Network Processors for a 1-MHz Trigger-DAQ System

Artur Barczyk, Jean-Pierre Dufey, Beat Jost, and Niko Neufeld

Abstract—Network processors (NP), originally developed for applications in high-end network routers, have a great potential as building blocks for a high-speed networked data acquisition (DAQ)-Trigger system. After an introduction to NP, we will present several applications which illustrate their power in coping with high event rates and aggregate traffic of above 10 GB/s. All these applications can be implemented in a single, versatile, generic NP-based module, where the specific functionality is provided by the software. We will discuss the impact of protocol overheads on link utilization, and demonstrate a NP-based way of enhancing the portion of the bandwidth occupied by user data. One other application deals with sorting of trigger decision frames, and finally, we will look into the advantages of implementing a large switching network with the NP-based module.

Index Terms—Data acquisition, fast networks, interconnected systems, microprocessor applications, parallel processing.

I. INTRODUCTION

IN THE LHCb experiment [1], the selection of events is organized in three trigger levels. The Level-0 trigger relies on hardware devices to perform a first selection that reduces the event rate to 1 MHz. The upper layers are implemented in software and require the assembly of event data fragments over a switching network that interconnects some 300 data sources to a large farm of ~ 2000 processors. The Level-1 trigger requires only a subset of each event data, namely a few tens of bytes per source from about half of the sources, or a total of about 4 kB on average at a rate of 1 MHz. A strictly limited latency is imposed on the Level-1 response time, determined by the buffer size in the front-end modules where event data have to be kept until this decision arrives. The high-level trigger (HLT) operates at an event rate of 40 kHz and requires the full event data, i.e., 40 to 50 kB. Both software trigger levels must use the same switching network and the same processor farm. Thus, the network must provide an aggregated bandwidth of ~ 10 GB/s. Details on the trigger-DAQ architecture can be found in [2].

These requirements, the high aggregate bandwidth and the high packet rate per link put stringent constraints on the choice of a technology to implement the network. High reliability is required since a single fragment missing at the destination makes the complete event unusable. At the same time the system has to be affordable. Presently, gigabit Ethernet, as the new emerging

standard in network technology, is a good candidate for the implementation of the Trigger-DAQ system. This technology provides presently the highest bandwidth per link at a low cost. However, at a frame rate of 1 MHz this link bandwidth is just sufficient for our application and the poor link utilization factor (ratio of user data over complete frame size) can be a problem. Furthermore, the availability of commercial switches that ensure a very low packet loss probability at high link loads and for the specific type of traffic generated by our event building application is not guaranteed yet. Another possible implementation based on scalable coherent interface (SCI) technology has been proposed [3], but is not covered in this paper.

The aim of the paper is to show that the networks processors (NP) technology can provide solutions to overcome possible limitations of gigabit links in the framework of our Trigger-DAQ system. It is not a proof of feasibility of the full system. We present in this paper a study of the possible use of NP in three domains of our trigger-DAQ system. The first application is the implementation of a method to enhance the link utilization, while fulfilling the low latency requirements for the Level-1 event building. The second one is the implementation of the Level-1 trigger decision sorting, and a third proposes an architecture of the full event building network based entirely on NP.

In Section II, we discuss the implications on the link bandwidth utilization of the overheads due to standard transport protocols. Section III presents the NP concept, illustrated by a specific commercial solution. The method to enhance the link utilization and its implementation using NPs will be presented in Section IV. Section V describes the implementation of decision sorting, and finally the architecture of a full network based on NPs in presented in Section VI.

II. TRANSPORT PROTOCOLS

Protocols, structured in layers, are required to properly deliver data packets in a network. The task of event building is to assemble all fragments of an event and to build an event record with the correct structure. Switching networks can be used to perform event building in parallel to many destinations.

The “event building” protocol forms the upper layer. Each event fragment must be identified by the data source and event number. Furthermore some information on error conditions at the source of data is usually also present. In the following discussion, an event fragment is the data from a detector segment, encapsulated in the event building protocol format. An event fragment is generated and delivered by a front-end module.

To deliver an event fragment to the selected destination, we rely on standard protocol layers. The minimum is the Ethernet

Manuscript received June 6, 2003; revised November 26, 2003.

A. Barczyk, J.-P. Dufey, and B. Jost are with CERN, CH-1211 Geneva, Switzerland (e-mail: A.Barczyk@cern.ch; Jean-Pierre.Dufey@cern.ch; Beat.Jost@cern.ch).

N. Neufeld is with CERN, CH-1211 Geneva, Switzerland, and also with the Institut de Physique des Hautes Energies, Université de Lausanne, CH-1015 Lausanne, Switzerland (e-mail: Niko.Neufeld@cern.ch).

Digital Object Identifier 10.1109/TNS.2004.828637

Fig. 1. Ethernet frame structure.

protocol (“Layer 2” in the OSI model) on top of which higher level layers may be invoked in order to enhance the functionality and safety of data transport. We adopt here the terminology of the TCP/IP protocol stack which adds two “layers” on top of the basic “subnetwork” layer (Ethernet in our case): the Internet protocol (IP) layer (“Layer 3”), and, above it, either the transmission control protocol (TCP) or the user datagram protocol (UDP).

A. The Protocol Layers

1) *Ethernet Protocol:* Fig. 1 shows the format of an Ethernet frame as defined by the IEEE 802.3 z [4] (1000 Base-X Gigabit Ethernet). In the case of full duplex connections, there is no additional overhead due to interframe delays, the “preamble” being sufficient to play this role.

The Ethernet protocol provides source to destination addressing, bit error detection (Cyclic Redundancy Check, CRC) and a basic flow control via XON/XOFF on a point to point connection. This flow control mechanism is not necessarily ensured across a switch. The total overhead due to the Ethernet protocol amounts to 26 bytes. It is fixed, independent of the payload. The payload is at least 46 bytes and at maximum 1500 bytes.

2) *IP:* It is the basic protocol supporting the third level protocols, either TCP/IP or UDP/IP, in order to isolate them from the specific subnetwork layer. IP is a connectionless protocol that performs data segmentation and reassembly of the upper layer data in order to conform to the subnetwork maximum transmission unit (MTU). The maximum packet size supported is 64 kB. IP does not guarantee data delivery, which is the task of the upper layers if at all. A check-sum covers only the IP header data.

The additional IP protocol overhead is a header of 20 bytes. Hence, an IP packet shipped by a single Ethernet frame has a protocol overhead of 46 bytes (the 26 bytes of Ethernet and the 20 bytes for IP). The maximum user data in a single Ethernet frame is 1480 bytes. IP might be compulsory on switching networks specialized in IP routing which do not accept bare Ethernet frames (Layer 3 switches).

3) *UDP:* On top of the IP functionality, this protocol layer adds “port” addresses to permit concurrent transmissions between different applications. It also provides a check-sum covering the data. The additional overhead due to UDP is 8 bytes.

4) *TCP:* is an alternative to UDP that implements a full duplex virtual circuit between two “ports” (applications). It ensures reliable data transmission by means of check-sums and data retransmission in case of corrupted or lost data, requiring

a constant dialog between the partners in order to acknowledge and regulate the data transfer.

B. Discussion on Protocol Choice

The Ethernet protocol is entirely generated and managed by hardware, whereas the protocols above it, usually, are implemented in software. For the fast event builder we need, protocol generation occurs in the front-end modules, where it is only practical to generate basic Ethernet protocol. The simplest protocol above it, namely IP, requires the implementation of segmentation and the calculation of the header check-sum. Although at 1 MHz event rate, the frame size is limited to stay well under MTU (typically 1500 Bytes), we have to consider the possibility of data taking at lower rates, but with much larger data sizes. This occurs in calibration runs that generate large amounts of data which would not fit into one Ethernet frame and require fragmentation.

For the higher level protocols, the only advantage of UDP for our application is the check-sum covering the user data, but Ethernet already provides this facility. Finally, TCP is not compatible with the high rate of very small packets nor with a hardware implementation in the front-end modules.

We are then left with Ethernet and IP. The problem of reliable data transmission will be addressed by carefully dimensioning the switching network to avoid congestion conditions as much as possible.

C. Protocol Overheads and Link Occupancy

The maximum theoretical payload $p_{\max}$ [bytes] for a frequency of frames F [MHz], a load factor L ($0 < L < 1.0$) and a protocol overhead “ov” on a gigabit link, is given by

$$p_{\max} = \frac{10^3 \times L}{8 \times F} - \text{ov}.$$

We assume here that the preparation of a frame does not interrupt the transmission of previous frames. If this were the case, the overhead factor should be increased by an equivalent amount of bytes.

Fig. 2 shows the frame rate F as a function of the payload, assuming a load factor L of 1.0 (100% link occupancy) for the Ethernet protocol (curve a) and for IP (curve c). The percentage of link bandwidth devoted to user data is also shown (curve b for Ethernet and d for IP).

The potential rate of packets up to 46 bytes of user data is 1.7 MHz on a gigabit link with the Ethernet protocol. At 1 MHz, the user data payload is limited to 99 bytes for Ethernet and 79 bytes for the IP protocol. Those values correspond to a 100%

Fig. 2. Maximum frame rate for (curve a) Ethernet. Link occupancy by user data Ethernet (curve b). Maximum frame rate for (curve c) IP. IP (curve d) as function of user data size.

link occupancy while the user data occupy 80% and 63% of this bandwidth, respectively. Dashed continuation of the occupancy lines shows the region where frame size would fall below the minimum defined by the standard¹.

It may be possible to maintain a high load on a point to point link, thus permitting high values of payloads compatible with the event rate. However, at the input ports of the event builder switch, the load must be significantly lower than 100%, in order to avoid overflow in the network buffers. The exact number depends on the design and buffer space of the particular switch in question, and can be only given upon evaluation of the hardware, but a safety margin of 30% is probably sufficient for our traffic pattern, as indicated by simulation studies.

III. A VERSATILE NP- BASED BOARD

All data in an Ethernet system travel in frames. Event building and routing is then basically a matter of frame processing. High-speed frame processing is not very efficiently done using commercial, general purpose CPUs, because it is an interrupt driven activity and because it is mainly a question of I/O and not only of raw CPU power. This is where NP come into play. They have been developed specifically for packet processing and manipulation, and are designed to do packet processing in modern high-speed networks with packet rates of 5 MHz and above, on links which run at 1, 5, and even 10 Gigabit/s. The necessary processing power is achieved by using a dedicated architecture, the details of which vary greatly, but some general features apply to most NPs: they use many packet-engines in parallel, usually RISC processors with special instructions for frame manipulation (like, e.g., check-sum calculation). NP implement very high bandwidth memory interfaces, both for storing packets, routing tables and other forwarding information (filtering). They include hardware assists for many common tasks in networking,

¹Some devices, such as, e.g., the NP4GS3 allow to handle also under-size frames.

like tree lookup, traffic shaping etc. Most major chip manufacturers have an NP line of products and there are also many startup companies in this business. Extensive information can be found at the “Network Processor Central” webpage [5].

We have studied in detail the NP4GS3 NP from IBM [6]. It includes four gigabit Ethernet media access controllers (MAC). A fully functional module needs a few external components, notably memory, physical interfaces for the chosen gigabit Ethernet implementation (optical or copper), power and a control interface for booting and monitoring (e.g., PCI). The NP4GS3 provides two full-duplex high-speed links [data aligned serial link (DASL)] to connect to a switch fabric. This link can directly connect two NPs, driving a cable and connectors over distances of several tens of centimeters. Even without a switching fabric we can get a fully connected and programmable 8-port module by connecting two NPs back to back. The architecture of the NP4GS3 is shown in Fig. 3.

The NP contains 16 picoprocessors, each of which supports two hardware threads used for data manipulation, although only one thread per picoprocessor can execute at any given time. Each two picoprocessors share a set of nine coprocessors for common tasks such as table lookup and manipulation, string copy, check-sum calculation and verification, counter management, and policy enforcement. The coprocessor calls can be executed synchronously and asynchronously. For optimal performance, they can also release the priority of the running thread, such that the second thread belonging to the same picoprocessor can execute while the first one is waiting for the coprocessor to complete its task. Since the threads are implemented in hardware, context switch does not cost any processor cycles. Threads are dispatched upon frame reception either from the front-end ports (ingress processing) or from the switching fabric (egress processing). Entry points are defined separately for both ingress and egress stages, and on port basis, significantly adding to the flexibility of the processor. At the ingress stage, frame data are

Fig. 3. Internal architecture of the NP4GS3: the data flow is indicated with block arrows. Frame data can be manipulated both at the input (“ingress”) and the output (“egress”) stage of the chip. The memory organization differs for the two stages, resulting in two different event-building, or more generally, frame-altering algorithms.

stored in a small but fast internal memory, while data entering the egress stage are put in large external memory banks.

One implementation available for evaluation exists in the form of the IBM PowerNP reference platform, which provides two NP boards and a control processor unit in a compact PCI crate. This hardware was used to do the tests and measurements presented in this paper. Another commercially available module is the “Copernicus” card, designed by the S3 corp. [7], a PCI card providing one NP4GS3 and three gigabit Ethernet ports, the fourth one being converted to PCI-X. Two such boards can be interconnected using a DASL cable.

While the final board to implement the functionalities presented here is currently under design, it is likely to resemble closely the Copernicus card. The board we are designing will have eight interconnected ports, driven by two NP equipped with the required infrastructure (mostly buffer memory). The maximum shared output buffer memory for four ports (1 NP) is 64 MB RAM; the input memory is 128 kB. With a small internal ingress data store and a large external egress data store, the NP is best suited for high rate, small size fragment modifications on the ingress side, while the processing of larger fragments at lower rate is best done on the egress side.

The NP4GS3, like most other NP, can handle several protocol layers. Its hardware assists facilitate frame alterations, helping with tasks such as CRC update for Layer 2 or address overlay for Layer 3.

The NP4GS3 has an elaborated software development environment, which includes an assembler, a simulator, an interac-

tive graphical debugger and recently also a C-compiler. Apart from needing to master a proprietary RISC assembly language the programmer is faced with two major challenges: like any NP the NP4GS3 is a highly parallelised device, where only the concurrent operation of many independent hardware threads allows exploiting the full potential. In applications like frame merging or sorting, synchronization problems have to be treated with great care; a dedicated semaphore coprocessor helps with this task. The memory is organized in buffers, i.e., there is no flat address space, which is advantageous for networking hardware working with packets. Buffer manipulation by user software, however, is rather complicated, and care must be taken to ensure efficiency in accessing the memory. All global resources of the NP, memory, counters, search-tables, semaphores, and global registers are accessed via dedicated coprocessors.

IV. FRAME MERGING AND $N \times M$ MULTIPLEXING

At a frame rate of ~ 1 MHz, the small payload size results in low link utilization, as demonstrated in Section II. Frame merging and $N \times M$ multiplexing are used in our system in order to enhance the fraction of the bandwidth occupied by user data, and reduce the rate as well the size of the switching network. The method is illustrated in Fig. 4.

The NP receives packets at the event frequency (1 MHz) on N input ports. The transport headers and trailers are stripped off, and the user data are concatenated to a single payload. The merged frame, equipped with a new transport header, is for-

Fig. 4. Principle of $N \times M$ multiplexing with frame merging for $N = 4$, $M = 3$.

warded for transmission on one of the M output ports. As the output ports alternate, the frame rate on each of them is $1/M$ of the input rate. For $M = 3$, the effect of multiplexing is indicated in Fig. 2: user data occupancy raises from 63% and 80% at 1 MHz to 85% and 92% at 330 kHz for IP and raw Ethernet, respectively. At the same time, the max. payload more than triples. Table I gives, for $M = 1$ to 5, the maximum aggregated output payload possible for output link loads of 100% and 70%, using Ethernet protocol.

The processor can also, if required, perform a protocol translation, the input fragments being in the Ethernet format with the lowest possible data overhead. The IP protocol is added only in the last stage where the additional overhead has much less impact on the bandwidth utilization.

Given the memory model of the NP4GS3, we have approached the multiplexing problem in a modular way with 2×2 units as basic building blocks. Interconnecting two NP, we can use these units to perform $4 \times M$ multiplexing, with $M \in [2, 4]$. Due to the completely different memory layout on the ingress and egress sides, two different routines had to be implemented for processing at each stage.

A. 2×2 Multiplexing

In the 2×2 setup, fragment merging is performed on the ingress side of the NP. The combined frame is wrapped on the DASL to the egress side, where it is enqueued for transmission on one of the two output ports in round-robin mode. A second NP is used for fragment generation and for reception of merged frames. The payload is varied between 15^2 and 89 bytes. Frame and data rates were measured using internal counters of the NP.

The measured aggregated throughput of the two output ports of the multiplexing unit for an input rate of 1 MHz is shown in Fig. 5 as a function of the aggregated input payload. It can be compared to the aggregated throughput of two links without data merging (dashed line).

We have measured the fragment merging performance of the NP up to 178 bytes payload. Below ~ 140 bytes input payload,

²“Short frames” were used, i.e., frames with length shorter than the Ethernet standard requires. Frame padding is a hardware assisted function in this NP, and has no measurable effect on the performance of the processor.

TABLE I
UPPER LIMITS FOR AGGREGATED PAYLOAD $\sum p_i$ (ETHERNET PROTOCOL)

M	$\sum p_i$	
	L=100%	L=70%
1	99	61
2	224	149
3	349	236
4	474	324
5	599	411

the measured throughput matches the theoretical value for 1 MHz input rate. Compared to 2 single links, the gain is ~ 12 bytes per input link for the same bandwidth, or ~ 12 MB/s per output link occupancy for the same payload at input. An optimal gain would be obtained for $M = 3$, as can be seen in Fig. 2, however, this configuration requires the use of two interconnected NPs.

Above this value we see a clear degradation in merging performance, the input rate has to be lowered in order to prevent data loss. This effect comes from resource starvation inside the NP. Although the processor can handle 32 picocode threads, only 16 of them can run truly simultaneously. Moreover, there are only eight instances of each coprocessor, so that only this number of threads can access the data stores, or use the string copy coprocessor at any given time.

In the NP4GS3 merging of small packets can only be achieved by actually copying the contents of the incoming packets into a new packet. The memory copy performance of the chip does not allow going beyond driving two simultaneous Gigabit output streams. Bearing in mind, that the typical application of such a device consists of route lookup, address overlay and traffic shaping, all these operations involving no or only small data accesses, it is clear that our application is out of the intended scope. However, modern networking applications such as encryption and spam filtering require more and more “deep packet processing,” which drives future NP generations to more power for full packet analysis.

B. $4 \times M$ Multiplexing

Interconnecting two NP via the DASL interface offers the possibility to use the 2×1 and 2×2 multiplexing code as logical units to perform $4 \times M$ multiplexing. Here, we use the 2×2 unit at the ingress stage as described in the previous section. The combined fragment is then routed via DASL to either of the processors, depending on the event number and the choice of output ports (M). The egress stage performs the 2×1 or 2×2 multiplexing for the final 4×2 or 4×4 , respectively. Other combinations, such as 3×2 and 4×3 are also feasible in this approach, although with uneven loads on the processors. The setup is shown in Fig. 6. It should be noted here that the egress fragment merging happens at a fraction of the input rate³, but with larger fragments, fitting very well the memory model of the NP.

We have measured the performance in the case of 4×2 multiplexing. In the test setup, one NP is used to carry out fragment merging, while the second NP is acting as traffic gener-

³ $1/2$ in case of 4×2 and 4×4 multiplexing. For 4×3 , the loads are $1/3$ and $2/3$ depending on the number of output ports the processor has to handle.

Fig. 5. Measured aggregated output data rate for fragments merged at 1 MHz. Dashed line represents the calculated throughput without frame merging.

Fig. 6. Connecting 2 NP for $4 \times M$ multiplexing ($M = 2$ here).

ator. Two frames of 46 bytes payload (min. Ethernet frame) have been sent on two gigabit Ethernet links at 1 MHz and one frame of 92 bytes is sent via the DASL at 0.5 MHz. We have measured on a single port of the merging NP a sustained output rate of 0.5 MHz, with 105 MB/s throughput (84% link occupancy). Out of the 32 threads, only eight have been busy on average. Since we have chosen a safety limit of 70% link occupancy on the output ports, we have demonstrated that 4×2 multiplexing with 2 NPs is feasible in our system.

V. DECISION SORTING

The LHCb front-end electronics specifications require the Level-1 trigger decisions to arrive in order of event number in the front-end electronics boards. The front-end electronics buffers have a limited capacity, thus, a decision whether to discard the data or to read out the full event data for further processing has to be reached in the relatively short time of ~ 58 ms. This is the absolute maximum time allowed to pass from the event entering the buffer, its passing through a multiple stage network, being examined by an event selection processor, and the decision coming back in the right order.

Decisions will be determined in one of the CPU nodes in the subfarms and sent to a dedicated module for sorting. The

farm nodes send them as small Ethernet frames. An NP module receives the decisions, sorts them and sends the sorted decisions to the timing and fast control (TFC) system, which is responsible for distributing trigger decisions to the front-end modules.

The decision sorter has also to keep track of possible time outs. This means that it has to keep up with 2 MHz of incoming information (= frames): 1 MHz of “event has entered” messages, 1 MHz of decisions. The implementation requires only one standard NP module with four ports. We have implemented an algorithm which, by careful lock management, sorts incoming decisions at a rate of up to 1.1 MHz. As no heavy frame alteration is carried out, frame sorting falls into the domain of applications for which the processor shows optimal performance.

VI. SWITCHING

It is obvious that a NP module can be used for switching. Rather than discussing the (comparatively trivial) algorithms here, we will briefly discuss the advantages of implementing the main event-builder switch using the same NP-based module. In short, there are two main advantages.

- 1) The large buffers of the NP and the complete control over the switching process guarantee the capability of the resulting switch to cope with the rather unusual (as compared to “normal” or “random” internet traffic) traffic pattern we foresee in the DAQ system.
- 2) The complete control over the routing process allows minimizing the overheads, because the routing software in the NP can be written in a way so that it uses the transport information of the data to do the routing (“source-routing”).

The disadvantage of this solution is mainly due to the limited number of ports of the elementary module, which requires lots of modules to provide the required overall connectivity. Provided that one can find suitable commercial equipment, the break-even point in cost today is probably close to a system

Fig. 7. A module with 30 usable ports built from six eight-port modules. Sources (S) and Destinations (D) are mixed at each point. Some of the modules (3, 4, and 5) connect to further homologous modules.

Fig. 8. Eight modules from Fig. 7 connected to build a switch with 72 destinations and 63 sources.

with approximately 150 usable ports⁴. To improve this situation, we studied optimized network topologies, which make use of the fact, that the dataflow in a DAQ system is almost unidirectional, however, all links in a switched Ethernet system are bidirectional. Almost half of the installed bandwidth is thus not used. A topology like the one shown in Figs. 7 and 8, makes use of both directions of the connecting links and can reduce the number of elementary modules compared to a classical Banyan configuration by some 20%–30%.

A priori having more elements in a network seems not desirable, because each “hop” for a packet induces some latency and possibly increases the risk of bit errors. Simulation on other hand reveals a beneficial effect: congestion is much less a problem, because there is more buffering distributed over several stages. The traffic patterns are smoothed and thus we find that the average latency of packets through such a network

⁴Obviously, expected price drops for commercial switches in the future will shift the balance even further against building a network with such small modules.

is even reduced, compared to a system built from only four large devices.

VII. CONCLUSION

NP are software programmable devices to perform high rate packet processing. They provide the necessary I/O capabilities and CPU performance to do processing of millions of packets per second. We have identified several places in our DAQ system where such device can be extremely useful and we have shown that all the required functionalities, which are quite different, can be implemented with a single type of module. The only disadvantage of NP is their programming model, which is today quite far from what is common in the microprocessor world, partly due to the memory model, partly due to the highly parallelized nature of the processing cores. This makes it difficult to use high-level programming tools like the C language. The next generation of NP will not only be more powerful to cope with the upcoming 10 gigabit network standards, but also provide a more mainstream programming model.

We have evaluated the possibility of using gigabit technology, in connection with readout unit based on the IBM NP4GS3 NP, for the purposes of DAQ at rates of up to 1 MHz. Overheads due to transport protocols limit the bandwidth occupancy for user data. At 1 MHz and 100% link load, user data use 80% and 63% of the bandwidth using Ethernet and IP protocols, respectively. Frame merging and $N \times M$ multiplexing enhances the fraction of bandwidth occupied by user data. The basic building block of 2×2 multiplexing on a single NP allows one to process incoming packets with up to ~ 140 bytes combined payload at 1 MHz, while maintaining a low output link occupancy, below 70%. Interconnecting two NP to obtain a 4×2 multiplexing unit, we were able to aggregate four fragments of 46 bytes payload each at 1 MHz, resulting in output link load of 84%. The resources of the NP4GS3 are clearly sufficient for our purposes, a trigger-DAQ system with a 1 MHz input rate can be built using a readout unit based on this NP.

We have demonstrated that the NP are a very powerful tool in building future DAQ systems. Their main advantage lies in the ability to define the application dependent functionality at software level. The three examples of DAQ applications we have presented in this paper (frame merging, sorting, and switching)

can be built using the same hardware—the functionality is fully defined in software. We are confident that future releases, following very fast technology development in this field, will make NP even more adequate for use in DAQ systems.

REFERENCES

- [1] “LHCb Tech. Proposal,” report, CERN/LHCC 98-4, 1998.
- [2] A. Barczyk, J.-P. Dufey, B. Jost, and N. Neufeld, “The New LHCb trigger and DAQ strategy: A system architecture based on gigabit Ethernet,” *IEEE Trans. Nucl. Sci.*, vol. 51, pp. 456–460, June 2004.
- [3] H. Müller, “SCI implementation study for LHCb data acquisition,” report, LHCb-98-030, 1998.
- [4] *IEEE Standard for Information Technology, Part 3: Carrier Sense Multiple Access With Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, Std. 802.3, 2002.
- [5] The Linley Group. (2002). [Online]. Available: <http://www.linleygroup.com/npu/>
- [6] IBM NP4GS3 Datasheet (2001). [Online]. Available: <http://www-3.ibm.com/chips/products/wired/products/np4gs3.html>
- [7] Application Reference Board for the IBM Power NP NP4GS3 Network Processor (2003). [Online]. Available: http://www.s3group.com/network_processing/copernicus