An open-source HyperTransport IP-Core

David Slogsnat
University of Heidelberg, Germany

Alexander Giese
University of Heidelberg, Germany

Mondrian Nüssle
University of Heidelberg, Germany

Ulrich Brüning
University of Heidelberg, Germany

Executive Summary

This white paper presents an open-source HyperTransport Intellectual-Property (IP) core optimized for low-latency and suitable for implementation both on reconfigurable hardware and ASIC technology. The core has been verified by simulation and in system using an FPGA. This exhaustive verification and the generic design allow the mapping to both ASICs and FPGAs. The in-system verification has been performed using a custom FPGA board that has been plugged into a HyperTransport extension connector (HTX) of a standard Opteron-based motherboard. The implementation described in this work supports a 16-bit link width, as used by Opteron processors.
On a Xilinx Virtex-4 FX FPGA, the core supports a link frequency of 400 MHz DDR and offers a maximum bidirectional bandwidth of 3.2GB/s. Performance analysis shows a unidirectional payload bandwidth of 1.4GB/s and a read latency of 180 ns.
The HT core is a proven, efficient technology, which in combination with the HTX board is an ideal base for prototyping systems and implementing FPGA coprocessors. The HT core is available as open source[1].

Introduction

Today, systems are most often built from Commercial-Off-The-Shelf (COTS) components. While networks and network interface controllers are very mature and highly optimized one serious weakness remains: computing and communication resources are usually only loosely coupled over a hierarchy of buses. Often, the bottleneck is the high-latency, non-cache coherent I/O bus, of which PCI-X and PCI-Express are the two most prominent representatives. This is the reason that a large fraction of the end-to-end message latency in modern high-performance networks is introduced by the I/O bus. Latency, however, is one of the most critical performance criteria of such high performance networks. To overcome this limitation, computation and communication resources have to move closer together.
AMD Opteron processors offer a unique opportunity for this through their HyperTransport (HT) interfaces. The adoption of the HT expansion connector (HTX) and motherboards that are equipped with this connector now make it possible to directly connect devices to an Opteron processor. This

[1] . The HT core is available for download from the **Center of Excellence for HyperTransport**. URL: http:// http://ra.ziti.uni-heidelberg.de/coeht/?page=home

results in a major increase of performance. Any device with high bandwidth or low latency requirements, such as FPGA coprocessors, will gain from the increased performance.
In this paper, we will describe such an FPGA implementation of a fully functional HT core which also maps to ASIC technology.

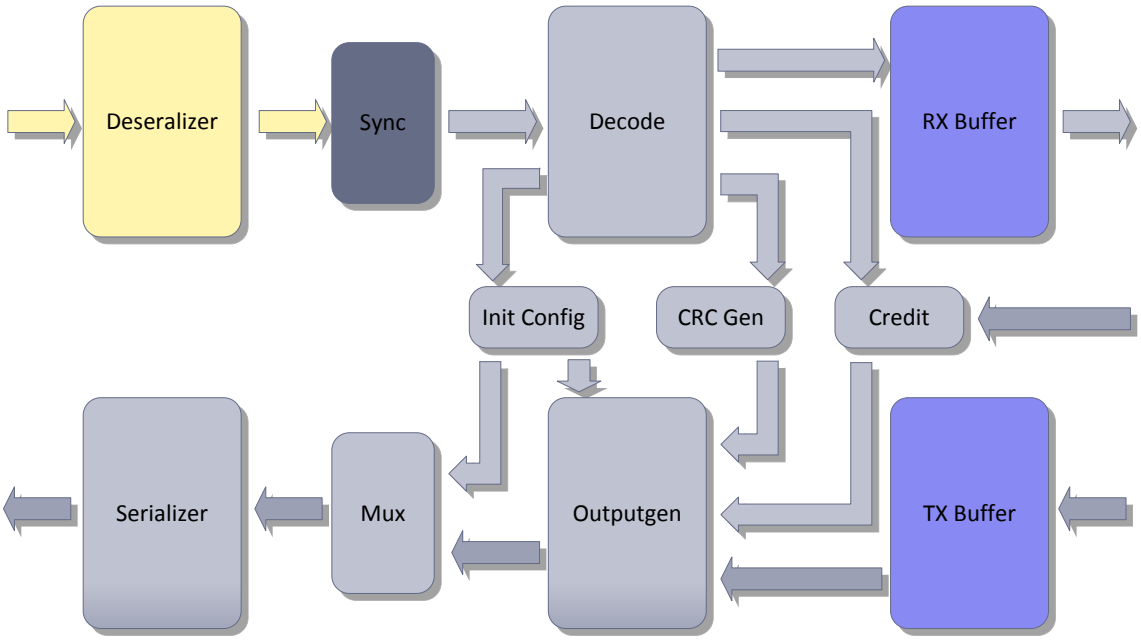

Figure 1: HT core architecture

HT Core Architecture

The architecture of the HT core is depicted in Figure 1. On the left side the HT link is shown, on the right side the application interface, which allows FPGA designs to access the HT core. This interface consists of three queues in each direction, one for every virtual channel.
The core utilizes three clock domains. Data on an HT link is transmitted at a high clock rate and needs the exact alignment with the related clock. Data is transmitted source synchronous. To handle the different clock domains a synchronization module is needed.
The HT core consists of 10 major sub modules, which are briefly described in the following paragraphs. At the interface to the HTX connector, there are deserializers at the incoming link and the respective serializers at the outgoing link. Both are configured to work with a parallelization degree of 4, so the internal data width of the core is four times higher than the link width. Behind the deserializers the data path has a width of 32-bit for each of the 8-bit CAD lanes.
Still within the link clock domains, the CRCs are checked. In HT 2.0, CRCs are periodically sent on each link. They contain the checksum of all packets on the bus within the respective timing window of 512 bit times. The Sync FIFO is used to synchronize the stream into the core clock domain.

The *decode* module distinguishes the different HT packets and generates the control signals for the virtual *rx buffers*, the *credit* module and the *init_config* module and shifts the assembled packets into one of these destinations.
Packets from the user application are placed in the *tx buffer*. The *outputgen* module has the task to arbitrate between the virtual channels and any other packets that have to be sent by the core. It performs arbitration based on priority to ensure that packets to be sent immediately are not delayed. While one packet is processed it may not be disturbed by packets other than the CRC packets. The *credit* module also keeps track of the remote credits to let the *outputgen* know on which virtual channel packets may be sent.
The *init_config* module is used for the low level initialization. Low level initialization has to set up the link after power-up and every time a link width or frequency change has taken place. Additionally, the HT configuration registers are located here.

Implementation & Verification

Figure 2: University of Mannheim (now Heidelberg) HTX-Board

For the implementation, the University of Heidelberg's HTX Reference Board[2] has been chosen. It consists of a PCB equipped with a Xilinx Virtex-4 FX series FPGA, six *Small Form Factor Pluggable (SFP)* optical transceivers and several other components such as DDR2 DRAM. This board is perfectly suited for networking applications.
While most of the core is written in portable Verilog HDL, some special FPGA macro blocks are used. This includes digital clock managers (DCMs), clock multiplexers, regional and I/O clock networks. FPGA embedded Block-RAMs and shift register look-up-tables (SRL) are used to efficiently implement queues and FIFOs and to save look-up-tables (LUT) resources. To serialize/deserialize the HT data the specialized ISERDES and OSERDES macro-blocks of Virtex-4 devices are used. Table I summarizes the resource usage of the HT core on a Virtex-4 FPGA.
These numbers show that 80% of the logic slices on a Virtex-4 FX60 are free for application circuits.

[2] More information and datasheets from http://ra.ziti.uni-heidelberg.de/index.php?page=projects&id=htx

	absolute	relative
Logic Slices	5222	20 %
LUTs	6371	12 %
Flipflops	2781	5 %
RAMs	33	14 %
DCMs	3	25 %

Table I: Resource requirements in a Virtex-4 FX FPGA

The hardware latency of the core is very low. Table II shows the number of pipeline stages for the incoming and the outgoing data path. The corresponding absolute timings are also given.

	Pipeline stages	Latency HT200 [ns]	Latency HT400 [ns]
Incoming path	12	120	60
Outgoing path	6	60	30

Table 2: Hardware latency of the HT core

The process of verification can be distinguished into three different phases. In the first phase, the HT core has been verified by simulation using low level directed tests, partly using a HyperTransport bus functional model (BFM) that is available from the HyperTransport Consortium. We observed that a simulation of the HT core with the bus functional model is in the order of 10,000 times slower than the FPGA system. In the second phase, the core has been mapped to the FPGA and verified in the system. The objective of this phase was to verify the low-level initialization sequence and the configuration phase, in which the BIOS configures the device using PCI compatible configuration cycles, and the application behavior. To be able to monitor the transmissions on the link, an HTX extender board is used. This board enables the user to connect direct probes of a logic state analyzer, which is a very efficient method for low-level debugging. More complex problems and bugs that appeared during in-system verification have been re-enacted in the simulation environment. There, the problems have been traced, analyzed, fixed and verified by simulation before testing it in-system again.

Results

To assess the performance of the HT core it has been tested with the *HT Example design*[3] enabling PIO, DMA and interrupt tests. The board was plugged into two systems to evaluate the performance of the HT core:

- Iwill DK8-HTX mainboard with two Opteron 870 (dual-core, 2 GHz, socket 940), 2 GByte RAM
- HP DL-145 G3 1U server with two Opteron 2218HE (dual-core, 2.6 GHz, socket F), 4 GByte RAM

The HT core was configured to run with 16-bit HT200 and 16-bit HT400 links. This corresponds with internal frequencies of 100 and 200 MHz. The core and the test design were driven by the same clock, no synchronizing FIFOs were implemented between core and test design.
Programmed I/O (PIO), i.e. load and store instructions from the host CPU, were able to reach up to 574 MB/s using 64-bit store operations. This is the equivalent of 50 million transactions per second. Using the write-combining feature of the CPU, up 1217 MB/s were possible. PIO read performance is considerably slower, since the tested Opteron CPUs support only 32-bit, strictly sequential load operations on I/O memory regions. Thus, a PIO read bandwidth of 20 MB/s, the equivalent of 5 million transactions per second, was measured.
The roundtrip latency, as it occurs for a PIO read, has been measured on both systems and the results are plotted in figure 3. So, the HT core reaches a best-case read-latency of 180 ns. For reference, we measured the read latency of two different FPGA based PCI-Express cards with x4 links and got a result of ~1100 ns respectively ~1200 ns (using the PCI-Express soft-core from the respective FPGA vendor).
All of the PIO measurements have been performed single-threaded. In a multithreaded environment one can expect higher aggregate performance numbers in the PIO read case.

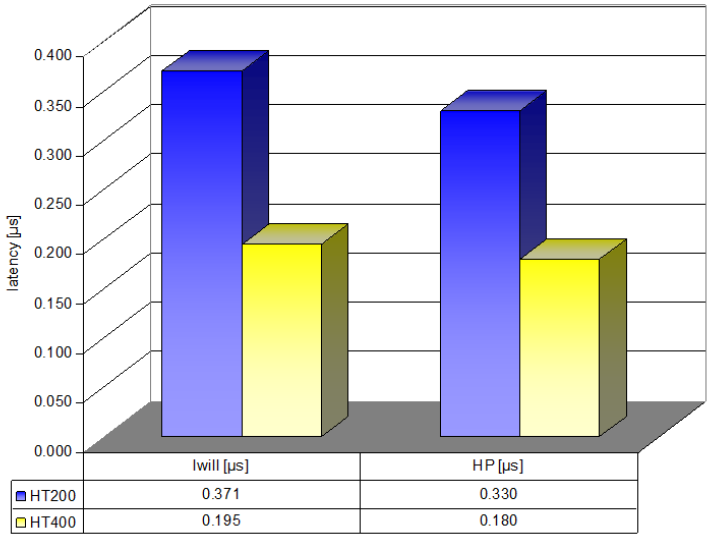

Figure 3: PIO Latency

[3] . The HT Example design is available for download from the **Center of Excellence for HyperTransport**. URL: http://ra.ziti.uni-heidelberg.de/coeht/?page=home

To test DMA performance the DMA engine of the *HT Example design* was programmed from software to perform transfers of different sizes, from 8 bytes to 8192 bytes. Transfers larger than 64 bytes trigger a sequence of 64-byte packets. The internal performance counter of the *HT Example design* was used to measure the time it takes to complete the transfers. This timer records the number of cycles from the start of the DMA job until the last word has been handed off to the core. An interrupt is then generated.
A theoretical analysis of the possible bandwidth of HT shows, that the peak link bandwidth of (*2 ·link clock frequency · ⅛ · bit width*) Bytes/s is diminished by CRC and header transmission. After CRC transmission, 99.22% of the link bandwidth remains for requests. For maximum sized packets 88.2% of the theoretical link bandwidth can be used for payload, for quadword sized (64-bit) packets this corresponds to $1411 \cdot 10^6$ Bytes/s reachable on a 16-bit HT400 link.

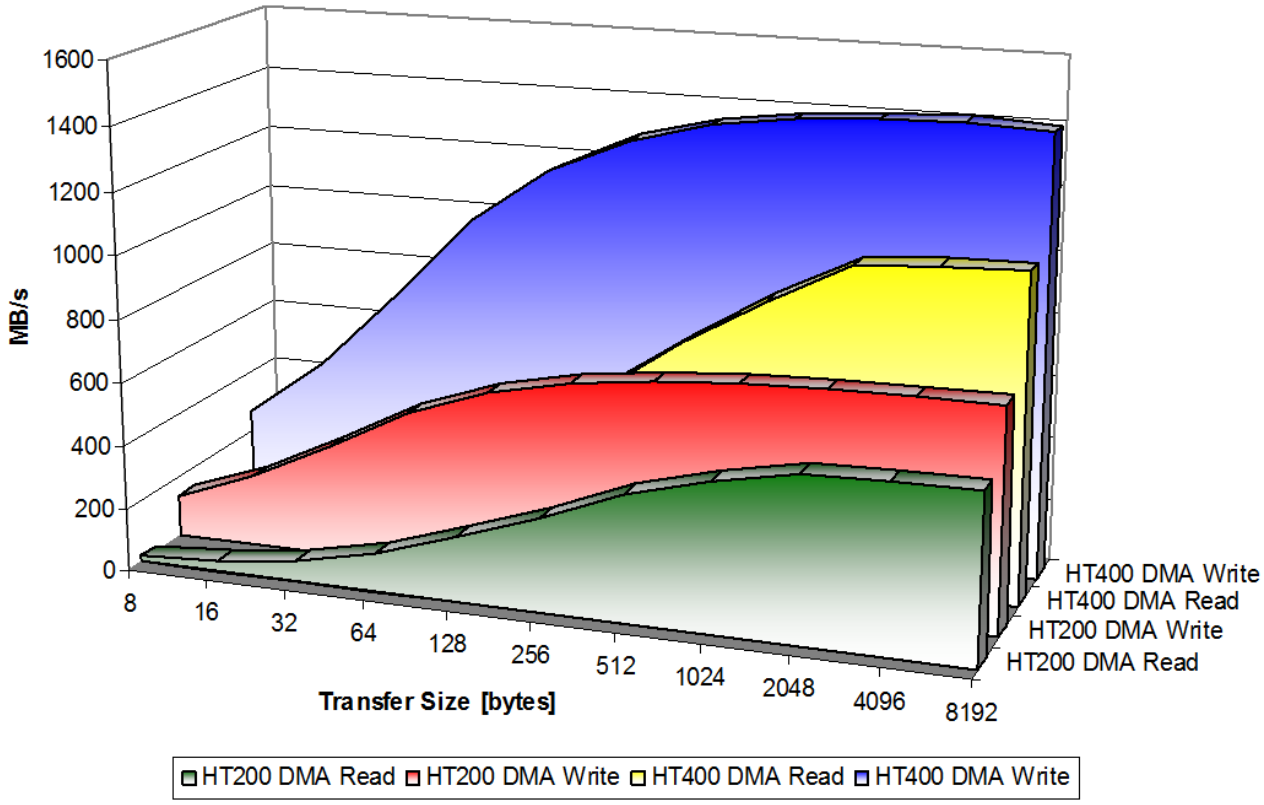

Figure 4: DMA Bandwidth

Figure 4 shows the results of the DMA transfer measurements performed. The HT400 write bandwidth starts at 267 Mbytes (8 byte transfer), while the read bandwidth starts at 29 MBytes/s. At 64 bytes, the maximum size of a single HT data size, write bandwidth reaches 984 MBytes/s and the read bandwidth reaches 216 MBytes/s. For large transfers, write bandwidth reaches up to 1410 MBytes/s (which is the theoretical limit) and the read bandwidth goes up to 1040 MBytes/s. HT200 shows roughly half the performance of HT400.
Finally, to measure interrupt latency, the FPGA issues an HT MSI packet and starts its internal timer. The interrupt handler function of the *HT Example* driver is triggered by the reception of the interrupt and reads the timer within the FPGA. The read-back time defines the interrupt latency. The results

show average interrupt latencies of 0.78 µs on the Iwill system and of 0.63 µs on the HP system (both HT400). This difference is caused solely by the two different CPUs.

Conclusion and Outlook

In this paper an efficient implementation of an HT 2.0b core is presented. The IP core successfully exploits the potential of the used FPGA in terms of bandwidth, latency and resource utilization. Offering an HT400 connection, the HT core can be used for more than just prototyping. Performance analysis shows that the HT core delivers impressive bandwidth. DMA transfers over 1.4 GByte/s of uni-directional payload bandwidth. PIO delivers over 1.2 GByte/s of uni-directional write bandwidth. Reading from the device exhibits a very low latency of 180 ns. With this performance the HT core can serve as link to the host CPU for production coprocessors. With the given characteristics, the HT core remains an interesting technological option for new projects.
The link clock frequency scales with the maximum clock speed of the FPGAs. Thus, faster links will be possible with faster FPGA families such as the Virtex-5 family. The HT core can also be mapped to an ASIC implementation by replacing FPGA specifics as DCMs, SERDES and RAM blocks with the corresponding ASIC macros. ASIC experience shows, that the core is easily able to reach HT800 and above with current standard cell libraries.
The core has also been successfully ported to other FPGA families and we continue to maintain the core as an open-source, downloadable solution for interested parties. A more detailed description and analysis of the HT core can be found in *An Open-Source HyperTransport Core* in ACM Transactions on Reconfigurable Technology and Systems (TRETS), Vol. 1, Issue 3, p. 1-21, Sept. 2008.

Acknowledgements

We want to thank the HyperTransport Consortium, Cadence Design Systems Inc. and Advanced Micro Devices Inc. for supporting our research work. In various ways we have received great help from them.