The PetaFlops Router: Harnessing FPGAs and Accelerators for High Performance Computing

Z.K. Baker, T. Bhattacharya, M. Dunham, P.S. Graham, R. Gupta, J. Inman,
A. Klein, G.J. Kunde, A. McPherson, M. Stettler, J.L. Tripp
Los Alamos National Laboratory
Email: *zbaker@lanl.gov*

Abstract

The PetaFlops Router is a new approach to computing wherein network architecture and compute decisions are dynamically customized for a particular application. New Field-Programmable Gate Array (FPGA) and router technologies including multi-gigabit transceivers and application specific blocks can provide vastly improved performance. The PetaFlops Router provides greatly improved data transfer rates, computational efficiency, and programmability compared with application-specific hardware. This represents a fundamental change in high-performance computing through optimized heterogeneous data processing elements and flatter memory structures.

1. Introduction

Achieving high performance on modern computers is difficult without detailed architectural knowledge. The goals of computer designers and end-user scientists are often at odds. A computer designer is happy to tie together a variety of high-end processors, each with its own strengths. Using these systems, without an efficient programming paradigm, requires intimate knowledge of the components, which makes it difficult for anyone other than an expert to achieve maximum performance. The end-user scientist would prefer to not have to learn these system details due to time constraints and the cost of learning through mistakes when one if forced to consider the ramifications of every programming decision. A scientist's time is better spent doing science, not struggling with low-level programming details. The PetaFlops Router provides the performance of highly parallel, custom hardware, while allowing the user to program easy-to-use, high-level concepts.

The PetaFlops Router extends the basic structure used in the design of the Matrix board [6], designed for the CMS Global Calorimeter Trigger used at CERN's LHC experiment, which processes over 340Gbits/s of data from 20,000 sensor inputs. The Matrix board (Figure 1) consists of a Xilinx Virtex 5 LX110T for processing, a Mindspeed crosspoint switch for routing, two banks of SDRAM for local storage, and a μTCA backplane connection. Input data from the sensors arrives through three fiber module on the front of the card. This is processed in the FPGA and interesting particle interactions are passed to the backplane for routing to permanent storage. Processed data can easily generate over a petabyte/s of information and thus it is only possible to capture very specific particle interactions. The FPGA examines the input collisions and implements the selection algorithm. The system is an excellent example of how FPGAs can be applied to data-flow style computation.

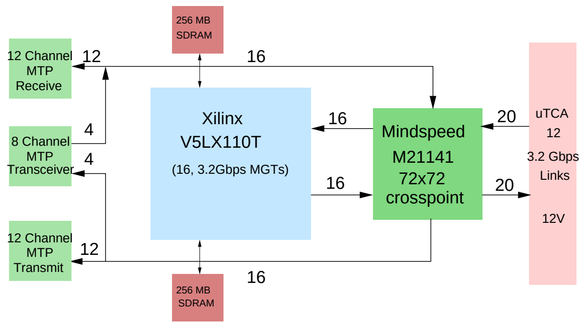

Figure 1. Matrix Board for the CMS Global Calorimeter Trigger

The idea of organizing computation in a data-flow style architecture is not new [1, 7]. Likewise, organizing heterogeneous computation in a data-flow architecture has been explored before [3]. However, the use of FPGAs to organize and control the movement of data, as well as providing certain custom computational blocks, is new as it can provide a mechanism to harness the power of FPGAs without requiring expert FPGA designers for the lifetime of a system. The FPGA can provide lower latency for determining the next phase of computation, as well as having the ability to connect natively to high-speed serial data links. The typical memory hierarchy is simplified and can be customized to provide low latency application specific caches. The cache memories can be shared between multiple nodes of the compute fabric taking advantage of the dynamic switching.

Other hardware systems, XD1 [2], SGI RASC [5], Xtreme Data [10], and Xilinx Accelerated Computing Platform [9], have used FPGAs as an accelerator or network router which can be built up into large clusters. However, they all require expert knowledge to program and use to their full advantage. Convey Computer [8] addresses this usability gap at the instruction level, but relies on conventional computer infrastructure.

2. Approach

The PetaFlops Router adds a co-processor to the Matrix card and connects the Mindspeed crosspoint switch to a configurable local network. Thus, the system is not limited to FPGA-based computation, but is capable of routing to other FPGAs and accelerators. The multi-gigabit transceivers on an FPGA can operate in a variety of modes to connect to a variety of serial standards (e.g., PCIExpress, Serial RapidIO, Infiniband, 1 and 10 Gigabit Ethernet, XAUI). High-speed serial has become the approach of choice for high bandwidth point-to-point connections since it is simpler to synchronize a single data line to a clock than dozens of data lines. FPGAs can include dozens of these serial transceivers, allowing high aggregate bandwidth. This is useful as modern graphics processing accelerators and co-processors are commonly connected to their hosts via PCI Express. The PetaFlops Router combines a large amount of network bandwidth with capable co-processors.

FPGAs can cheaply implement boolean, byte, and fixed-point operators, making them well-suited for signal processing, encryption, and other highly parallel kernels such as genome sequencing. However, certain operations, in particular double-precision floating point, are resource intensive and reduce the total amount of parallelism achievable. Co-processors can provide operators that are costly to implement on FPGAs. While FPGAs have moved beyond 'glue logic', they still retain the volume of input/output pins required for providing high bandwidth connections to memory interfaces; these pins can be used to connect an FPGA to a floating-point accelerator or any other custom ASIC to provide application-specific co-processing.

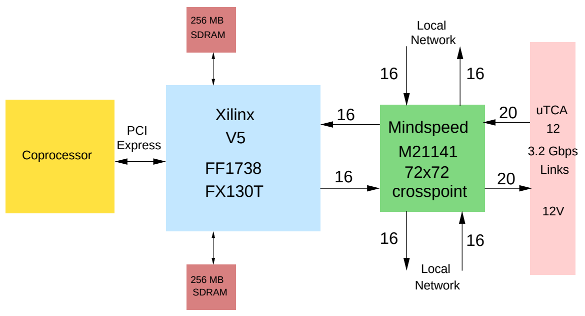

Figure 2. PetaFlops Router Block Diagram

Beyond the FPGA and attached co-processors, the PetaFlops Router includes a highly capable network switch that operates on high-speed serial interconnect. The basis of the network is a protocol agnostic crossbar switch, a commodity device, which is revolutionary when applied at a node level. Crossbars are typically used to connect together groups of nodes. Placing a crossbar on every node allows a significant extension of capability. The crossbar provides a set of n^2 connections to n input ports and n output ports. Thus, any input can be connected to any of the outputs. Not only can it provide an independent path for each of the inputs to each of the outputs, it can also efficiently perform one-to-all communication by connecting an input to all of the outputs. A single crossbar can connect up to 32 nodes, which is useful but not scalable. With a crossbar on every processing node, essentially any network topography can be built from a collection of nodes. The crossbars can be modified at runtime to act as network switches, or can be configured by an application for a particular network topology.

An important aspect is the ability to adapt the system to the particular application. The PetaFlops Router takes application topology into account, and adapts algorithm dynamics and knowledge of the system state to optimize performance. This allows the system to be optimized for a particular class of applications, either by adjusting the crossbar connections or by adjustments to physical connections. The flexibility provided by the crossbars makes the system ideal for a wide range of scales, from a small embedded system to large HPC installations. Each node can be connected to a host, to other nodes, or through a backplane to additional crossbars and racks.

At the node level, the PetaFlops Router programming approach builds up components into functional templates, and then efficiently connects them through the on-node switch structure. This allows a user to program the system in an easy high-level fashion. A user interested in building a signal-processing application might want a collection of FFT blocks, CORDIC cores (for trigonometric functions and exponentials), and matrix/vector units. An image-processing user might want some of the same cores, plus windowing operators and other support for 2-D data. These cores would be available as firmware libraries for the FPGA, built to natively interface with the system. Memory load and stores, as well as data movement between devices and co-processors implementing some other advanced functionality are all controlled within the same framework.

This framework is built on a language that would be familiar to anyone with a background in high-level parallel programming. However, instead of thinking in terms of explicit scheduling and element-by-element movement and computation, computation is scheduled when its data is ready, and data is moved as appropriate for its scheduled computation. In this way, the user can program operations in a way that makes sense for a scientist, and the hardware can handle the underlying issues of timing and operator scheduling. Unlike classic notions of a vector machine, the system can have dozens of functions operating on a stream at one time, with those functions being highly complex (like an FFT or layered matrix multiply). This allows vast parallelism whereas a traditional microprocessor can only execute few multiplies or adds in a given cycle.

A key advance in our architecture is that FPGAs serve as "Smart Glue," not only connecting pieces together but also making decisions about how the computation will move forward through a hybrid data-flow architecture. The Smart Glue is far more capable that just moving data between computational units, although providing scheduling and routing decisions will be a large part of both the development and the contribution of the system. The Smart Glue has the potential to be both self-optimizing and error-tolerant. Link failures or simple congestion (not necessarily network, but computational or memory bandwidth) can make the flexibility of an FPGA controlled routing fabric a powerful feature. Having multiple routes to get data in and out transparently to the user is a very powerful feature.

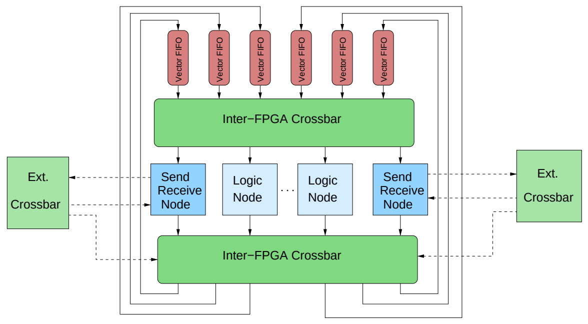

Figure 3. General internal crossbar architecture to connect functional units

The SmartGlue system is based around input/output crossbars and a series of First-In First-Out (FIFO) registers (Figure 3). The FIFO registers are essentially a vector register with a read and write port, such that the system can simultaneously read and write to the same register. The FIFOs are allocated via a renaming scheme that also prevents read/write hazards. The crossbar is a network component that allows connection from any one of the inputs to any number of outputs.

This foundation connects together the components that actually do computation, as well as link the system to other co-processors. The system can work like a classic vector processor in some cases, namely, when the size of the data vectors is smaller than the size of the FIFO register. However, breaking long vectors up into blocks is not always necessary; when there are sufficient computational units and data path to move the data between units, it is possible to start streaming data into the next computation unit before it has fully populated the transfer register.

Each chip has potentially hundreds of computation units and several hundred 512x32 data FIFOs. These handle computation and data movement within one FPGA. Between FPGAs, the system becomes more complex as data bandwidth outside of the chip is necessarily less than on-chip band-width. Communication between FPGAs is handled by external crossbars that are operationally similar to the on-chip crossbars.

One of the keys to the performance of the system is that the high granularity of the individual instructions provides significant native parallelism. Rather than trying to schedule small operations (multiply, add) across nested loops, all of the issues of fast implementation for a limited number of complex functions have already been handled in hardware. A second aspect of high performance is the ability of the system to recognize and generate pipelined implementations based on a dependency analysis. Thus, in many cases, a result can be generated every cycle as long as data is available from memory.

The following code sample gives a sense of how the system functions, and the scale of the instruction operators. The operators used in the assembly code correspond directly to functional units in the SmartGlue system; they are not assembly macros.

```
#get 256 words from address 0
# and put it in name R1
FETCH R1 0 256

#get 256 words from address 256
# and put it in name R7
FETCH R7 256 256

# do an FFT of R1 and put it in R2
FFT R1 R7 R2 R8

# get a mag of the complex output
ABS R2 R3
```

3. Conclusion

In this paper, we present the PetaFlops Router, a new approach to computing wherein network architecture and compute decisions are customized for a particular application. By integrating FPGA, co-processor and router technologies including multi-gigabit transceivers and application specific blocks, the system provides vastly improved performance.

References

[1] F. J. Burkowski. A multi-user data flow architecture. In *ISCA '81: Proc. of the 8th annual symp. on Computer Architecture*, pages 327–340, Los Alamitos, CA, 1981. IEEE CS Press.

[2] Cray, Inc., Seattle, WA. *Cray XD1 FPGA Devel.*, 2004.

[3] J. R. Gurd, C. C. Kirkham, and I. Watson. The manchester prototype dataflow computer. *Comm. ACM*, 28(1):34–52, 1985.

[4] J. Jones. Real-time supercomputing in the search for new physics. Presentation at LANL, November 2008.

[5] SGI, Inc. SGI products: RASC technology. `http://www.sgi.com/products/rasc`, 2006.

[6] M. Stettler, K. Fountas, M. Hansen, G. Iles, and J. Jones. Modular trigger processing: The GCT muon and quiet bit system. In *TWEPP07*, Prague, Czech Rep., 2007.

[7] A. Vinod and K. Vinod. A multiple processor data flow machine that supports generalized procedures. In *ISCA '81: Proc. of the 8th annual symp. on Computer Architecture*, pages 291–302, Los Alamitos, CA, 1981. IEEE CS Press.

[8] S. Wallach. Computer architecture: Past, present, future. `http://www.lanl.gov/conferences/lacss/2008/slides/Wallach.pdf`, Oct. 2008.

[9] Xilinx, Inc. `http://www.xilinx.com/prs_rls/2007/events_corp/0757_intelforum.htm`, Apr. 2007.

[10] XtremeData, Inc. `http://www.xtremedata.com`.