. Business data mining and genome sequencing, for example, process large amounts of structured data using string-matching, logical testing and numerical aggregation operations. On the other hand, medical imaging, scientific visualization, and computational chemistry involve operations like coordinate mapping, mathematical transformations, and filtering. Each of these applications demands a particular combination of mathematical and logical operators, coupled with efficient memory access.

It is difficult for general-purpose CPUs or specialized processor solutions such as graphics processing units (GPUs) or network processors to provide an optimal solution for the broad spectrum of HPC applications. The FPGA, however, is a re-configurable engine. It can be optimized under software control to meet the particular requirements of each HPC application. This allows one hardware solution to address many HPC applications with equal efficiency.

FPGAs accelerate HPC applications by their ability to exploit the parallelism inherent in the algorithms employed. There are several levels of parallelism to address. A good starting point is to structure the HPC application for multi-threaded execution suitable for parallel execution across a grid of processors. This is task-level parallelism, exploited by cluster computing. There are software packages available that can take legacy applications and transform them into a structure suitable for parallel execution.

A second level of parallelism lies at the instruction level. Conventional processors support the simultaneous execution of a limited number of instructions. FPGAs offer much deeper pipelining, and therefore can support a much larger number of simultaneously executing "in-flight" instructions.

Data parallelism is a third level that FPGAs can readily exploit. The devices have a fine-grained architecture designed for parallel execution. Thus, they can be configured to perform a set of operations on a large number of data sets simultaneously. This parallel execution performs the equivalent work of numerous conventional processors all in a single device.

By exploiting all three levels of parallelism, an FPGA operating at 200 MHz can outperform a 3 GHz processor by an order of magnitude or more, while requiring only a quarter of the power. In bioinformatics, for instance, FPGAs have demonstrated 50X – 1000X acceleration over conventional processors in running sequence alignment algorithms for DNA sequences [1]. Medical imaging algorithms for 2D and 3D computed tomography (CT) image reconstructions have routinely shown at least a 10X acceleration. Commonly used signal processing algorithms, such as FFTs (Fast Fourier Transforms) show performance factors of 10X over the fastest CPUs.

Opportunities and Requirements in Financial Analytics

One of the major markets where computing speed is an extremely important asset is financial analytics. A key application within this market space is the analysis of "derivatives": financial instruments such as options, futures, forwards, and interest-rate swaps. Derivatives analysis is a critical on-going activity for financial institutions, allowing them to manage pricing, risk hedging, and the identification of arbitrage opportunities. The worldwide derivatives market has tripled in size in the last five years, as show in Table 1 [2].

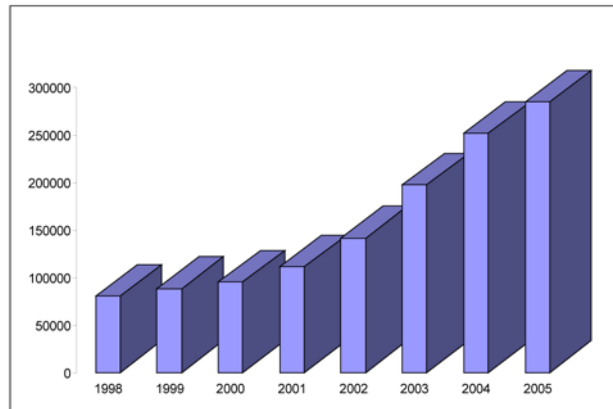


Table 1. Worldwide Derivatives Market, (notional amounts in billions of \$)

The numerical method for derivatives analysis uses Monte Carlo simulations in a Black-Scholes world. **The algorithm makes heavy usage of floating-point math operations such as logarithm, exponent, square-root, and division. In addition, these computations must be repeated over millions of iterations.**

The numerical Black-Scholes solution is typically used within a Monte Carlo simulation: where the value of a derivative is estimated by computing the expected value, or average, of the values from a large number different scenarios, each represent a different market condition.

The key point here is the need for “a large number”. Because Monte Carlo simulation is based on the generation of a finite number of realizations using a series of random numbers (to model the movement of key market variables), the value of an option derived in this way will vary each time the simulations are run. Roughly speaking, the error between the Monte Carlo estimate and the correct option price is of the order of the inverse square root of the number of simulations. **To improve the accuracy by a factor of 10, 100 times as many simulations must be performed [3].**

With so many iterations of intense computation required, derivatives analysis is clearly a prime candidate for acceleration of HPC performance. Performance is not the only concern for this application, however. The ideal solution must also address some critical operational issues.

Accurate price estimates are critical for financial institutions; inaccurate or compromised models create arbitrage opportunities for other players in the market. The algorithms and parameters used by the analysts thus can vary widely for different financial instruments (and portfolios of instruments) and are constantly being tweaked and refined. For this reason and for reasons of maintainability, financial analysts (“quants”) typically develop their algorithms in a high-level language, such as C, Java, or Matlab.

Because the accuracy of the analysis represents an edge in the market, a high degree of secrecy shrouds the exact algorithms employed by the quants. Disclosure of the algorithm details could expose billions of dollars to arbitrage risk. In addition, there are regulatory (SEC) requirements for verification and validation of risk-return claims made on financial instruments. **It is therefore often not practical or advisable to modify, transform, re-factor, or optimize the application codes in order to speed its execution.**

To summarize, the requirements of the financial analytics community are very stringent: high-precision, intense math computation, millions of iterations, programmed in a high-level language that should not be re-factored or altered. Can FPGA coprocessors meet these challenges? The answer is “YES”, but only if a complete solution is provided. An FPGA hardware platform, a high-level programming environment, and a library of key FPGA functions are the keys.

One example of a solution is the XtremeData-Impulse C environment. Impulse C provides the high-language (ANSI C), which is coupled with the XtremeData Platform Support Package (PSP) for a seamless path to FPGA implementation. XtremeData also provides a full double-precision, IEEE-754 compliant math library that includes all the operators necessary for Black-Scholes-type of simulations. (This is the first environment to be supported and released by XtremeData. In the near future, we plan to support other high-level language compilers, and Matlab-oriented approaches, such as Star-P from Interactive Supercomputing).

The XtremeData x86 FPGA Coprocessor

XtremeData offers an FPGA coprocessor, the XD1000 which is socket-compatible with an AMD Opteron CPU. This configuration allows the XD1000 to plug into standard commercial HPC enterprise, rack, and blade servers by replacing one of the CPUs on a motherboard, as shown in Figure 2.

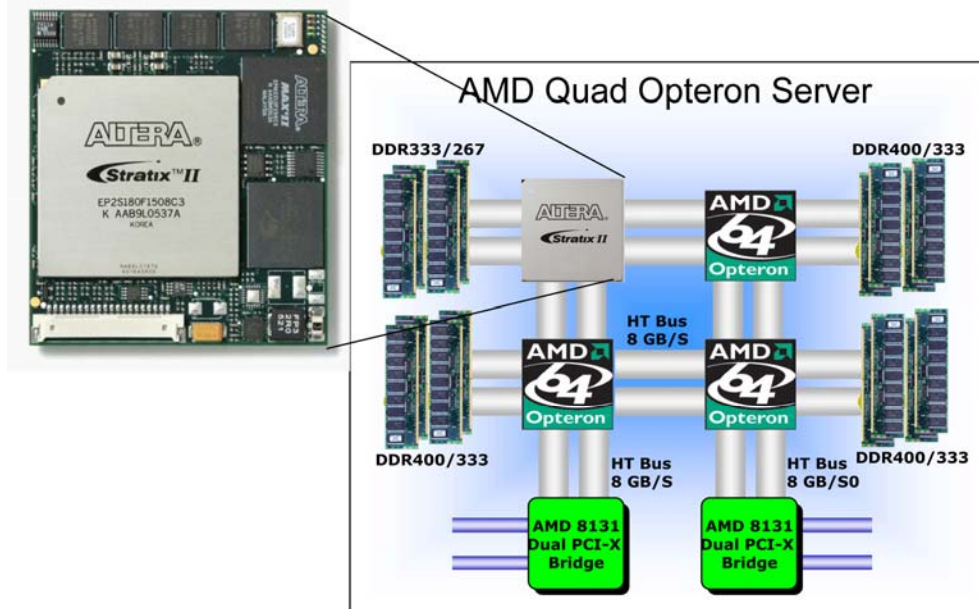


Figure 2. The XD1000 is an FPGA-based coprocessor that plugs directly into an AMD Opteron CPU socket, providing a simple hardware integration path.

This combination of an AMD Opteron and FPGA coprocessor is attractive for many reasons. The Opteron offers a high-bandwidth, low-latency interface to other processors via the open-standard HyperTransport™ (HT) interconnect. It also includes an on-chip memory controller that interfaces to memory on the motherboard. Placing the FPGA module in one of the Opteron sockets thus gives the coprocessor a high-speed link to memory as well as a fast connection to the host processor, without requiring any board modifications.

Adding the XD1000 module to an Opteron motherboard represents a minimal-impact upgrade to an HPC system. Thus, the FPGA is able to seamlessly integrate into cluster environments and scale up in keeping with computational requirements.

Impulse C Compiler and XDI Support Libraries

An FPGA hardware coprocessor needs to be programmed in order to accelerate the application software. The traditional world of HPC application software developers, however, is unlikely to include hardware-oriented programming tools such as VHDL and Verilog. Therefore, for broad acceptance, a software-oriented development flow is mandatory.

One such design flow is provided by Accelerated Technologies via its Impulse C compiler as shown in Figure 3.

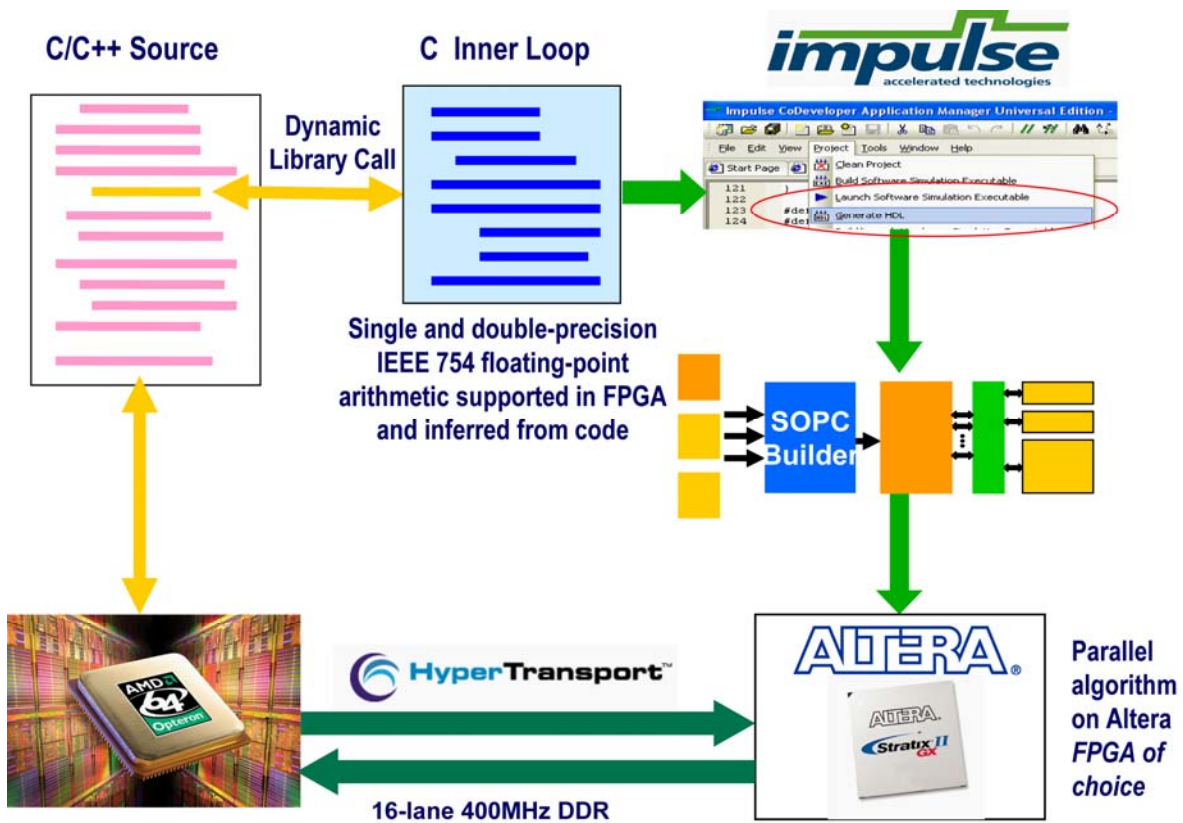


Figure 3. The Impulse C compiler along with the XtremeData platform support package (PSP) allows developers to accelerate application code at a high level of abstraction.

Impulse C uses the standard ANSI C language with extensions that provide an easy-to-use API for FPGA-specific functions. To further orient this tool toward a software-oriented programming experience, Impulse C and XtremeData have collaborated to produce a Platform Support Package (PSP) that enables the use of Impulse C in mixed software/hardware applications targeting the XD1000 FPGA coprocessor. XtremeData's libraries integrated into the PSP provide support for IEEE 754 single- and double-precision floating-point arithmetic automatically inferring the proper FPGA logic for the XD1000 when variables of the type *float* or *double* are supplied to an arithmetic operator.

Using the XD1000 PSP, software application programmers can create an FPGA accelerator description written entirely in Impulse C, eliminating the need to write low-level VHDL or Verilog code. To create an application accelerator using Impulse C and the XD1000 Coprocessor Module, simply:

- Identify the performance-critical portion of the application to be accelerated
- Describe the performance-critical portion of the algorithm in Impulse C, targeting the XD1000 Coprocessor Module
- Compile the Impulse C code to VHDL
- Export the design to the XD1000 Coprocessor Module
- Execute FPGA place-and-route by executing a batch file produced during the export
- Compile the application software by executing make using the makefile produced during the export
- Download the bitfile produced by the place-and-route to the FPGA Coprocessor, and
- Execute the accelerated application

The result of this procedure is a complete application accelerator, including all FPGA logic, memory interfaces, FPGA-to-CPU (HyperTransport) communication routines, and application software. This is a completely push-button flow from the C-level description to the FPGA programming file, producing a complete Altera® Quartus® II project and a single script that when invoked performs synthesis, placement, and routing of the FPGA, and generates a binary file suitable for downloading to the FPGA on the XD1000 Coprocessor Module.

Case Study: Monte Carlo Black-Scholes Simulation

The Black-Scholes world refers to seminal work done by Fisher Black and Myron Scholes in the late 60's and early 70's. In a paper published in 1973, they modeled the behavior of the price of an asset (share price of a traded stock, for instance) as a stochastic process, which could be described by a stochastic differential equation. The now famous "Black-Scholes" equation is a linear parabolic partial differential equation:

$$\frac{\partial V}{\partial t} + \frac{1}{2} \sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS \frac{\partial V}{\partial S} - rV = 0$$

Equations of this form are well known in thermo-dynamics as "heat" or "diffusion" equations. When applied to derivatives, the notations in the equation are:

- V : value (price) of derivative
- t : the time variable
- S : the price of the underlying asset (share price)
- r : interest rate
- σ : measure of the volatility of the price movement

The terms of the equation can be understood through an analogy to a physical process. The equation can be accurately interpreted as a reaction-convection-diffusion process. The first two terms define a diffusion, the third term a convection and the fourth a reaction process. An example would be the dispersion (diffusion) of a pollutant along a flowing river (convection), with some absorption by the sand (reaction) [4].

Prevalent numerical solutions to the Black-Scholes equation are of the form shown below for the price of call option ("European" option) [5].

$$c = Se^{-\delta(T-t)} N[d_1] - Ke^{-r(T-t)} N[d_2]$$

Where:

$$d_1 = \frac{\ln \frac{S}{K} + (r - \delta + \frac{1}{2} \sigma^2)(T - t)}{\sigma \sqrt{T - t}}$$

$$d_2 = d_1 - \sigma \sqrt{T - t}$$

$N[x]$ = the standard cumulative normal distribution function evaluated at x .

And the notations used are: S = asset price, K = option strike price, T = time to maturity, t =time variable, δ =dividend on asset, σ =volatility of S .

As can be seen, the computation demands of the numerical solution to the Black-Scholes equation are significant. Computationally expensive operations such as logarithm, exponent, square-root, and divide are used repeatedly.

Analysis shows that the necessary computations are:

Operation	# of times used
Add / Subtract	9
Multiply	30
Divide	6
Exponent	3
Logarithm	1
SquareRoot	2

XtremeData has run a benchmark to evaluate the performance of Monte Carlo Black-Scholes simulations both with and without an FPGA accelerator. Using the XtremeData-Impulse C development environment to extract the computationally-expensive operations from the code and implement them in the XD1000 FPGA showed **that the FPGA performs 20X faster than a 3 GHz x86, 64-bit CPU. Both FPGA and CPU implementations were IEEE-compliant floating-point and the source remained ANSI C-code.**

Future Directions

The results of the benchmark show what is achievable while maintaining the application code as a high-level language program. If it is possible to relax this constraint, higher performance is possible. Performance factors of ~200X have been reported by some vendors. However, these performance levels reflect the relaxed conditions of re-factoring and re-writing parts of the application in a Hardware Description Language (HDL) and not maintaining IEEE-compliant floating point for the computations.

At the extreme end of the spectrum, it should be possible to design a custom application specific integrated circuit (ASIC) that could achieve ~500X or greater performance factors. This, of course, comes with an extreme cost and lack of design

flexibility. A custom ASIC is unlikely to be adaptable to changes in the algorithm, and development costs are high.

XtremeData's goal is to achieve performance factors of around ~50X, while still satisfying all the flexibility and high-level language requirements of the market, as shown in Figure 4.

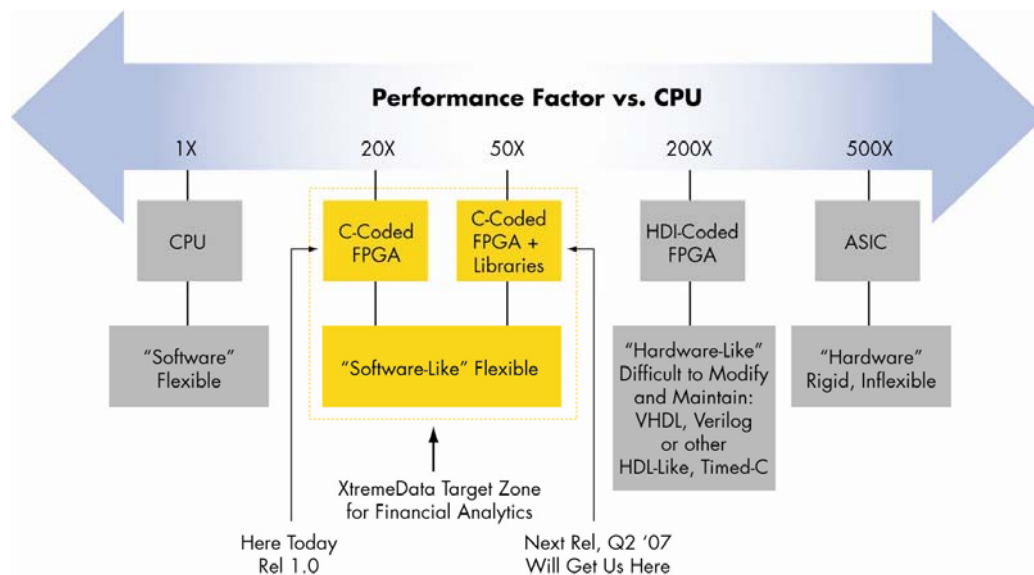


Figure 4. HPC acceleration concepts involve tradeoffs between flexibility and speed. XtremeData targets the ideal balance for financial analytics.

The use of Impulse C is the first environment to be supported and released by XtremeData. In the near future, we plan to support other high-level language compilers, and Matlab-oriented approaches, such as Star-P from Interactive Supercomputing.

The XtremeData roadmap also includes a next generation release of the development environment (Rel 2.0) in the Q2 2007 timeframe, where a library of key math functions will be integrated into the compiler. This library will include functions including random number generators that meet many different probability distribution characteristics and a set of linear algebra functions which are heavily used in Monte Carlo simulations. The next release will also seamlessly support the dynamic re-load of the FPGA, as required during the computation flow. This feature will enable the usage of a large library of FPGA functions, all of which will be too large to simultaneously fit within one FPGA. Dynamic re-load will permit the developer to treat FPGA functions in the same manner as functions within a software library.

Conclusion

Today's high performance computing systems based on the cluster computing architecture are falling short of delivering the performance needed in demanding applications. Using an FPGA accelerator module such as the XtremeData XD1000 can boost system performance with fewer nodes, reducing the system's overall power demands.

The financial analytics market space has intense computing requirements, especially in the area of derivatives analysis. In this application, there is a heavy usage of floating-point math operations such as logarithm, exponent, square-root, and divide. These

computations are also required to be repeated over millions of iterations. Additionally, it is also a requirement to easily refine the algorithm; therefore the source must be maintained in a high-level language without re-factoring or re-writing.

XtremeData offers an FPGA-acceleration solution to this market that satisfies these stringent requirements. XtremeData, along with key partners such as Altera Corporation, has outlined a roadmap for future releases of the solution that will add key features to enable even further acceleration, while still adhering to the market requirements.

In this context, XtremeData is currently engaging with interested parties who can provide feedback and validation of the platform and its planned evolution. We welcome your input. Please contact us via either of the following contacts:

Altera Corporation

Bryce Mackin

Manager, Strategic Marketing

(408) 544-7000

bmackin@altera.com

XtremeData, Inc.

Nathan Woods, Ph.D.

Principal Scientist

(847) 483-8723

nathan@xtremedatainc.com

References

- [1] Global Smith-Waterman algorithm implemented on XtremeData platform by University of Delaware researchers, P. Zhang, G. Tan and D. Kreps
- [2] Source: Bank for International Settlements,
<http://www.bis.org/statistics/derstats.htm>
- [3] From *"Derivatives: The Theory and Practice of Financial Engineering"*, by Paul Wilmott, 1998, John Wiley & Sons
- [4] From *"Derivatives: The Theory and Practice of Financial Engineering"*, by Paul Wilmott, 1998, John Wiley & Sons
- [5] From *"Implementing Derivatives Models"*, by Les Clewlow and Chris Strickland, 1991, John Wiley & Sons