AN FPGA IMPLEMENTATION FOR A HIGH-SPEED OPTICAL LINK WITH A PCIE INTERFACE

Edin Kadric
Dept. of Electrical and Comp. Eng.
McGill University
Montreal, Quebec, Canada

Naraig Manjikian
Dept. of Electrical and Comp.Eng.
Queen's University
Kingston, Ontario, Canada

Zeljko Zilic
Dept. of Electrical and Comp. Eng.
McGill University
Montreal, Quebec, Canada

***Abstract*— To achieve speedup for multi-node, multi-GPU computing platforms, it is necessary to overcome performance bottlenecks in networks based on Ethernet or Infiniband. This paper describes an FPGA implementation of a custom network interface for an optical link between PCIe buses of compute nodes. The implementation uses an Altera Stratix IV chip with integrated PCIe interface logic and high-speed input/output for connecting optical fiber interfaces. The interface is designed with control and buffering for concurrent data transfers. A software driver enables application programs on the host computer to use the high-speed link. A bandwidth of 8.5 Gbit/s was achieved between software applications, exceeding bandwidth reported in recent work [7].**

I. INTRODUCTION

There are a variety of applications where performance demands are such that high-performance computation (HPC) in a parallel or distributed manner is necessary. Commodity chips that integrate up to 10 sophisticated processor cores now support low-cost, small-scale parallel processing in a general-purpose computer. To achieve higher performance, more expensive computers with many multicore chips can be used. Alternatively, parallel computation could be performed using commodity graphical processing units (GPUs) that contain hundreds of simpler cores and provide excellent performance relative to their cost. For even higher performance, computer nodes that combine multicore and GPU chips can be interconnected with high-speed communication links for a large aggregate computing capacity. Attaining the potential performance of such systems requires partitioning the computation in a manner that balances the workload across heterogeneous processing units, and limiting the adverse impact of communication on performance by reducing its volume and by overlapping it with computation.

The physical links for high-speed communication between compute nodes need to provide high bandwidth to feasibly achieve good speedups. Current networking devices used in HPC, such as Ethernet and Infiniband, provide the physical interconnection and protocols for communication using optical links, and custom chips based on these standards are available for system implementation. It is also possible to develop custom protocols for commodity physical links based, for example, on the Small Form-factor Pluggable (SFP) optical interface standard. Because of the high cost for full-custom implementation, the use of a prefabricated field-programmable gate-array (FPGA) is now an economically viable alternative for implementing the hardware support for a custom protocol.

This paper describes the system architecture and logic implementation details for a high-speed optical communication link employing a custom protocol. The communication support is implemented in an Altera Stratix IV FPGA that incorporates high-speed electronic physical interfaces for SFP-based connection to optical fiber, as well as an integrated PCIe interface to convey data to and from an associated computer node. The programmable logic and memory within the FPGA is configured with control and buffering for concurrent bidirectional data transfers with the attached computer node. A software driver supports an application programming interface for a program executing on the attached computer to use this communication support for a high-speed optical link connected to another computer with the same communication support. For a transfer size of 512 Kbits, a bandwidth of 8.5 Gbit/s is achievable with software involved in the communication.

The remainder of this paper is organized as follows. Section II presents related work. Section III describes the system architecture. Section IV explains implementation details. Section V presents results from an assessment of performance for the implementation. Finally, Section VI concludes the paper.

II. RELATED WORK

Due to their high attenuation and susceptibility to interference, conventional copper-based links cannot satisfy the demands for networks with increasingly

higher speeds. As a consequence, they are being replaced with optical fiber implementations, as is the case for the most recent generations of Ethernet, the traditional link of choice for local area network (LAN) communications. Reflecting this transition, Altera is pursuing the development of FPGAs that integrate interfaces for direct connections to optical fibers [1].

This evolution in interconnect bandwidth capabilities must also be accompanied by improvements to the I/O interface with the computer's memory. To address this need, the widely-used PCIe interface standard provides increased bandwidth with every new generation while maintaining backward compatibility and transparency to the underlying implementation. The benefit of the latter feature is exemplified by the use of PCIe links in the IBM Roadrunner supercomputer where a customization of the PCIe channel variables improves performance with no modifications to the overall system architecture [2].

Prior work has considered an FPGA-based implementation to support optical communication links in a data acquisition system [7]. In that work, the architecture and communication protocol are based on a master-slave system where a PCIe interface links a computer to a master FPGA controlling a scalable number of FPGA-based slave front-end cards connected to the master through fiber optic links. This system is used to support a new accelerator at the Facility for Antiproton and Ion Research (FAIR). The authors report reliable operation with data rates of 1.6Gbit/s with PCIe Gen1, whereas our implementation has twice the link speed with PCIe Gen2.

Furthermore, our implementation takes the approach of using PCIe as a link between the host and an I/O subsystem, accompanied by a custom optical fiber link for host-to-host communication. Indeed, PCIe is not deemed efficient for directly linking two hosts and a dedicated cluster is usually used in such interfaces. Nevertheless, other approaches have been explored. For example, the Dolphin Express [8] tries to address the shortcomings of PCIe when directly used for host-to-host communication. PCIe 1.0 is used in that work, where a PCIe-only link is shown to outperform a 10 Gigabit Ethernet one. In our work, PCIe 2.0 is used, thus achieving double the bandwidth.

III. System Architecture

For the implementation presented in this paper, the system consists of two computers connected as shown in Figure 1. Each computer has an Altera DE4 board with a ×4 PCIe Gen2 interface to the computer. Each DE4 board has a Stratix IV FPGA with on-chip memory that can buffer data being sent and received. Each DE4 board also has a High Speed Mezzanine Connector (HSMC) where a small daughterboard with Small Form-factor Pluggable (SFP) slots is attached. Four two-way optical fiber cables are connected to the SFP slots to complete the connection between the two computers.

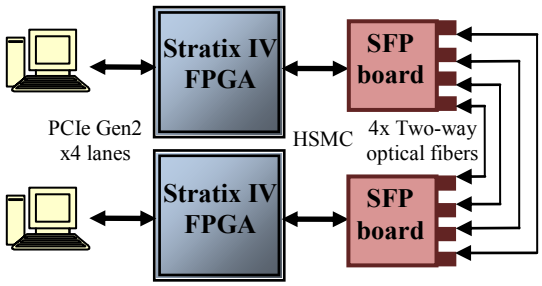

Figure 1. Overview of the system setup

For the implementation of the logic, the system consists of a PCIe module communicating with the computer, and an optical fiber module communicating with the other board, as shown in Figure 2. The two modules are instantiated in the top-level entity of the design and linked to each other with a custom communication protocol. An on-chip memory buffers the data being exchanged between the two modules in order to support the communication between the two computers.

Altera's PCIe Compiler was used to configure a hard IP block in the Stratix IV FPGA with the transaction, data link, and physical layer features of the PCIe specification. The software support for this interface relied on the altpciechdma (ALTera PCIE CHaining Direct Memory Access) driver available in the Linux kernel. Programs in the C language were written to perform basic communication tests and performance measurements using this driver. The driver and the PCIe interfaces both required important modifications in order to allow for faster speed, more flexible communication with the FPGA's internal memory, and more convenient use at the software level.

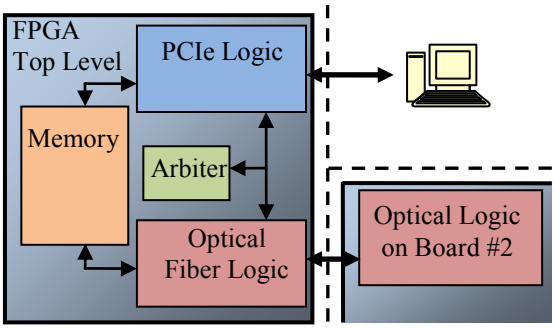

Figure 2. High-level view of the design structure

The optical fiber interface was based on the source code of the loopback demonstration provided by Terasic for its SFP HSMC daughter card. The optical communication was enhanced with internally controllable reset, channel bonding, channel alignment, and start/end of transmission signals. The data is encoded using the 8b/10b line code in order to provide DC-balanced transmission and allow for proper clock recovery by a receiver. A custom synchronization mechanism between the two FPGAs was implemented in order to signal *ready*, *start,* and *end* of transmission states and thus synchronize the two physically separate nodes.

The implementation consumes under 22,000 logic elements in the FPGA. The PCIe interface logic operates at 100 MHz and the optical interface logic operates at 125 MHz.

IV. Elements of the Implementation

A. Memory structure

Each one of the four PCIe lanes is quoted at a maximum physical throughput of 5 Gbit/s, thus forming a 20-Gbit/s link. Each one of the four optical fiber channels runs at 6.25 Gbit/s, thus forming a 25-Gbit/s link. When taking into account protocol, encoding, and software overhead, the available raw bitrate for the payload is reduced. Nevertheless, the optical fiber link remains faster than the PCIe, which can be considered the bottleneck of the system.

This difference in the relative data rates for optical fiber and for PCIe motivated a structure where the FPGA's memory is divided into three buffers so that when a PCIe DMA operation to fill one buffer is completed, the interface does not wait for the buffered data to be transmitted to the other node. Instead, another PCIe DMA operation is initiated to fill a different buffer, while the data in the first buffer is transmitted to the other node through the optical fiber link. With this approach, the PCIe link is always busy with DMA operations. Meanwhile, the optical interface logic waits for the next full buffer for data transmission. As a result, the overall system speed does not drop lower than that of the PCIe link. This also works using only two buffers, but a third one gives more time to potentially process the stored data before the next transfer, together with providing an additional margin to counter a potential transient slowdown in optical fiber communication.

For this approach, each FPGA needs three memory areas for the three buffers used in the design, and because both links allow for two-way communication, the buffers are further divided into two subparts, each storing the data for one of the two directions.

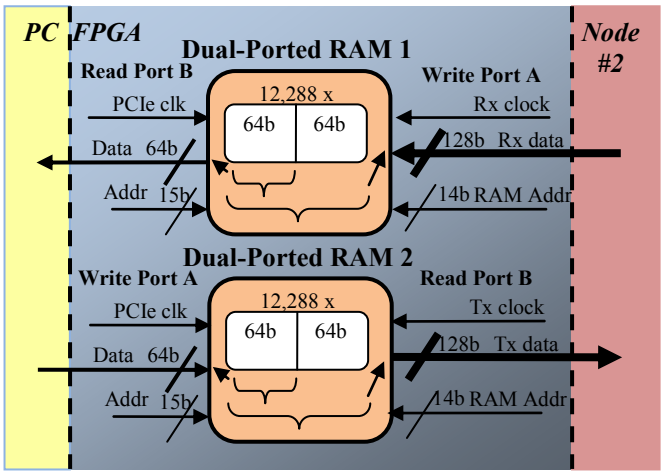

Figure 3. Structure of the FPGA's internal memory

Furthermore, because of the different clock domains, each FPGA actually needs two RAM blocks instead of one. The first RAM is written by the optical fiber logic at the receiver's clock speed and read by the PC at the PCIe clock speed, whereas the second RAM is written by the PC at the PCIe clock speed and read by the optical fiber logic at the transmitter's clock speed. The PCIe accesses the data in 64-bit words, whereas the optical fiber logic sends 32 bits of data per channel each clock cycle, thus accessing one 128-bit word on every clock cycle with all four optical channels involved.

The RAM structures must therefore provide different sizes for the input and output data, with different input and output clocks, as illustrated in Figure 3. The size of each buffer is chosen to be 512 Kbits as justified by the results in Section V. Hence, the PCIe logic deals with the RAM as a group of 24,576 64-bit words accessed with a 15-bit address, whereas the optical fiber interface logic deals with it as a group of 12,288 128-bit words accessed with a 14-bit address.

B. Interface between PCIe and Optical Fiber

After the system is reset, the transmit logic at each endpoint of the optical link immediately begins sending synchronization data. The receive logic at each endpoint performs comma detection, which involves identifying sequences of five consecutive 1s and 0s in the 8b/10b control codes. After successful detection, the four channels are bonded and ready for exchange of data. At this point, dummy data is continuously sent in order to maintain a synchronized state. When the PCIe logic indicates that user data is available to be transmitted to the other node, the optical logic will send *ready* control codes (chosen to be K28.0 in 8b/10b) to the other node. The receiver provides an acknowledgement by sending *start* control codes (K28.2). The transmitter then sends the user data, terminated by an *end* control code

(K28.4). If there is more user data to be transmitted, the *ready/start/*data/*end* sequence is repeated. Otherwise, dummy data is sent continuously to maintain synchronization until the next PCIe DMA is completed.

The PCIe logic and the software driver together control the start and end of DMA transactions by setting three flag bits in the PCIe Base Address Register (BAR), corresponding to the three available memory buffers. When the driver has data available to be sent, it polls the flag for the current buffer. If the flag is 0, a DMA transaction is initiated, and the driver sets the flag to 1 upon completion. Otherwise, the driver has to wait for the buffer to become available after the current data is transmitted by the optical logic. Setting the flag to 1 informs the optical fiber logic that the buffer contains data to be transmitted. The driver then moves to the next buffer. The PCIe logic responds to the setting of the flag by asserting the corresponding flagAssertedOut signal. Each buffer has its own flag bit and flagAssertedOut signal.

As shown in Figure 4, the flagAssertedOut signal is connected to a *flagger* module that conveys control signals to the optical fiber logic. A 3-bit signal SFPtransfer indicates which buffer is ready. After completing transmission for a buffer, the optical logic sets one of 3 SFPflag signals. This signal is conveyed to the PCIe logic through one of three flagAssertedIn signals. When flagAssertedIn is 1, the flag in the BAR is set back to 0, so that a subsequent PCIe DMA transaction can use the corresponding buffer.

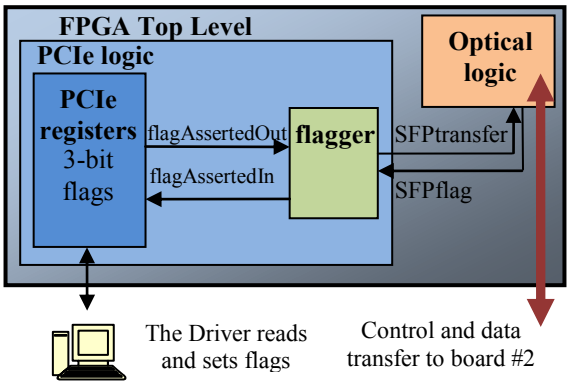

Figure 4. Synchronization between PCIe and optical fiber logic

V. Performance Evaluation

The system has been developed and tested between two Linux computers running Ubuntu 10.04 with Kernel version 2.6.32-33. Each computer has a quad-core AMD Phenom™ II X4 955 processor and 8 Gbytes of memory. An Altera DE4 board connected through a PCIe interface in each computer has a Stratix IV EP4SGX530KH40C2N. Altera Quartus II was used for synthesis of the FPGA design. The SFP transceiver modules for the optical fiber interface are Finisar's FTLF8524P2BNV 1000Base-X model that emit 850-nm light using Vertical-Cavity Surface-Emitting Lasers. Each optical fiber cable has a length of 50 cm. The Lucent Connectors (LC) at the ends of the cables are inserted in the SFP modules.

The specification for the SFP modules [3] indicates a reliable operation on optical cables with lengths up to 70 m over an extended range of conditions. Furthermore, for a supply voltage of 3.3V, the stressed receiver sensitivity is 0.55mW for the four channels. This is the minimum optical power required at the receiver to recover the signal with a Bit Error Rate (BER) of at most 10^{-12}.

The SFP HSMC daughter card supports multiple clock rates: 61.44, 125, 155.52 and 156.25 MHz. It also offers eight SFP connectors: four LVDS (low-voltage differential signaling) and four transceiver based ones [4]. The later four are used in this work.

The handshaking protocol of the PCIe link requires nearly constant time relative to the variation in the amount of data being transmitted. Large transfers therefore reduce protocol overhead at the expense of increased latency. Figure 5 shows plots of the PCIe DMA bandwidth and latency versus the DMA transfer size. The PCIe link capacity is more effectively utilized for large transfers, but there are diminishing returns in performance beyond a certain size at the expense of increased FPGA resource utilization and latency. For example, doubling the transfer size 512 Kbits to 1 Mbit results in only a marginal improvement in bandwith with double the latency. The transfer size was therefore set to 512 Kbits, for which a bandwidth of 8.5 Gbit/s can be achieved.

The optical fiber link bandwidth is 20 Gbit/s. This level of performance lies between the 16-Gbit/s double data rate (DDR) and 32-Gbit/s quad data rate (QDR) of InfiniBand. The 20-Gbit/s performance is also comparable to the bandwidth achievable with two 10-Gbit/s Ethernet links. The optical link described in this paper is therefore appropriate for high-performance applications. Together with latency information, these bandwidth comparisons are shown in Table I.

TABLE I. Comparison of the Optical Link to Other Standards

	10G Ethernet	Infiniband DDR x4	Custom x4
Data rate (Gbit/s)	10	16	20
Latency (µs)	~10	~1	0.73

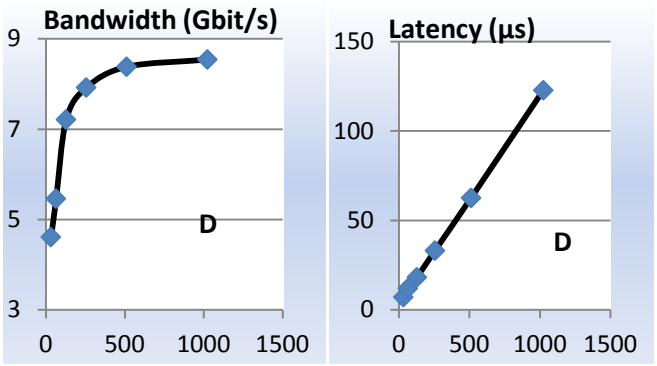

Figure 5. Bandwidth and latency measurements

The intended application of the optical link described in this paper is for high-performance distributed simulation on computers equipped with several graphics processing units (GPUs) to accelerate parallel logic simulation. Latency is especially critical in such an application because the transmission delay for messages between computing nodes must be minimized in order to reduce the frequency of rollbacks, avoid deadlocks, and mitigate adverse performance effects of other events during the execution of a parallel simulation. For the optical link alone, the measured latency is approximately 0.73 µs, reflecting line encoding and the overhead due to the 160-bit packet size. This latency compares favorably with the 1-µs and 10µs nominal latencies for Infiniband and 10 Gigabit Ethernet [5], respectively.

The final application latency for the optical link in this paper is at most 260 µs when the effects of software overhead, the PCIe link, and the buffering within the FPGA are considered. A transfer size less than 512 kbits would reduce this system-level latency, but would also reduce the achievable bandwidth. In comparison, Koop et al. [6] consider the performance of PCIe 2.0 with QDR InfiniBand and also observe that the throughput stagnates around its maximum value for transfer sizes of 512 Kbits and larger, whereas the latency consistently increases with increasing message sizes larger than 64 bytes.

VI. Conclusions and Future Work

For utilizing commodity compute nodes in high-performance computing applications, the interconnects play a critical role in ensuring the efficient execution of parallel computation algorithms. These interconnects must provide high throughput and low latency, which is presently achieved through optical fiber based links such as 10 Gigabit Ethernet and InfiniBand. The high-speed interface described here links the compute nodes together with four optical fiber channels (25 Gbit/s total), and can provide significantly better performance than Ethernet and InfiniBand counterparts. More importantly, our implementation eases the programming task while also avoiding link bottlenecks.

We built a networking interface directly on standard PCIe, that provides good performance, reliability, scalability, and avoids latencies associated with traditional networking interfaces. On each node, the data can be stored in the FPGA's internal memory. The node communicates by DMA transfers with the FPGA. Our software driver allows its use from a high-level language user program. Due to the triple buffering structure, the overall link speed is as fast as the PCIe interface, and the FPGA can conveniently process the data as it transits through it. Including protocol and software overhead, the data rate of the design from one C program to another was measured to be 8.5 Gbit/s.

In future, we plan to extend the same concept to large-scale cache-coherent heterogeneous multiprocessors using FPGAs, similar to NUMAchine [9].

References

[1] Altera Corporation. “Overcome Copper Limits with Optical Interfaces.” Internet: http://www.altera.com/literature/wp/wp-01161-optical-fpga.pdf, April 2011 [accessed February 2012].

[2] P. Germann, M. Doyle, R. Ericson, S. Lewis, J. Dangler, A. Patel, “Pushing the limits of PCI-express: A PCIe application within an IBM supercomputing environment,” *Electronic Components and Technology Conference*, pp. 495-501, 2008.

[3] Finisar Corporation. “Product Specification - 4.25Gb/s RoHS Compliant SFP Transceiver - FTLF8524P2xNy.” Internet:http://www.finisar.com/sites/default/files/FTLF8524P2xNy%20Spec%20RevJ.pdf [accessed February 2012].

[4] Terasic technologies. "SFP HSMC User Manual." Internet: www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=342&FID=c4b7d99c26cb60c9e3bdb4f2ecdf10dd, [accessed February 2012].

[5] HPC Advisory Council. “Interconnect Analysis: 10GigE and InfiniBand in High Performance Computing.” Internet: http://www.hpcadvisorycouncil.com/pdf/IB_and_10GigE_in_HPC.pdf 2009 [accessed February 2012].

[6] M.J. Koop, Huang Wei, K. Gopalakrishnan, D.K. Panda, "Performance Analysis and Evaluation of PCIe 2.0 and Quad-Data Rate InfiniBand," *IEEE Symp. High Performance Interconnects,* pp.85-92, August 2008.

[7] S. Minami, J. Hoffmann, N. Kurz and W. Ott, "Design and Implementation of a Data Transfer Protocol Via Optical Fiber," *IEEE Transactions on Nuclear Science, vol. 58*, no. 1, pp. 1816-1819, 2011.

[8] V. Krishnan, “Towards an integrated IO and clustering solution using PCI express,” *IEEE International Conference on Cluster Computing 9th*, pp. 259-266, September 2007.

[9] A. Grbic, S. Brown, S. Caranci, R. Grindley, M. Gusat, G. Lemieux, K. Loveless, N. Manjikian, S. Srbljic, M. Stumm, Z. Vranesic, and Z. Zilic, "Design and Implementation of the NUMAchine Multiprocessor," *Proceedings of the 35th IEEE Design Automation Conference*, San Francisco, June 1998.