Hardware and Software Design of FPGA-based PCIe Gen3 interface for APEnet+ network interconnect system

R. Ammendola¹, A. Biagioni², O. Frezza², F. Lo Cicero², A. Lonardo², M. Martinelli², P. S. Paolucci², E. Pastorelli², D. Rossetti³, F. Simula², L. Tosoratto², P. Vicini²

¹ INFN, Sezione di Roma Tor Vergata, Italy
² INFN, Sezione di Roma, Italy
³ NVIDIA Corp, Santa Clara, California

E-mail: michele.martinelli@roma1.infn.it

Abstract. In the attempt to develop an interconnection architecture optimized for hybrid HPC systems dedicated to scientific computing, we designed APEnet+, a point-to-point, low-latency and high-performance network controller supporting 6 fully bidirectional off-board links over a 3D torus topology.

The first release of APEnet+ (named V4) was a board based on a 40 nm Altera FPGA, integrating 6 channels at 34 Gbps of raw bandwidth per direction and a PCIe Gen2 x8 host interface. It has been the first-of-its-kind device to implement an RDMA protocol to directly read/write data from/to Fermi and Kepler NVIDIA GPUs using NVIDIA peer-to-peer and GPUDirect RDMA protocols, obtaining real zero-copy GPU-to-GPU transfers over the network.

The latest generation of APEnet+ systems (now named V5) implements a PCIe Gen3 x8 host interface on a 28 nm Altera Stratix V FPGA, with multi-standard fast transceivers (up to 14.4 Gbps) and an increased amount of configurable internal resources and hardware IP cores to support main interconnection standard protocols.

Herein we present the APEnet+ V5 architecture, the status of its hardware and its system software design. Both its Linux Device Driver and the low-level libraries have been redeveloped to support the PCIe Gen3 protocol, introducing optimizations and solutions based on hardware/software co-design.

1. Introduction

The APEnet+ project was born to equip COTS-based clusters with a point-to-point interconnect adapter, derived from custom designed HPC systems optimized for High Energy Physics simulations (HEP), supporting a 3D toroidal network mesh with high bandwidth and low latency. APEnet+ has now specific support for GPUs for use in heterogeneous clusters; in this capacity, its usage in HPC simulations has tackled simulations as diverse as HEP or Neural Networks, even spawning a sibling device employed as a low latency, real-time interface between GPUs and DACs or trigger systems in particle detectors.

In the hybrid clusters, low latency and high bandwidth are key factors for overall efficiency and scalability: to improve on them, APEnet+ V5 employs the Remote DMA paradigm (RDMA), removing the bottleneck of the operating system managing the network transfers and...
implementing the so-called zero-copy. Following the latest enhancements in FPGA technology and PCIe standard, we redesigned the APEnet+ V4 board to support the latest PCIe Gen3 revision onto a state-of-the-art Altera FPGA. This new card, implemented on a DK-DEV-5SGXEA7N Development Board with a 28 nm technology FPGA, is called APEnet+ V5; it is equipped with HSMC — for an eventual external daughterboard for future expansion —, optical cage QSFP and Ethernet PHY connectors (10/100/1000Mbps). In this paper we describe the hardware and software enhancements made possible by the new embedded transceivers and the larger amount of programmable resources at our disposal and the improved performances in terms of latency and bandwidth. Among the upgrades provided by the new platform, we mention that the PCIe Gen3 x8 IP is used in combination with a 3rd party core, controlled by an AXI bus; this gives us the option for future implementation of a full-featured ARM processor, which is slated to be one of the most significant additions to the next generation of FPGA platforms.

The remainder of this paper is organized as follows: after an outlook of the state of the art in section 2, we describe the hardware design of APEnet+ V5 in 3, with an overview of the architecture and improvements like PCIe Gen3 compliancy and the new Altera Stratix V FPGA features. Section 4 is about the software architecture; our test results are shown in section 5 and conclusion and future works are described in 6.

2. APEnet+ and related work

Our most recent work is [1], where we presented the preliminary results obtained for APEnet+ V5 with a description of the hardware and software improvements that led us into achieving better performance in terms of latency and bandwidth. The GPU communication support that was the highlight of the previous version of APEnet+ V4 has now been extended to work also on the APEnet+ V5 revision through NVIDIA GPUDirect [2] RDMA. Nowadays, many HPC platforms that deploy custom network subsystems to accommodate the use of accelerators can be found; the Gemini interconnect [3], used in the Cray XK hybrid platforms, or the TH2 Express interconnect [4], used in the Tianhe-2 supercomputer which currently holds the first place in the TOP500 [5], are such examples. The PCI Express Adaptive Communication Hub is a project which in its latest version PEACH3 [6] also uses an Altera Stratix V FPGA together with a PCIe Gen3 host interface to provide a PCIe ring based network; it focuses more on being dedicated to tightly coupled accelerators (TCA) than to provide a general purpose, modular NIC. Another approach relies on directly attaching the accelerators within a cluster to the network fabric and treating them on the same footing as ordinary nodes; overcoming the problems of how to enumerate, configure and operate the accelerators over the network is the original contribution of the EXTOLL NIC [7].

Whatever task they are put to — as a networking system component like in our case or as a computing accelerator in itself — FPGA-based devices also need high throughput towards their own hosts, often requiring non-trivial software infrastructure. In [8] an open source framework is presented enabling easy integration of GPU and FPGA resources providing direct data transfer between the two platforms with minimal CPU coordination at high data rate and low latency. In [9] it is described an efficient and flexible host-FPGA PCIe communication library whose presented results were obtained on PCIe Gen2 Xilinx FGPAIs.

3. APEnet+ V5 hardware

Fig. 1 shows the APEnet+ V5 architecture; it is split in three main blocks: Network Interface, Router and Torus Link.

- **Network Interface** is in charge of managing the data flow between PCIe bus and APEnet+: on the TX side it gathers data coming from the PCIe port, generates packets and forwards them to the **Router** block, while on the RX side it provides hardware support for the
RDMA programming model. The APEnet+ packets have 256 bits-wide header, footer and data words with variable size payload. Network Interface uses a 3rd party device, a Gen3-compliant soft core by PLDA (see section 3.1).

- **Router** manages dynamic links between APEnet+ ports: it inspects the packet header and translates its destination address into a proper path across the switch according to a dimension-ordered routing algorithm, managing conflicts on shared resources.
- **Torus Link** allows point-to-point, full-duplex connections of each node with its neighbours, encapsulating the APEnet+ packets into a light, low-level, word-stuffing protocol able to detect transmission errors via CRC. It implements 2 virtual channels and proper flow-control logic on each RX link block to guarantee deadlock-free operations (see section 3.2).

![Figure 1. Outline of the main logic blocks of APEnet+ architecture.](image1)

APEnet+ V5 was developed on the Altera DK-DEV-5SGXEA7N development kit (see fig. 2), a complete design environment that features a 28 nm FPGA (*Stratix V GX 5SGXEA7K2F40C2N*). This device offers a PCIe connector, which supports the connection speed of Gen3 at 8.0 Gbps/lane with the host machine and supplies power to the board, a 40G QSFP connector and two HSMC ports which allow the interconnection between boards.

Thanks to the modularity of its architecture, a first implementation of APEnet+ V5 with only three links was developed in a straightforward way.

### 3.1. PCIe Gen3 Interface

PCIe Gen3 is based on 8.0 Gbps lanes using a 128b/130b encoding, therefore the protocol overhead is reduced to less than 1% from ~20% for previous PCIe generations. The upgrade to Gen3 for the host interface allows, with 8 bonded lanes, a total raw bandwidth around 8GT/s. To keep the Network Interface clock frequency at 250MHz, the back-end datapath must be widened to 256-bits.

To simplify the interface to the PCIe Hard IP we used a third party device, the PLDA XpressRICH Gen3 soft IP core. The interface of this core with the backend is an AXI4 bus as depicted in fig. 3. Inclusion of AXI4 inside our architecture is a useful preparatory exercise for...
future use of embedded ARM hard IP processors, foreseen only on high-end FPGAs like the future Altera 10 device family.

The XpressRICH core consists of 3 different layers: the PCIe layer, the Bridge layer and the AXI layer. The PCI Express layer contains a PCIe Controller and the interfaces with the Bridge layer i.e. in order to access the PCIe Config Space and manage the low power states and interrupt signals for the IP. The Bridge layer interconnects and arbitrates between input and output flows, implementing up to 8 independent DMA Engine modules and translating between the AXI and PCIe interfaces.

In the V4 revision, doubling the DMA engine on the TX side — the logic of the Network Interface loading data from host onto the FPGA — earned all by itself a ∼40% increase in bandwidth; in the same way, a significant boost was brought about for APEnet+ V5 by introducing a TX side double DMA channel; a comparison between the finalized revision and a single engine prototype (described here [10]) is shown in fig. 9.

The DMA IF has the task of programming the DMA channels by means of the AXI Memory mapped interface. A state machine collects the requests coming from APEnet+ V5 and programs dedicated registers according to a Round Robin algorithm to perform PCIe transactions.

Figure 3. Design of PCIe Gen3 interface, based on AXI4 protocol.  
Figure 4. APEnet+ V5 Torus Link architecture.

3.2. Enhanced embedded transceiver

FPGAs of the Stratix V GX family feature full-duplex transceivers with data rates from 600 Mbps to 12.5 Gbps, offering several programmable and adaptive equalization features. The Torus Link block Physical Layer is made up of an Altera Custom IP Core (with the corresponding reconfiguration block) and a proprietary channel control logic (Sync_ctrl block), as shown in fig. 4. The Altera Custom IP Core is a generic PHY which can be customized in order to meet design requirements. The receiving side consists of a Word Aligner, 8b/10b decoder, Byte Ordering Block and RX phase compensation FIFO. Similarly, on the transmitter side, each transceiver includes a TX phase compensation FIFO, an 8b/10b encoder and a Serializer. The Word Aligner restores the word boundary based on an alignment pattern that must be received during link synchronization. A status register asserted by the Altera Avalon interface triggers the Word Aligner to look for the word alignment pattern in the received data stream. The Byte Ordering block looks for the byte ordering pattern in the parallel data: if it finds the byte ordering pattern in the MSB position of the data it inserts pad bytes to push the byte ordering pattern to the LSBByte(s) position. Finally, the RX phase compensation FIFO compensates for the phase difference between the parallel receiver clock and the FPGA fabric clock. The write
and read enable signal of the Deskew FIFOs are managed by the Sync_ctrl block: the write signals are asserted for each lane after recognition of 8b/10b keyword /K28.3/, while the read signal, common to all FIFOs, is asserted when all FIFOs are no longer empty. Altera Transceiver Reconfiguration Controller dynamically reconfigures analog settings in Stratix V devices: it is able to compensate for variations due to process, voltage and temperature in 28 nm devices.

| Cable                    | BER       | Data Rate |
|--------------------------|-----------|-----------|
| 10 m Mellanox optical cable | < 2.36 E-14 | 11.3 Gbps |
| 1 m Mellanox copper cable  | < 1.10 E-13 | 10.0 Gbps |

Table 1. APEnet+ BER measurements on Altera 28 nm FPGA.

In conclusion, three bi-directional data channels could be implemented on the development board. The X channel was implemented using the 4 lanes of the QSFP connector; the Y and Z channels were implemented onto the HSMC interface. Very preliminary BER measurements were performed on the X channel of APEnet+ V5 (see table 1). The testbed consists of two Stratix V FPGAs connected by InfiniBand cables with different lengths and support media (optical and copper). Copper wires results derive from experiments conducted with 10 Gbps-certified cables; they are expected to improve as soon as they are repeated with commercially available 14 Gbps-certified ones.

4. APEnet+ V5 Software Design

Following the scheme in fig. 5, let us assume that the user space process wants to send data from a TX user buffer to a certain RX user buffer, which can reside in the CPU/GPU memory of the same host or in that of a different node in the network.

Figure 5. Software Architecture of APEnet+ V5 NIC.

To start the TX part of the communication, the user process invokes the send request primitive — provided by the user space library (see fig. 6) — which calls a function in the device driver that translates the request into one or more descriptors in the so-called TX queue. These latter structures contain all the relevant information to program the APEnet+ V5 data transfer, i.e. the physical address of the source buffer in memory and the virtual address of the destination memory buffer.

The device driver notifies APEnet+ V5 by writing a PCI register of the board about the presence of one or more descriptors that are ready to be DMA-read and processed. The mechanism of descriptors is necessary because with the virtual memory paradigm a data buffer
is typically split into a set of non-contiguous memory pages. The idea is to handle every memory page with a different descriptor, each of which can move up to $\text{PAGE\_SIZE}$ data (usually 4kB).

When the board has processed a send request, (i.e. the board has DMA-read the data in memory) a $\text{TX\_done}$ completion is generated by the APEnet+ V5 hardware and DMA-written into the host memory. The presence of this completion is then notified to the device driver, which pushes it in a completions queue, ready to be grabbed by the user process that spins in wait for events status. When a new inbound packet lands, it brings along the destination virtual memory address the sender must have signed it with. This virtual address is used by the Nios II microcontroller to browse an index of preregistered buffers — the so-called BSRC (Buffer SeaRCh) process — to find the correct memory area where the packet is to be received. The buffer list is populated during a user space process setup phase by means of the register buffer primitive, triggering a device driver function call that instructs the onboard Nios II. If the buffer is found, the associated information are used to retrieve the physical address in the so-called V2P (Virtual To Physical) translation. To hasten the BSRC and the V2P processes, a hardware Translation Look-aside Buffer (TLB) is employed [11]. Once the physical address has been retrieved, the board can instantiate a DMA-write of the data into memory.

When data has been DMA-written in memory, a $\text{RX\_done}$ completion is generated and DMA-written in memory as well and notified to the device driver. A mechanism on the receiving side of the communication, similar to one on the transmission side, informs the user that the board has DMA-written a buffer and data is ready to be read: the user starts waiting in a wait event blocking call, which polls for the so-called $\text{RX\_done}$ completion.

5. Test results
All development, testing and debugging were performed on the Test and Development (T&D) platform which is composed by two X9DRG-QF SuperMicro servers (its dual sockets populated with Sandy Bridge E5-2609@2.40 GHz CPUs) hosting the APEnet+ V5 IP on the DK-DEV-5SGXE7N Development Board and NVIDIA Tesla K20Xm GPUs; debugging was aided by a Tektronix TLA-7012 Gen3 Logic Analyzer. Given the Sandy Bridge platform limitations, performance tests were instead conducted on an Ivy Bridge-supporting X9DRG-HF server equipped with Intel® Xeon® CPUs (E5-2630 v2@2.60 GHz). Both systems use the NVIDIA GPU driver version 352.30 and CUDA 7.0. Performance was gauged on a single machine with one APEnet+ V5 closed in a loopback configuration — the switch is set to bounce egressing packets into ingressing ones. Latency is measured by sending a packet of a certain size and measuring the time required for the $\text{TX\_done}$ and $\text{RX\_done}$ completions to arrive; tests are repeated 100000 times and total time is averaged. In the bandwidth test, all packets are sent in at once and time required for all completions to land is measured. Timekeeping was done by using the Intel x86 internal cycles counter (Time Stamp Counter (TSC) register).

Test results are shown in fig. 7 for every combination of host/GPU memory and TX/RX modes. As can be seen, the GPU path is slower than the host one, which is probably due to the
DMA engines/IOMMU limitations while best performance is achieved for host-to-host memory transfers. The performance loss after 128kB is due to the size of the hardware TLB, that has a fixed size of 32 entries for the V2P cache: this means that after $32 \times 4kB$ there are only miss in cache. This effect is absent in the GPU RX plot because of the larger GPU memory page.

Measured latency for small packets (shown in fig. 8) is barely influenced by data size. We think this is due to the long programming time required by the PCIe core to instantiate a DMA transfer; a single transfer requires writing many registers, each write costing multiple hardware cycles. Measurement errors are mainly due to the priority of running processes at a certain time which influence especially the switch from user application to kernel space device driver.

![Figure 7. APEnet+ V5 bandwidth.](image1)

![Figure 8. APEnet+ V5 latency.](image2)

Inhibiting either the Nios II micro controller and the hardware TLB, another set of measurements was thus taken using physical addresses instead of virtual ones so as to have a better overview of the raw performances.

![Figure 9. host-to-host bandwidth measurement using physical addresses shows more clearly the improvements in performance using two parallel DMA engines (see section 3.1).](image3)

Finally, in fig. 10 we report the bandwidth when just taking into account the transmission part of the communication; since we achieve around 4.5 GB/s, which leaves us thinking that the current bottleneck is on the receiving side of communication.

6. Conclusion and future work
We have presented the software and hardware design for the new APEnet+ V5 board, showing encouraging results regarding expected performance gains with respect to the old APEnet+ Gen2 board version (see fig. 10 and fig. 11 for the bandwidth/latency comparison). As mentioned in
Figure 10. Host-to-host bandwidth in APEnet+ V4 and V5. Figure also shows V5 performance discarding the receiving part of the communication, showing headroom for a future improved design.

Figure 11. CPU-to-CPU APEnet+ latency. In V5 design, small packets latency is barely influenced by data size, mainly due to the Gen3 PCI Core.

3.1, doubling the DMA engine on the TX side managed to saturate the available bandwidth on APEnet+ V4 revision; we see that the same is apparently true, at least restricting on the TX side of APEnet+ V5. We are confident to be able to increase saturation for APEnet+ V5 PCIe channel once the RX side equally doubles its DMA engines so as to push up closer to the available bandwidth granted by the enhanced PCIe standard.

Acknowledgments
This work was partially supported by the EU Framework Programme 7 project EURETILE under grant number 247846 and by MIUR (Italy) through INFN SUMA project.

References
[1] Ammendola R, Biagioni A, Frezza O, Lonardo A, Lo Cicero F, Martinelli M, Paolucci P, Pastorelli E, Rossetti D, Simula F, Tosoratto L and Vicini P 2015 Journal of Instrumentation 10 C02005 URL http://stacks.iop.org/1748-0221/10/i=02/a=C02005
[2] NVIDIA GPUDirect technology https://developer.nvidia.com/gpudirect
[3] Alverson R, Roweth D and Kaplan L 2010 High Performance Interconnects (HOTI), 2010 IEEE 18th Annual Symposium on pp 83–87
[4] Pang Z, Xie M, Zhang J, Zheng Y, Wang G, Dong D and Suo G 2014 Frontiers of Computer Science 8 357–366 ISSN 2095-2228 URL http://dx.doi.org/10.1007/s11704-014-3500-9
[5] http://www.top500.org
[6] Kuhara T, Kaneda T, Hanawa T, Kodama Y, Boku T and Amano H 2014 Computing and Networking (CANDAR), 2014 Second International Symposium on pp 377–381
[7] Neuwirth S, Frey D, Nuesle M and Bruning U 2015 High Performance Computer Architecture (HPCA), 2015 IEEE 21st International Symposium on pp 627–638
[8] Thoma Y, Dassatti A and Molla D 2013 Reconfigurable Computing and FPGAs (ReConFig), 2013 International Conference on pp 1–6
[9] Gong J, Wang T, Chen J, Wu H, Ye F, Lu S and Cong J 2014 Field Programmable Logic and Applications (FPL), 2014 24th International Conference on pp 1–6
[10] Ammendola R, Biagioni A, Frezza O, Lo Cicero F, Lonardo A, Paolucci P, Pastorelli E, Rossetti D, Simula F, Tosoratto L and Vicini P 2013 Reconfigurable Computing and FPGAs (ReConFig), 2013 International Conference on pp 1–6
[11] Ammendola R, Biagioni A, Frezza O, Geurts W, Goossens G, Lo Cicero F, Lonardo A, Paolucci P S, Rossetti D, Simula F, Tosoratto L and Vicini P 2015 Future Generation Computer Systems, Elsevier ISSN 0167-739X URL http://www.sciencedirect.com/science/article/pii/S0167739X14002726