A Hypertransport based low-latency reconfigurable testbed for message-passing developments

Mondrian Nüssle, Holger Fröning, Alexander Giese, Heiner Litz, David Slogsnat, Ulrich Brüning

Abstract

High-bandwidth, low-latency MPI implementations are of key importance for clustered, parallel computing machines. The performance of message-passing is based on the underlying network hardware, the network-interface architecture as well as the software design of the message-passing library and ultimately the parallel applications itself. This paper analyzes the application of the HTX-Board, a Hypertransport Extension (HTX) expansion board, in conjunction with the HT-Core, an open-source Hypertransport Intellectual Property Core for the development, rapid-prototyping and implementation of parallel message-passing systems. In particular the architecture and special hardware mechanisms will be presented and the performance thereof will be evaluated. This new platform enables high-performance, low-latency networking with end-to-end latencies of less than 1 μs using reconfigurable logic.

1 Introduction

Message-passing systems have been extensively used in the past years as basis for parallel computing machines of all sizes. Cluster computing is an especially noteworthy implementation. In the November 2006 top 500 list of the fastest computers of the world [1] more than 72% of the entries are cluster computers. Usually the nodes of a cluster are assembled of commodity of the shelf components (COTS). To achieve greater parallel performance, often specialized network adapters are used. Well known examples for these kind of network adapters and networks are Myricom's Myrinet [2], Quadrics QsNETII [3], and Infiniband from many vendors [4], [5]. This results in a substantial speedup of the system. In 2004 PathScale introduced Inifinipath [6], the first Hypertransport [7] enabled Infiniband hostadapter (HCA) fitting into an HTX slot and featuring very low-latency communication. Hypertransport in itself offers the potential to reduce network latencies substantially in contrast to PCIe and PCI-X, because of the direct communication between the host processor and the network interface controller (NIC) without any intermediate bridging (see [8] and [9]). The low-latency and high bandwidth that is offered by Hypertransport and the availability of a standardized slot form factor led to an increased interest in special Coprocessors and application accerelerator hardware connected to the processor via Hypertransport. In message passing applications the concepts of NIC and coprocessor can be conceptually combined to form network coprocessors that may provide special features to the host CPU such as for example direct support for MPI collective operations. The Message Passing Interface (MPI [10]) is the most popular programming interface for parallel, message-passing applications, the de-facto standard in this area. While there have been and still are several open source and closed-source MPI implementations available, our focus with this platform will be on OpenMPI [11] here since it can easily be adapted to new networking hardware.

This paper presents the HTX-Board Hypertransport reference platform [12] in conjunction with the HT-core IP [8] as a perfect platform for message passing hardware prototyping and research. The development of both was substantially driven by the need to have a prototype platform for rapid prototyping of NICs and network hardware. The usage of current, high-performance FPGA technology together with high-speed serial transceiver components, memory and a high speed Hypertransport interface allow for excellent performance of such prototypes. The reconfigurability of the platform leads to an increased level of design space exploration and hardware-software codesign research in the area of NICs and network hardware, not previously possible.

The remaining paper is organized as follows. Section 2 gives a short overview of the HT Core IP, Section 3 presents the HTX board briefly. In section 4 the resulting platform is analyzed in respect to message passing applications. Section 5 presents first experimental results and the paper closes with a conclusion and an outlook in Section 6.

2 The HT Core IP

Hypertransport defines a packet oriented protocol used mainly for interchip communications. There have been several versions of the protocol. Current AMD Opteron processors use the 2.0b version [6]; the newest released specification is the Hypertransport 3.0 [13]. Opteron processors use the Hyper-

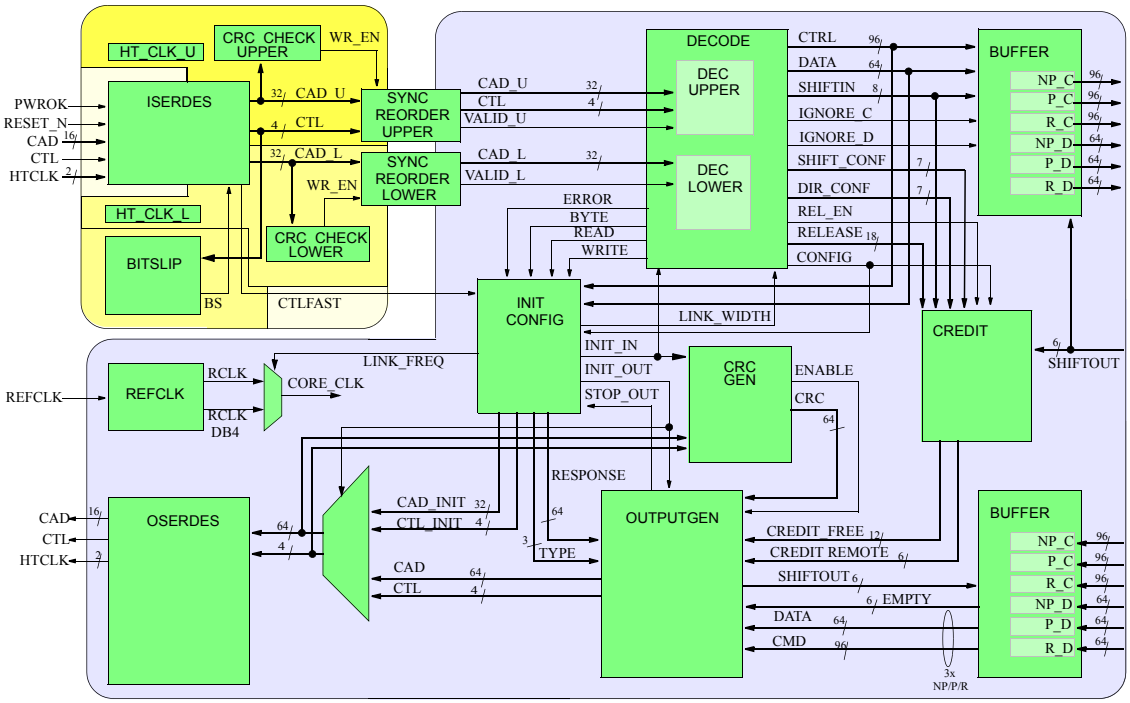

Fig. 1. Blockdiagram of the HT core

transport protocol to communicate with each other in a multi-processor system as well as to communicate with main memory and peripheral chips. For this communication the Hypertransport IO link is used. This protocol is implemented in the HT core IP. A complete introduction to Hypertransport is not in the scope of this paper, but more detailed information is available in the public specification itself [6]. Hypertransport links support several different link-widths. The HT core supports 8-bit and 16-bit wide links. Data is always transported with double-data-rate (DDR) on the link which means that a link with a clock rate of 200MHz actually transfers 400 Mega Transfers per second. A link can have different link speeds, the exact link speed to use is negotiated on system startup (exactly as the bit width is also negotiated). While the HT Core itself is not limited here, the hardware platform limits the possible link frequency to 400MHz, since the Xilinx Virtex4 I/O cells support at most 1Gbit/s and 400MHz is the highest availalbe Hypertransport frequency that renders a data rate smaller than 1Gbit/s. So realistically 200MHz and 400MHz link frequencies can be supported.

Figure 1 shows a block diagram of the core. On the left side the Hypertransport interface is shown. It consists of the 16 CAD (Command, Address, Data) lines, the CTL signal to differentiate data from control and the various clock lines. There is one incoming clock for each receiving 8-bit width as well as one outgoing clock for each 8bit data word. This enables source synchronous operation. Incoming packets are 4-times deserialized and then cross the clock domain border to the core clock using a synchronizing FIFO. The decode engine decides wheter the incoming packet is a configuration packet, a posted, non-posted or response packet. Configuration packets are forwarded to the init and config block, while the other packets are inserted into the respective queue on the right side of the diagram. These queues (again FIFOs) implement the interface to the user application logic. On the sender side, there are also queues on the user application interface. Data is forwarded to the Outputgen unit which converts the data into correct Hypertransport packets. The data is then serialized and put on the outgoing link. There are also some more units in the core, notably a CRC checking and a CRC generating unit and a credit unit which implements the credit based flow-control of Hypertransport.

TABLE I
RESSOURCE REQUIREMENTS IN A VIRTEX-4 FX 60 FPGA

Ressource	8 bit link	16 bit link
Logic Slices	2699(10%)	4123(16%)
FIFO16/RAM16s	30(12%)	30(12%)
DCM_ADVs	3(25%)	4(33%)
ISERDES	10(1%)	19(2%)
OSERDES	9(2%)	17(2%)

Table I summarizes the ressources dedicated to the HT core when mapped on to the Virtex4 FX 60 which is used on the HTX board.

The latency of the core has been highly optimized to enable the design of very low latency applications, which is especially important in message passing.

TABLE II
HARDWARE LATENCIES OF THE HT CORE

Direction	Clock Cycles	Delay @ HT200	Delay @ HT400
In	11	55ns	27.5ns
Oout	7	35ns	17.5ns

TABLE III
PEAK BI-DIRECTIONAL BANDWIDTH

Bit-width	Bandwidth @ HT200	Bandwidth @ HT400
8-bit	800MB/s	1.6GB/s
16-bit	1.6GB/s	3.2GB/s

Table II shows the latencies of the input and ouput paths in clock cycles and in Wall time with different link speeds. So, in a 400MHz environment, the core will only contribute 17.5 + 27.5 ns = 45ns to the latency of a message (outgoing latency on the sending NIC and incoming latency on the receiving NIC). It is interesting to note that the input latency is higher because of the synchronization FIFO which is responsible for transferring the input data into the HT core's clock domain. The theoretical peak bandwidths of the different configurations are summarized in Table III. Some application bandwidth numbers are presented in Section 5.

3 HTX Board Architecture

The HTX-Board provides the physical platform for the system presented here. It is an HTX compatible board featuring a Xilinx Virtex4 FX 60 or 100 [15] as its core component. The Xilinx Virtex4 FX FPGA series has been chosen as it features fast Serialized/Deserializer (Serdes) components which can be used for serial links in a network design. Figure 2 shows a block diagram of the board. The HTX connector is directly connected to the I/O-pins of the FPGA for use by the HT Core IP described in Section 2. Six of the 16 available serdes of the Virtex4 FX-60, are connected to SFP connectors on the frontside of the board. These SFP connector can be populated with modules conforming to the SFP Multi Source Agreement [16], usually these are optical transceivers, but there are also electrical solutions available for short distances or loopback applications. Other important features of the board in respect to message passing applications are the DDR2-DRAM components located on the right. This memory can be used as buffer space or as system memory for the embedded PPC405 processors [17], of which two are embedded into the FPGA. To further support the processors there is also a Flash component. For FPGA programming purposes, a JTAG connector as well as a USB connector are available on the board; there is also a Xilinx Platform Flash to support automatic programming after power has been applied to the system.

The 6 serial links allow to easily build a fully connected network of 7 nodes. If a switch is implemented in the FPGA, different network topologies become possible. One direct network topology that is very promising with 6 links is the 3-D torus, other topologies include a hypercube for a 64-node system. Of course any network topology can be implemented as long as the node degree is not higher than 6. Each FPGA serdes can provide 6.25 Gbits/s as of the current Xilinx specification. SFP modules are readily available up to 4 Gbit/s. In current designs, only 2 and 4 Gbit/s have been tested, this also matches the internal frequencies of 100 or 200 MHz, that are natural because of the available Hypertransport frequencies. With all six SFPs in 4Gbit/s mode, an aggregate bandwidth of 24Gbits in each direction is available translating to 4.8 GByte/s bidirectional bandwidth if the obligatory overhead because of the 8b/10b-coding is taken into account. If a higher bandwidth than 4GBit is requested, several links can be bundled to form a x2 or x4 link (similar to the PCIe or Infiniband). The optical transmission makes the inter-node communication robust in terms of errors, and allows relatively long ranges to be bridged. Network applications including NICs and switches or routers need buffer space. The Virtex FPGA in the FX-60 configuration features nearly 4Mbit of internal SRAM organized in 18kbit BRAM primitives. In addition up to 256 MByte of external DDR2-DRAM are available. With the low latency of the Hypertransport interconnect, it is also worthwile to explore the possibility to use host memory instead of local DRAM. The embedded PPC405 processors can be applied as protocol, control or management processors in a networked application. For example, more complex functions of the NIC or router can be implemented in software on one or both of these processors. Of course, this means generally a lower performance, as these processors are relatively simple 32-bit RISC implementations that run with no more than 450MHz in the fastest FPGAs available.

For debugging and expansion several features are avilable on the board. There is one optional assembly were the 3 SFP connectors are traded for a dual ATOLL [18] link connector. This connector interfaces over 20 LVDS I/O pins of the FPGA, which can be arbitrarily used. Using the mezzanine connector another 20 LVDS pins are available. Using the JTAG port, a Xilinx Platform JTAG cable and the ChipScope software, in circuit debugging of FPGA logic is possible. Finally the HTX extender card enables Hypertransport debugging with the aid of a Logic State Analyzer. Figure 3 shows the HTX board and the Extender card plugged into the HTX slot of a motherboard. The two ribbon cables connect to a Logic State Analyzer.

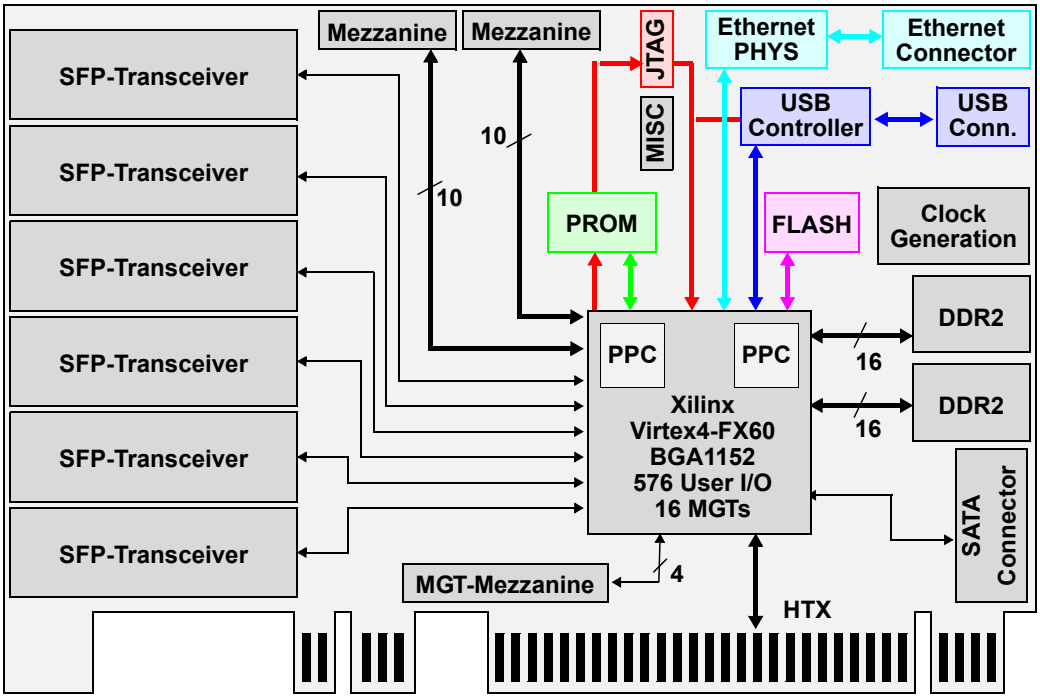

Fig. 2. Schematic view of the HTX board and its components

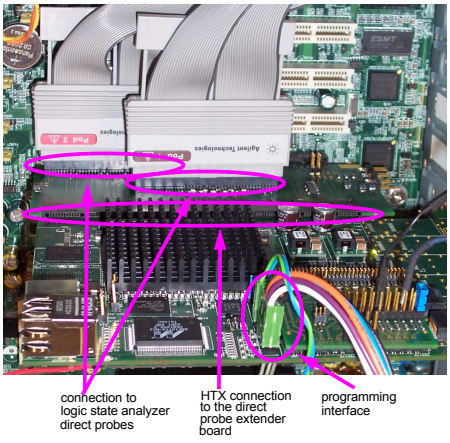

Fig. 3. In-system verification setup

4 Application in Message Passing

The HTX-board platform with the HT core is perfectly suited for message passing applications. The platform enables the rapid, reconfigurable design of network implementations that feature a very low half roundtrip latency. There have been numerous other projects involving FPGAs and message-passing networks for example [19], [20] and [21] just to name a few. The platform presented here offers unsurpassed latency and bandwidth. The most important factor here is the low latency of the Hypertransport interface of current AMD Opteron processors and the low latency of the HT core. Of course Hypertransport related latency is lower if a 16bit implementation is chosen over a 8bit implementation, and also if a link speed of 400 MHz is used instead of 200 MHz. The rest of the message passing system can be custom built and with carefull design can also exhibit low latency costs. On the other hand, the platform also offers reasonable bandwidth, for example in a 16bit/400MHz configuration 3200 MBytes bidirectional bandwidth from/to the host; this would compare to about 6 PCIe lanes. The bandwidth of the serial links matches these numbers nicely. Of course, the FPGA enables researchers to implement custom NIC functionality and experiment with different host-to-network interfaces. For example, custom hardware to support efficient message passing, special protocol engines or support for collective MPI operations in hardware becomes possible. The low latency and high bandwidth of the platform enables users to develop prototype designs that perform in a close relation to a custom solution involving the design of special ICs. Thus possible research results can be tested in real systems without the risk involved when building an Application Specific Integrated Circuit (ASIC).

5 Experiments and first results

In this section we describe two experiments we performed with the platform. The first setup is used to measure basic performance data of the Hypertransport core in conjunction with a complete PC

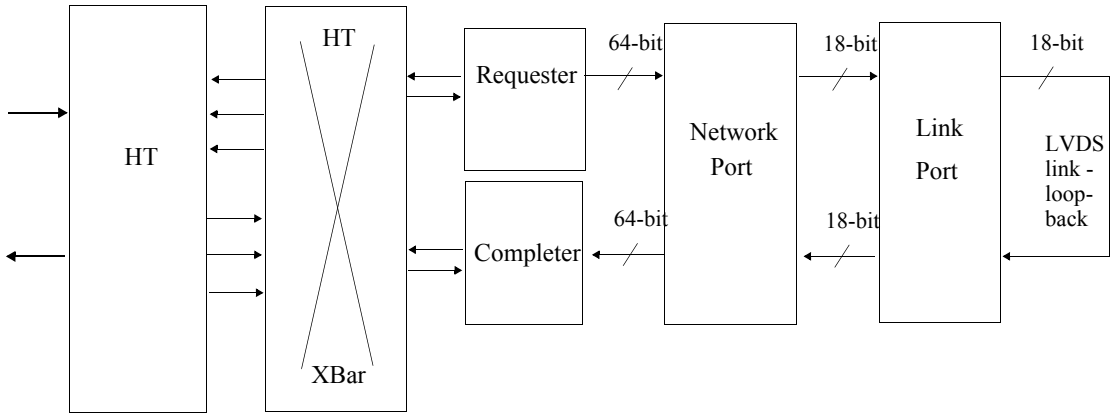

Fig. 5. Blockdiagram of the message passing experiment design

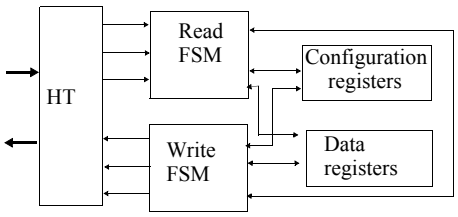

Fig. 4. Block diagram of the HT example design

system. The second setup is a simple design to measure achievable message passing performances. All experiments were performed using one PC-system equipped with one AMD Opteron 246 processor (2Ghz clock, 1MB 2nd-level-cache), 2GB RAM, Iwill DK8 HTX motherboard featuring one HTX socket. The software environment was SuSE Linux 10.0 installation with Linux Kernel 2.6.X. The BIOS of the motherboard has been replaced with LinuxBIOS [22]. For the first experiment the FPGA was configured with the HT example design that is available from the CoEHT download page [14]. The block diagram of this design is shown in Figure 4. The Hypertransport core used for these measurements was 8-bit wide and had a link clock of only 200MHz. We expect the measurement results to improve significantly once we repeat these measurements using a 16bit, 400MHz core. The results from read latency measurements as well as write bandwidth measurements are summarized in Table IV and Table V. Programmed I/O (PIO) read bandwidth is very low, since the Opteron processor initiates single 32-bit read packets regardless of operand size. So a 64-bit load instruction is translated into two 32-bit Hypertransport read packets that are executed sequentially. These results show that PIO mode receive operations are not efficient on the platform. So for our first message passing measurements we chose to implement the send functionality using PIO write operations and the receive operation using DMA write transfers while polling on main memory.

First network interconnection hardware suitable for a message passing system has been implemented on the HTX platform. This is an early version offering still possibilities for improvement. Figure 5 shows a block-diagram of the implemented system. Currently, send operations are realized by using a direct copy of the data to mapped I/O memory. This approach is similar to the traditional PIO approach and generally offers low latency. A disadvantage is the high CPU utilization, i.e. every byte to be sent must be touched by the CPU. The current implementation uses I/O memory marked as uncachable and the write-combining buffer of the Opteron processor originally introduced to x86-processors to optimize store performance to framebuffer memory. The biggest message that can be sent in one fragment using this approach is 64byte in length, corresponding to the size of one cache-line, the size of the write-combining buffers. After the data to be sent is copied to the I/O memory a sfence (store fence) operation is executed to force the wc-buffer to be flushed. When the hardware receives a send request with associated data, it starts assembling a network packet, which in turn is send through the network- and linkports of the design. The network port translates the 64-bit wide internal data format to a 18-bit wide network format, that is also to be used for an integrated switch. The linkport is responsible for the link-layer including retransmission and link management. On the receiving side, the network packet is first received by the receiver part of a linkport which then forwards the message to a network port and finally to a completer unit. This completer unit writes the message payload into a DMA buffer in host memory. The buffer is physically continuos and is allocated by system software. In addition a status line is also written back to the memory. The host processor can poll on this status line to be informed about newly arrived messages. This method offers the lowest possible latency, in particular when compared to interrupt driven mechanisms. Again the experiment was conducted with an 8-bit wide Hypertransport interface with 200MHz clock, i.e. 400 Megatransfers/s.

and an internal clock frequency of 100 MHz for all other modules. As physical link, a parallel LVDS cable connected to the ATOLL port of the HTX board has been used in a loop-back configuration. The end-to-end latency for a send operation directly followed by a receive operation was measured. For this configuration 16 byte is the smallest possible message size; a (hardware) message of this size is usable to carry a 0-byte sized MPI message, i.e. the necessary message header. The results are very encouraging, latency in this configuration starts with less than 750 ns.

TABLE IV
WRITE BANDWIDTH

Transfer Size (Byte)	Bandwidth (MBytes/s)
4-Byte	95 MB/s
8-Byte	189 MB/s
16-Byte	253 MB/s
32-Byte	303MB/s
64-Byte	337 MB/s

TABLE V
READ LATENCY

Transfer Size(Byte)	Latency (ns)
4-Byte	320 ns
8-Byte	620 ns
64-Byte	4900 ns

6 Conclusion and Outlook

The results presented in Section 5 are very promising and already very good for a reconfigurable hardware platform. To implement a "real" network several components are still missing, in particular the serial links and a switch. The next step will be to use serial links for data transport, which will introduce some additional latency for serialization, deserialization, encoding and decoding etc. Then, a crossbar, which has recently been developed at our group, is to be introduced between the networkport and the linkports connected to the different links, enabling a switched network. Again the crossbar is going to introduce several cycles of latency. On the other hand, the migration to a 16-bit wide Hypertransport core will cut the latency used for the Hypertransport transactions by half. An increase of the internal frequency to 200MHz in conjunction with a 400 MHz Hypertransport link clock would cut the complete HW latency in half (assuming that no additional pipeline stages have to be introduced). On the software side an integrtion with OpenMPI is planned.

This paper presented the HTX board and the HT core IP as a platform for message passing prototyping. As the experiments show, it is a comprehensive prototyping station well suited for network prototyping. The performance reached by the first implementation is already remarkable. With careful optimization, these numbers could even be improved. A word of caution is appropriate here, though. The used FPGA has limited resources in terms of capacity; and the more logic is to be implemented in it, the more difficult it becomes to reach timing goals. Using the larger FX-100 version of the chip may help here in some situations.

References

[1] *Top500 Supercomputer Sites*, www.top500.org, November 2006.

[2] *Myricom: A Gigabit-per-Second Local Area Network*, IEEE Micro, 1995.

[3] *QsNet: The Quadrics Network*, Quadrics Supercomputer Ltd., Jan 1999.

[4] *Mellanox*, Website: http://www.mellanox.com.

[5] *Volataire*, Website: http://www.voltaire.com.

[6] Hypertransport Technology Consortium, *Hypertransport I/O Link Specification Revision 2.00b*, Document #HTC20031217-0036-0009, 2005.

[7] *Pathscale*, Website: http://www.pathscale.com.

[8] David Slogsnat, Ulrich Bruening, Alexander Giese, *A Versatile, Low Latency HyperTransport Core*, to be published: FPGA'07, 2007, Monterey.

[9] Hypertransport Technology Consortium, *The Future of High Performance Computing: Direct Low Latency Peripheral-to-CPU Connections*, www.hypertransport.org, 2005.

[10] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard*, www.mpi-forum.org, 1994.

[11] Edgar Gabriel, et al., *Open MPI: Goals, Concept, and Design of a Next Generation MPI Implementation*, Euro PVM/MPI, 2004.

[12] Holger Fröning, Mondrian Nüssle, David Slogsnat, Heiner Litz, Ulrich Brüning, *The HTX-Board: A Rapid Prototyping Station*, 3rd annual FPGAworld Conference, 2006, Stockholm.

[13] Hypertransport Technology Consortium, *Hypertransport I/O Link Specification Revision 3.00*, Document #HTC20051222-0046-0008, 2006.

[14] *Center of Excellence on research in HyperTransport technology*, Website: http://www.ra.informatik.uni-mannheim.de/coeht/.

[15] *Xilinx: Virtex-4 User Guide*, Xilinx, 2007.

[16] *Small Form-factor Pluggable (SFP) Transceiver Multi-Source Agreement (MSA), 2000*, 2000.

[17] *PowerPC 405 Processor Block Reference Guide - Embedded Development Kit*, Xilinx, 2005.

[18] Jörg Kluge, Ulrich Brüning, Markus Fischer, Lars Rzymianowicz, Patrick Schulz and Mathias Waack, *ATOLL - A Next Generation System Area Network*, HPC Asia 2000, 2000, Bejing.

[19] Mario Trams, Wolfgang Rehm, *SCI Transaction Management in our FPGA-based PCI-SCI Bridge*, Proceedings of SCI Europe, 2001, Dublin.

[20] Nicolas Fugier, Marc Herbert, Eric Lemoine and Bernard Tourancheau, *MPI for the Clint Gb/s Interconnect*, Lecture Notes in Computer Science,Volume 2840/2003, Recent Advances in Parallel Virtual Machine and Message Passing Interface, 2003.

[21] Manuel Saldana and Paul Chow, *TMD-MPI: An MPI implementation for multiple processors across multiple FPGAs*, IEEE 16th International Conference on Field Programmable Logic and Applications, 2006, Madrid.

[22] *LinuxBIOS*, Website: http://www.linuxbios.org.