WP375 (v1.0) September 10, 2010

High Performance Computing Using FPGAs

By: Prasanna Sundararajan

For over a quarter of a century, Xilinx® FPGAs continue to be the platform of choice for designing programmable systems. Due to their inherent flexibility, Xilinx FPGAs have been used in programmable solutions such as serving as a prototype vehicle as well as being a highly flexible alternative to application-specific integrated circuits.

Advancements in silicon, software, and IP have proven Xilinx FPGAs to be the ideal solution for accelerating applications on high-performance embedded computers and servers.

This white paper describes the various use models for applying FPGAs in High Performance Computing (HPC) systems.

Introduction

Until the early 2000s, general purpose single-core CPU-based systems were the processing systems of choice for HPC applications. They replaced exotic supercomputing architectures because they were inexpensive, and performance scaled with frequency in line with Moore's Law. Presently, the HPC industry is going through another historical step change. General-purpose CPU vendors changed course in the mid-2000s to rely on multicore architectures to meet high-performance demands. The technique of simply scaling a single-core processor's frequency for increased performance has run its course, because as frequency increases, power dissipation escalates to impractical levels.

The shift to multicore CPUs forces application developers to adopt a parallel programming model to exploit CPU performance. Even using the newest multicore architectures, it is unclear whether the performance growth expected by the HPC end user can be delivered, especially when running the most data- and compute- intensive applications. CPU-based systems augmented with hardware accelerators as co-processors are emerging as an alternative to CPU-only systems. This has opened up opportunities for accelerators like Graphics Processing Units (GPUs), FPGAs, and other accelerator technologies to advance HPC to previously unattainable performance levels.

High Performance Computing: System Types

Systems in the HPC market span a spectrum of system types. They range from massive compute server farms to computers embedded inside equipment. For discussion purposes, these systems are classified into two different groups:

- High-performance servers
- High-performance embedded computers

High-Performance Servers

High-performance servers comprise a class of systems typically used by scientists, engineers, and analysts to simulate and model applications and analyze large quantities of data. Typical systems range from server farms to big supercomputers. A summary of different industries and applications are provided in Table 1.

Table 1: **Typical Applications for High Performance Servers**

Industry	Sample Applications
Government labs	Climate modeling, nuclear waste simulation, warfare modeling, disease modeling and research, and aircraft and spacecraft modeling
Defense	Video, audio, and data mining and analysis for threat monitoring, pattern matching, and image analysis for target recognition
Financial services	Options valuation and risk analysis of assets
Geosciences and engineering	Seismic modeling and analysis, and reservoir simulation
Life sciences	Gene encoding and matching, and drug modeling and discovery

These applications are compute and data intensive and are in constant need of increased compute power and bandwidth to memory. With higher compute power, algorithms of greater complexity can be employed to produce more accurate results.

For instance, to identify possible threat activities, better algorithms can be employed in the analysis of internet voice and data captures to match "patterns of interest." In the financial community, more accurate modeling-based Monte Carlo simulations can be employed for better risk analysis of assets and investments.

High-Performance Embedded Computers

High-performance embedded computers are computers that are incorporated in equipment or an appliance to perform specific compute- and data-intensive tasks. A summary of different industries and applications for embedded computers in certain equipment is provided in Table 2.

Table 2: **Typical Applications for High-Performance Embedded Computers**

Industry	Application and Equipment
Defense	Beam forming in radar
Airborne Electronics	Image compression and analysis in payload
Communications	Encryption in network routers
Medical Imaging	Image rendering in CT and MRI scanners
Financial Services	Low latency and high throughput data processing in trading solutions

In the past, these systems were typically based on proprietary boards using custom integrated circuits and were designed to handle high data processing, I/O, and memory rates to meet performance requirements. A growing trend in the industry is to adopt commercial off-the-shelf (COTS) plug-in cards and server platforms for these applications. Since COTS platforms are manufactured in high volume, they are generally inexpensive compared to proprietary designs. Additionally, system development times can be reduced because COTS platforms are readily available compared to custom systems.

Computing Using FPGAs

FPGAs are now being used for acceleration in a wide range of applications, both in high-performance servers and embedded computers. The ready availability and high-power efficiency of high-density FPGAs make them attractive to the HPC community. Since their invention in the mid-1980s, FPGAs have been used to accelerate high-performance applications on custom computing machines.

FPGAs have historically been restricted to a narrow set of HPC applications because of their relatively high cost. Over time, however, advancements in process technology have enabled vendors to manufacture chips containing multi-millions of transistors [Ref 1][Ref 2]. The architectural enhancements, increased logic cell count, and speed contribute to an increase in FPGA logic compute performance. For instance, with an average 25% improvement in typical clock frequency for each FPGA generation, the logic compute performance (clock frequency increase x logic cell count increase) has improved approximately by 92X over the past decade while the cost of FPGAs has decreased by 90% in the same time period (see Figure 1). These developments have made it feasible to perform massive computations on a single chip at increased compute efficiency for a lower cost. Figure 1 shows the advancement of FPGA density, speed, and reduction in price over time.

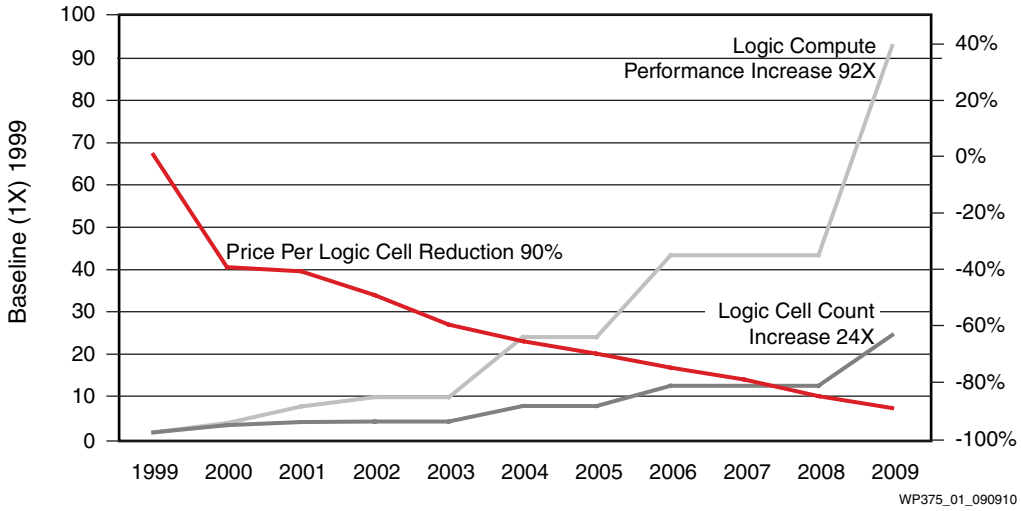

Figure 1: **Historical Advancement of FPGA Technology**

Massive Parallelism Offered by FPGA Architecture

FPGAs are composed of a large array of configurable logic blocks (CLBs), digital signal processing blocks (DSPs), block RAM, and input/output blocks (IOBs). CLBs and DSPs, similar to a processor's arithmetic logic unit (ALU), can be programmed to perform arithmetic and logic operations like add, multiply, subtract, compare, etc. Unlike a processor, in which architecture of the ALU is fixed and designed in a general-purpose manner to execute various operations, the CLBs can be programmed with just the operations needed by the application. This results in increased compute efficiency.

Figure 2 shows the general architectural layout of an FPGA.

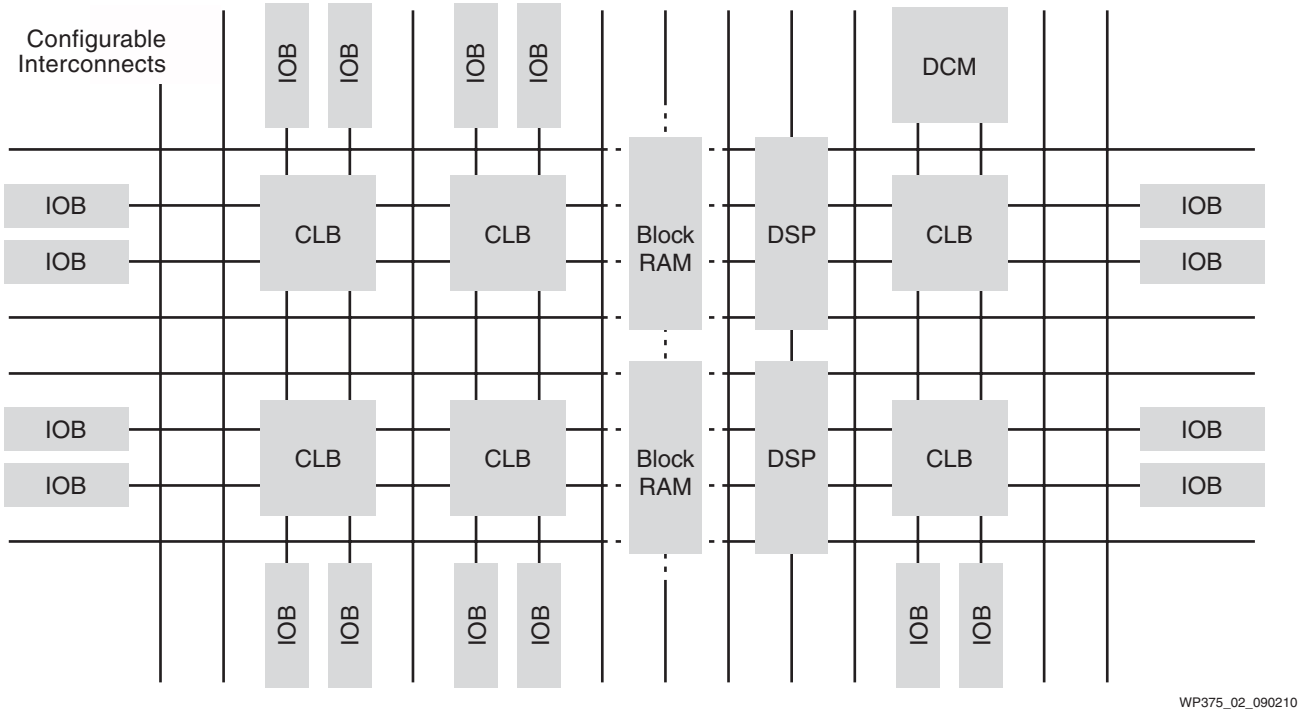

Figure 2: **High-Level Block Diagram of FPGAs**

Depending on the type of operators used, CLBs and DSPs can perform integer, floating point, and bitwise operations. The results of the operations are stored in the registers present in CLBs, DSPs, and block RAM. These blocks within an FPGA can be connected via flexible configurable interconnects. The output of one operator can directly flow into the input of the next operator, meaning that the FPGA's architecture lends itself to the design of data flow engines.

The FPGA architecture provides the flexibility to create a massive array of application-specific ALUs that enable both instruction and data-level parallelism. Because data flows between operators, there are no inefficiencies like processor cache misses; FPGA data can be streamed between operators. These operators can be configured to have point-to-point dedicated interconnects, thereby efficiently pipelining the execution of operators.

The parallelism offered by FPGA architecture can be easily seen in a few examples of HPC-relevant parameters:

- Internal bandwidth to move the operands and results of application-specific ALUs are in order of terabytes/sec (TB/s)
- Throughput on integer operations are in the order of Tera-operations/sec (TOPS)
- Throughput on floating point operations are in the order of gigaflops/sec (GFLOPS)

Memory Interface

The IOBs in the FPGA architecture, as shown in Figure 2, offer several features that can be interfaced with computing system components, and in particular, are designed to support various memory and processor-interface standards. For instance, FPGAs can support multiple DDR3 memory controllers—as many as six DDR3 controllers on FPGAs with the highest densities. The higher the number of memory controllers on an FPGA, the higher the bandwidth to the external memory. In addition to the DDR3 interface, FPGAs also provide support to interface with DDR, DDR2, RLDRAM, and QDR SRAM memories [Ref 3][Ref 4].

Processor Interfaces/Protocols and In-Socket Accelerators

FPGA architecture provides support to interface and run PCIe® Gen1/Gen2/Gen3, Intel's Front Side Bus (FSB), and Quick Path Interconnect (QPI) protocols. Support for these processor interfaces and protocols enables the computing applications running on FPGAs to interact with the processor and access the data required to accelerate the applications.

A new development in recent years is the in-socket FPGA accelerator. With the ability to run FSB and QPI protocols on FPGAs, one or more processors in a multi-processor server can be replaced with FPGAs, allowing portions of the application to be accelerated using in-socket FPGA accelerators. In-socket accelerators provide the additional capability of keeping the data coherent with the processor memory space compared to PCIe-based accelerators. For instance, GPU accelerators are all PCIe-based and cannot keep the data coherent with the processor memory space; an FPGA in-socket accelerator provides this unique capability. This fact has important implications for the type of applications that can be accelerated as well as for the accelerator systems programming model [Ref 5].

The convergence of storage and Ethernet networking is driving the adoption of 40G and 100G Ethernet in data centers. Traditionally, data is brought into the processor memory space via a PCIe network interface card. However, there is a mismatch of

bandwidth between PCIe (x8, Gen3) versus the Ethernet 40G and 100G protocols; with this bandwidth mismatch, PCIe (x8, Gen3) NICs cannot support Ethernet 40G and 100G protocols. This mismatch creates the opportunity for the QPI protocol to be used in networking systems. This adoption of QPI in networking and storage is *in addition* to HPC.

Doubling Performance Generation to Generation

Xilinx FPGAs double their device density from one generation to the next. Peak performance of FPGAs and processors can be estimated to show the impact of doubling the performance on FPGAs [Ref 6], [Ref 7]. This doubling of capacity directly results in increased FPGA compute capabilities. Figure 3 shows the predicted peak single- and double-precision performance of FPGAs versus multicore AMD Opterons over the last three generations. As of 2009, an interesting trend to observe is the widening performance gap over generations that is favoring the FPGAs.

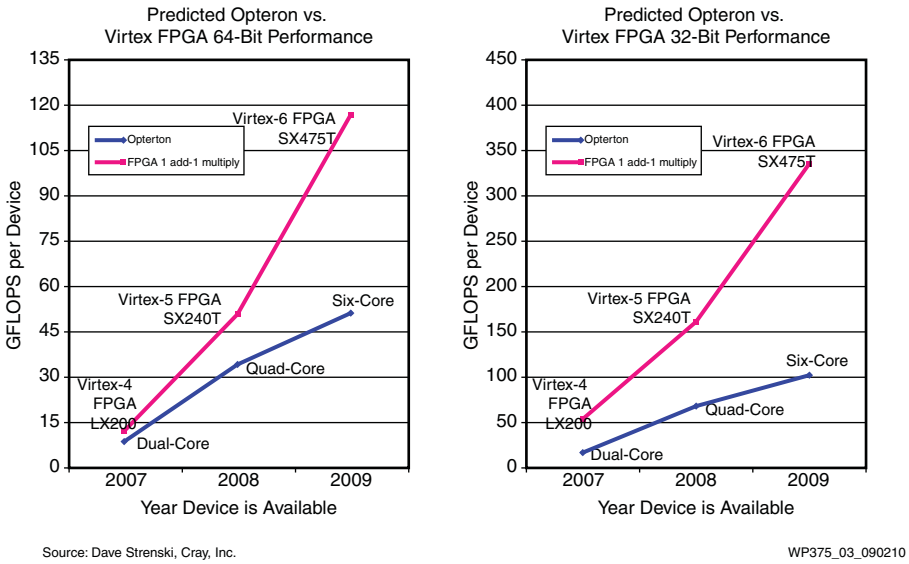

Figure 3: **Single and Double Precision Performance of FPGAs versus Multicore CPUs**

Compared to CPUs released by Intel and AMD in 2010, the performance delivered by one Virtex®-6 FPGA is competitive to one CPU. According to an article by Jeff Layton, Ph.D., Dell Enterprise Technologist - HPC, the peak performance of one Intel Nehelam-EX CPU and one AMD Magny-Cours CPU is 72.25 GFLOPS and 110.25 GFLOPS respectively for double precision operations while the predicted performance of the Virtex-6 SX475T FPGA is estimated around 116 GFLOPS [Ref 8].

Performance to Power Efficiency of FPGAs

FPGAs tend to consume power in tens of watts, compared to other multicores and GPUs that tend to consume power in hundreds of watts. One primary reason for lower power consumption in FPGAs is that the applications typically operate between 100–300 MHz on FPGAs compared to applications on high-performance processors executing between 2–3 GHz.

The ability to parallelize the applications on FPGAs, coupled with lower power consumption compared to CPUs and GPUs, results in increased performance-to-power-efficiency of FPGAs. For instance, an application that runs 10X faster than a

multicore at 4X lower power results in 40X improvement in performance-to-power-efficiency on FPGAs.

Harnessing the Power of Xilinx FPGAs

The Virtex series of FPGAs provides the features and compute density required to accelerate various high-performance applications. Table 3 summarizes the peak performance offered by one of the devices, the Virtex-6 SX475T FPGA.

Table 3: **Virtex-6 SX475T Device Peak Performance Estimates**(1)

Performance Parameters	Peak Performance
I/O bandwidth (GB/s)	28
Memory bandwidth (GB/s)	34
Double precision compute (GFLOPS)	160(2)
Single precision compute (GFLOPS)	450(2)
Compute (32-bit GOPS)	5,579

Notes:

1. Assume four DDR3 at 1,066 Mb/s, 32 LUTs for 32-bit adders for GOPS, all transceivers dedicated to I/O bandwidth at 64B/66B encoding.
2. Floating point estimates provided by Dave Strenski, Cray, Inc.

Though peak numbers are useful for assessing the capabilities of overall silicon architecture, speed-up measurements on real-world applications illustrate the realizable gains obtained with FPGAs. Table 4 lists acceleration of various applications using Virtex FPGAs.

Table 4: **Acceleration Benefits on Virtex FPGAs**(1)

Algorithm/Application	FPGA	CPU	Speed-Up Over Processor
Cryptography: DES	Virtex-6	Intel Quad Core i7 at 2.67 GHz	101X
Cryptographic Key Recovery: NTLM	Virtex-6	Intel Quad Core i7 920 at 2.67 GHz	20X
Seismic Imaging: Convolution	Virtex-5	8-core Xeon at 2.66 GHz	240X
Proteomics: InsPecT/MS-alignment	Virtex-5	Xeon 2-core at 2.13 GHz	100X
Financial options valuation: Quadrature methods	Virtex-4	Pentium-4 at 3.6 GHz	33X
Dense linear equations: LU factorization	Virtex-5	Xeon Woodcrest at 3 GHz	140X
Sparse iterative equations: Conjugate gradient	Virtex-5	Xeon Woodcrest at 3 GHz	82X

Notes:

1. Data sources: Pico Computing, Maxeler Technologies, Convey Computer, Imperial College, and Accelogic [Ref 9].

Among several factors, the amount of application acceleration realized on FPGA systems is primarily dependent on:

- Choice of accelerator platform, which provides appropriate number of FPGAs, sufficient FPGA device density, memory capacity, memory bandwidth, and I/O bandwidth (between CPU and FPGAs, and between FPGAs)
- Appropriate partitioning of code to run on the FPGAs, taking into account the data transfer latency between the CPU and the FPGA subsystem
- Balancing compute, I/O, and memory bandwidth of the application running on the FPGAs

FPGA HPC Platforms and Out-of-the-Box Experience

Irrespective of the nature of an HPC application, there are certain FPGA infrastructure features that HPC platforms need to support. These features include, but are not limited to, interface to the processor, onboard memory, and a mechanism to interconnect multiple FPGAs on the board. For example, for an HPC application to communicate with the local onboard memory, the FPGA needs to be configured with the necessary memory controllers. To enhance the out-of-the-box experience, Xilinx is working with COTS HPC systems and board vendors to provide users with designs preconfigured with platform infrastructure elements. The availability of these infrastructure designs helps HPC users to focus on application development rather than spending their engineering resources on developing platform infrastructure elements of their own.

FPGA Use Models

In high performance computing, FPGAs are used in several different ways:

- **Connectivity use model**: Used in bridges and switches to connect different interface logic and subsystems.
- **Fixed function hardware acceleration use model:** Used in high throughput data processing.
- **Software acceleration use model:** Used to offload portions of a software application running on the CPU to FPGAs.

Connectivity Use Model

FPGAs have historically provided connectivity and integration within compute systems. FPGAs are used to connect parts of a system that use different logic and connectivity standards and protocols. Examples of FPGA applications in this use model include interfacing with processors, sensor networks, memories, and various backplane standards. The teams designing these systems typically comprise electrical engineers who are well-versed in traditional FPGA design methodologies.

Figure 4 shows how an FPGA can be used to bridge a sensor network and processor. Multi- gigabit rate transceivers provide FPGAs with the ability to interface with high-speed interconnects.

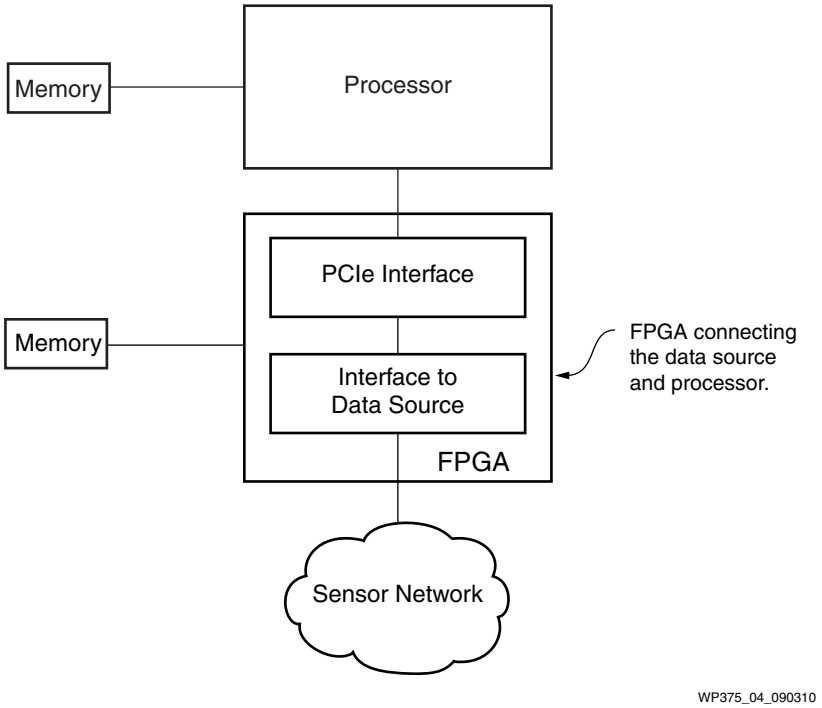

Figure 4: **Block Diagram of Example Connectivity Use Model**

Additionally, I/O blocks are designed to support multi-voltage, multi-standard parallel processing connectivity technologies—HSTL, LVDS, and more. The multi-gigabit transceivers in Xilinx FPGAs support high-speed serial connectivity. These I/O features provide FPGAs with the flexibility to implement several connectivity protocol standards, including PCIe (Gen 1/Gen2/Gen3), PCI, Intel's Front Side Bus, Serial Rapid I/O, XAUI, and Intel's Quick Path Interconnect (QPI). FPGA I/O blocks are designed to interface with different memory types including SRAM, DRAM, and RLDRAM.

An FPGA design can implement multiple memory controllers. Depending on the application need, the data rates and width of the memory controller can also be customized. For instance, servers requiring high memory bandwidth can implement multiple DDR3 controllers at 1,066 Mb/s on Xilinx FPGAs while an HPC system requiring low latency memory access can implement SRAM memory controllers on Xilinx FPGAs. The ability of the FPGAs to interface with different memory types and processor interface protocols makes them suitable to bridge data movement between subsystems that implement different connectivity standards.

Table 5 shows the solution elements offered through Xilinx and its ecosystem vendors to service this use model.

Table 5: **Solution Elements for Connectivity Use Model**

Solution	Specific IP Block	Available From
System Interfaces	PCIe (x8, Gen1)	Xilinx
	PCIe (x8, Gen2)	Xilinx, Northwest Logic, PLDA
	FSB	Xilinx/Nallatech
	Aurora	Xilinx
	XAUI	Xilinx
	Serial Rapid I/O	Xilinx
Memory Interface	DDR3 1,066 Mb/s	Xilinx
	DDR2 800 Mb/s	Xilinx
	DDR 400 Mb/s	Xilinx
	QDR II/QDR II+ SRAM	Xilinx
	RLDRAM	Xilinx

Fixed-Function Hardware Acceleration Use Model

In the fixed-function hardware acceleration use model, FPGAs are used to accelerate fixed functions and tasks in various applications. The ability to interface with high-speed interconnects and process high volumes of data makes FPGAs suitable for applications that demand high-throughput, low-latency data processing.

Typically, multiple compute kernels are executed in parallel to accelerate certain fixed functions on FPGAs. Figure 5 shows the block diagram of a high-throughput radar array. One of the compute- and data-intensive portions of the application is the beam-forming algorithm. Here is a simple exercise to show the power of FPGAs in fixed function accelerator systems. The example radar system uses 10 Gigabit Ethernet to interface with the high-speed I/O processing system, has one or more DDR3 controllers to buffer the computed data on the FPGA subsystem, and uses PCIe Gen2 to interface with the processor.

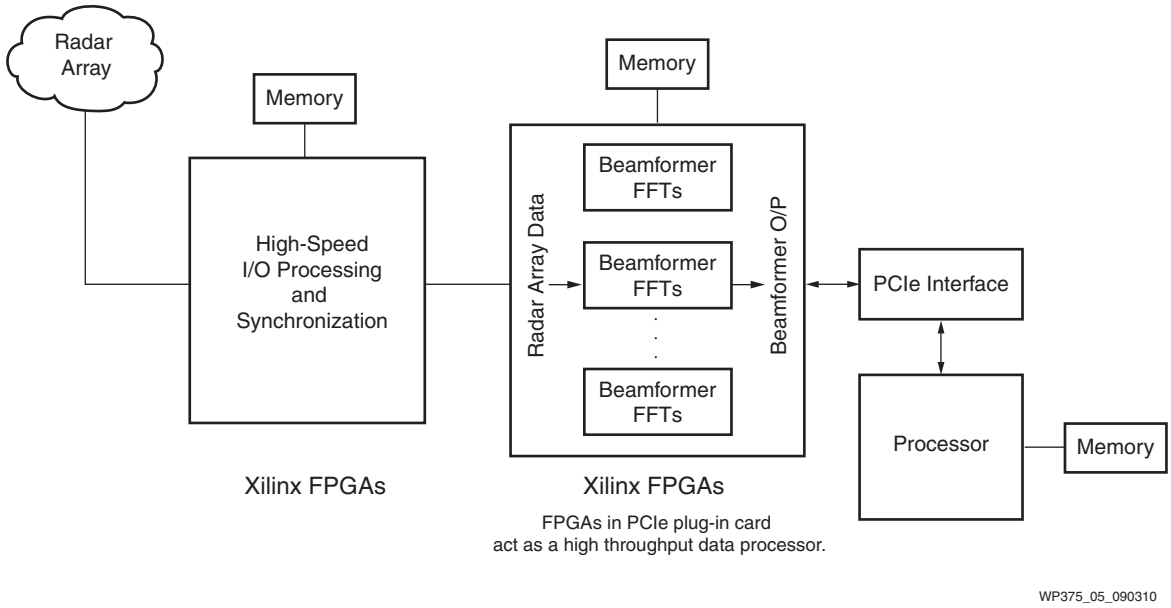

Figure 5: **Block Diagram of Example High-Throughput Processing System**

The generally accepted rule of thumb is that 1 hertz of CPU processing is required to send or receive 1 bit/s of TCP/IP [Ref 10],[Ref 11]. Put differently, 10 Gb/s of network traffic requires a 4-core processor running at 2.5 GHz each. For example, on a Virtex-6 FPGA, 10 Gb/s of network traffic can be processed using a TCP/IP off-load engine (TOE) that includes the MAC and NIC interfaces. The TOE, NIC, and PCIe DMA engines typically use 15–20% of the largest Virtex-6 devices, leaving approximately 80–85% of the device area to accelerate other system functions.

An implementation of the adaptive beam-forming algorithm, wherein an adaptive weight computation is accelerated on FPGAs, is presented in an IEEE course entitled, "ASIC and FPGA DSP Implementation" [Ref 12]. The resource estimates for an 8-channel x 8-beam implementation are shown in Table 6.

Table 6: **Resource Estimates for Adaptive Weight Computation of 8 Channel x 8 Beam Implementation**

Resources	8 Channel x 8 Beam Implementation
DSP48Es	352
Block RAMs	123
LUTs	31,000

An example implementation of the adaptive beam-forming algorithm on an FPGA includes two of the adaptive weight computation functions and the beam-forming function. Using the resource estimates in Table 6, there are enough resources to implement these functions on Virtex-6 FPGAs (e.g., XCV6SX315T) in addition to the TOE function and DDR3 controllers. The IEEE course [Ref 12] lists the throughput of one adaptive weight computation function at approximately 90 GOPS on the FPGA and less than 1 GOPS on a Pentium-4 3.6 GHz processor. With two instances of adaptive weight computation executing in parallel, the Virtex-6 FPGA implementation provides around 180 GOPS.

Other applications that are already taking advantage of accelerating fixed functions on FPGAs include security systems, low-latency trading platforms, image processing,

and analysis systems. The high throughput data processing capabilities coupled with tens of watts of FPGA power consumption enable designers to meet the size, weight, and power constraints of various embedded computers.

Xilinx acknowledges the growing need to provide embedded development tools to decrease the design development times of fixed-function accelerator systems. Table 7 shows a list of IP and tools from Xilinx and its vendors. More information about the results of the BDTi evaluation of the high-level language tools can be found in [Ref 13].

Table 7: **Solution Elements for Fixed-Function Hardware Acceleration**

Fixed Function Acceleration Solution Elements	Available From
Encryption	Helion Technology
TCP/IP off-load	IPBlaze
FFTs, DSP filters	Xilinx
Floating point	Xilinx
DSP cores	Xilinx
Partial reconfiguration	Xilinx
Low-latency trading, image processing, math libraries	Impulse Accelerated Technologies
COTS platforms	Annapolis Micro Systems, Nallatech, Mercury Computer Systems, Curtiss Wright, Alpha Data, and Pico Computing
Development Tools (e.g., high-level language synthesis and graphical development environment)(1)	National Instruments, The MathWorks, Xilinx, and AutoESL Design Technologies(1)

Notes:

1. BDTi certified.

Software Acceleration Use Model

The software acceleration use model addresses the needs of customers who use FPGAs within a high-performance server and speeds up the performance of several software application functions. Examples of these software applications include weapon simulation, nuclear waste simulation, threat analysis, oil and gas reservoir simulation, gene matching, and risk evaluation of stocks and portfolios. The FPGA devices, typically more than one, are incorporated into a standard form-factor computer, for example a 1U server, using PCIe plug-in cards or in-socket FSB/QPI accelerators.

The team developing the application typically is a group of software engineers who have minimal or no experience using FPGAs. The application development process involves writing software code and incorporating industry-standard software libraries. A block diagram of the software system architecture use model is shown in Figure 6.

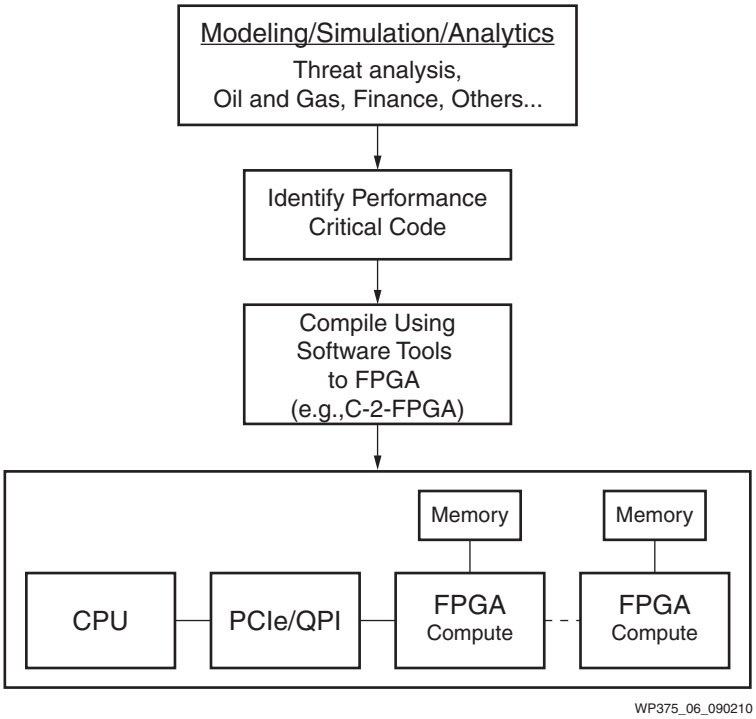

Figure 6: **Block Diagram of Example Software Acceleration Use Model**

The class of users in this use model is in constant need of more compute power. The acceleration offered by FPGAs enables the use of algorithms that are computationally more complex and thus provide more accurate modeling and results. Additionally, given the growing concern of cooling and power costs of data centers, there is a need to build power-efficient servers. FPGAs are a key technology that can be used to accelerate applications while being power-efficient [Ref 14][Ref 15].

Xilinx recognizes the need to deliver a well integrated system with software tools to increase the adoption of FPGAs in this market and is providing the solutions through vendors, including Maxeler Technologies and Convey Computer. These vendors provide FPGA-based HPC servers and computers that accelerate software applications on FPGAs.

Summary

Xilinx FPGAs continue to be used in a myriad of programmable systems. Technology and feature advancements make Xilinx FPGAs ideal for use in HPC applications. Within the HPC market, systems that are implementing FPGAs include multi-function, high-performance servers and clusters as well as fixed-function, high-performance embedded computers. Due to the massive parallelism offered by FPGA architectures, many HPC applications can be accelerated in performance by one or even two orders of magnitude when compared to stand-alone CPUs.

From an FPGA silicon standpoint, the enormous performance gains are due to architecture enhancements and increased chip density, directly correlating to significant applications speedups. Together with low operating power consumption, extremely high performance/power ratios can be realized on FPGA-based HPC systems. In addition to their longstanding reputation as the platform of choice for designing programmable systems, FPGAs are rapidly becoming a valuable and

lasting solution to meet the challenging processing and interface demands of HPC applications. Xilinx is actively working along with its ecosystem vendors to bring various elements of the HPC solution together so that customers can quickly tap into the benefits of using FPGAs in their HPC systems.

References

1. DS150, *Virtex-6 Family Overview*, http://www.xilinx.com/support/documentation/data_sheets/ds150.pdf
2. DS100, *Virtex-5 Family Overview*, http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf
3. UG086, *Memory Interface Solutions User Guide*, http://www.xilinx.com/support/documentation/ip_documentation/ug086.pdf
4. DS186, *Virtex-6 FPGA Memory Interface Solutions Data Sheet*, http://www.xilinx.com/support/documentation/ip_documentation/ds186.pdf
5. *Intel Xeon FSB FPGA Accelerator Module*, http://www.nallatech.com/Intel-Xeon-FSB-Socket-Fillers/fsb-development-systems.html
6. *FPGA Floating Point Performance—A Pencil and Paper Evaluation*, http://www.hpcwire.com/features/FPGA_Floating_Point_Performance.html
7. Revaluating FPGAs for 64-bit Floating-Point Calculations, by Dave Strenski of Cray, Inc. and Jim Simkins, Richard Walke, and Ralph Wittig of Xilinx, Inc. http://www.hpcwire.com/features/Revaluating_FPGAs_for_64-bit_Floating-Point_Calculations.html
8. PetaFLOPS for the Common Man- Pt 3 In the next few yrs what could PetaFLOPS Systems Look Like, by Jeff Layton, Ph.D., Dell Enterprise Technologist - HPC http://www.delltechcenter.com/page/PetaFLOPS+for+the+Common+Man-+Pt+3+In+the+next+few+yrs+what+could+PetaFLOPS+Systems+Look++Like
9. Juan Gonzalez and Rafael C Núñez, "LAPACKrc: Fast linear algebra kernels/solvers for FPGA accelerators," *SciDAC 2009 Journal of Physics: Conference Series 180 (2009) 012042*.
10. *TCP Offload Engine*, http://en.wikipedia.org/wiki/TCP_Offload_Engine
11. A. P. Foong et al., "TCP Performance Re-Visited," *Proceedings of the 2003 IEEE International Symposium on Performance Analysis of Systems and Software*
12. H. Nguyen, "ASIC and FPGA DSP Implementation," *IEEE Current Topics in Digital Signal Processing course*. Woburn, MA. April 22, 2008
13. *High-Level Synthesis Tools for Xilinx FPGAs*, http://www.bdti.com/articles/Xilinx_hlstcp.pdf
14. Convey Computer Announces Record-Breaking Smith-Waterman Acceleration of 172x http://www.marketwatch.com/story/convey-computer-announces-record-breaking-smith-waterman-acceleration-of-172x-2010-05-24
15. R. Dimond et al, MAXware: acceleration in HPC, IEEE HOT CHIPS 20, Stanford, USA, August 2008

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
09/10/10	1.0	Initial Xilinx release.

Notice of Disclaimer