Use of Network Processors in the LHCb Trigger/DAQ System

J-P. Dufey, R. Jacobsson, B. Jost and N. Neufeld

CERN, CH-1211 Geneva 23, Switzerland
beat.jost@cern.ch, niko.neufeld@cern.ch

Abstract

Network Processors are a recent development targeted at the high-end network switch/router market. They usually consist of a large number of processing cores, multi-threaded in hardware, that are specialized in analysing and altering frames arriving from the network. For this purpose there are hardware co-processors to speed-up e.g. tree-lookups, checksum calculations etc. The usual application is in the input stage of switches/routers to support de-centralized packet or frame routing and hence obtain a better scaling behaviour.

In this paper we will present the use of Network Processors for data merging in the LHCb dataflow system. The architecture of a generic module will be presented that has the potential to be used also as a building block of the event-building network for the LHCb software trigger.

I. Introduction

Network processors are a relatively new development. The first one was introduced by C-Port (now Motorola) in 1999. Nowadays every major and a lot of smaller chip-manufacturer has one in their product line.

A network processor is a dedicated processor for network packet (=frame) handling. It provides fast memory and dedicated hardware support for frame analysis, address look-up, frame manipulation, check summing, frame classification, multi-casting and much more. All these operations are driven by software, which runs in the network processor (NP) core. These processors are usually multi-threaded in hardware, multiple threads are running at the same time with zero-overhead context switching. They were primarily designed as powerful and flexible front-ends for high-end network switches and switching routers. Because they are software driven they can easily be customised to various network protocols, requirements or new developments. They allow to create really big switching frameworks, because the decentralise the address resolution and forwarding functions traditionally performed by a single, powerful control processor. Thus they enable switch manufactures to construct large switches (up to 256 Gigabit ports and more), with dedicated software in a short time. Currently the “Gigabit” generation of network processors is on the market, while the next one will be able to handle 10 Gigabit speeds (either as 10-Gigabit Ethernet or OC-192). These processors will be available in the course of 2002. More information can be found in [1].

We present the use of a specific network processor, the IBM NP4GS3 to implement a versatile module for LHCb. The NP4GS3 can be operated either together with a switching fabric or in back-to-back with a second NP4GS3. In this note we will summarise our experiences so far, and will demonstrate how a NP-based module can fulfil many uses in the LHCb data acquisition system, but also potentially in the Level 1 trigger system.

II. The IBM NP4GS3

The IBM NP4GS3 is a network processor, which comprises 8 dual processor units (DPPU), each being able to run 2 out of 4 total threads at the same time. Each DPPU shares a set of coprocessors, which regulate the efficient access to external resources, such as port queues, memory, tree look-up, check-summing and policy. The chip includes also 4 media access controllers, to which Gigabit Ethernet Physical Layer Interfaces (PHYs) can be directly attached either using the GMII or the 8/10 bit encoding. The processor has a 128 kB fast, on-chip input buffer, and a 64 MB output buffer, made from DDR RAM chips. The access to the memory is via a 128-bit wide data-path. The chip also includes a PPC 405 core for control and monitoring and exception handling. This PPC can run an operating system, if desired Also attached are various memory interfaces for very fast and fast address look-up memory, in total up to 64 MB. For more details the datasheet can be consulted [2].

The data-flow through the NP4GS3 is shown in Figure 1. Data is coming in from the ports, are stored in the ingress memory, can be accessed here and are then transferred to the Switch Interface Link (the DASL). From here they can either reach their own blade or a twin processor connected back to back to the first one. They will arrive in any case in the output buffer or egress memory, where they can be accessed a second time, before they are finally put on to one of several output queues for transfer over the network.

III. A Versatile 8 Gigabit Port Module

The IBM NP4GS3 has a high-speed interface to connect to a switching engine, the Data Aligned Synchronous Link (DASL). In fact it has 2 such interfaces. When there is no switching engine to connect to, one of these interfaces can used to connect to another NP4GS3, thus creating effectively an 8-port switch. The other DASL will be usually wrapped to itself to ensure full connectivity.

In addition to the DASL the NP4GS3s (can) share the following resources: power and clock distribution and access to a PCI interface for configuration and monitoring. Each

NP4GS3 requires its own memories and physical layer interfaces. The Media Access Controller (MAC) is already incorporated on-chip.

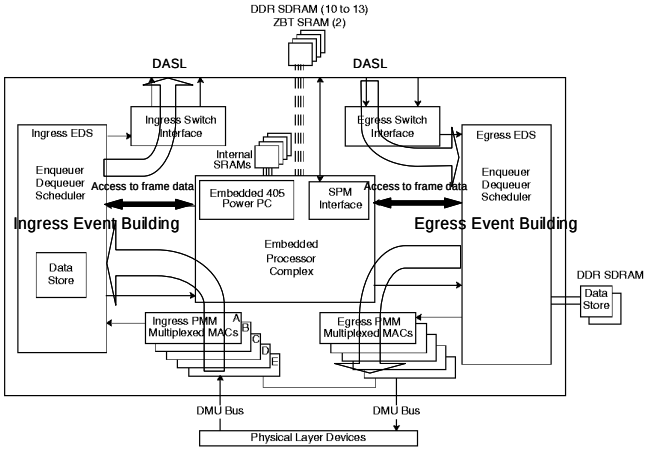


Figure 1: Main components of the NP4GS3 together with an indication of the standard data-flow paths. Data can be accessed and modified at the input/ingress and output/egress stage, leading to two different event building algorithms. One of the two DASL interfaces is always wrapped, so that each NP can send to itself. Also indicated are the various external memories.

Since the network processor and memory carrying part of the module is by far the more complex and deep (in terms of layers), it is very attractive separate this module of as a daughter/piggy-pack board, which carries everything belonging to one NP4GS3 alone (except the physical layers interfaces), and feeding out the connections for PCI, DASL (to the other Processor if present) and PHYs.

The common, "simpler" functionality and the control processor (Credit Card PC) would be housed on a motherboard. The two boards will be described in more detail in the following:

A. Motherboard

The mother board will provide all common "infra-structure" needed for the operation of the NP4GS3's. This includes power generation, clock generation and the physical layer interfaces. It will also include a Credit-Card PC (CC-PC) , the standard LHCb interface to the Experimental Control System (ECS). It provides the connectors for the two carrier cards with the Network Processors. These connectors carry the following lines and interfaces: DASL, PCI, JTAG, DMU. PCI is used by the CC-PC to configure and monitor the NPs and also to communicate with the embedded PowerPC. JTAG is needed for the boundary scan and for the hardware debugging using RISCWatch [3]. The DASL is used to connect two NPs, as has been said already. The DMU (Data Mover Unit) interfaces connect the Media Access Controllers integrated on the NP4GS3 to the physical interfaces. These could be hot-plugable, thus allowing more flexibility in configuring them either as 1000BaseT (CAT 5 copper) or 1000BaseSX (multi-mode fibre).

Except for some length requirements on the DMU and DASL lines and some necessary screening for the high frequency signals this mother-board will not be particularly complex. A simple layout is shown in Figure 2.

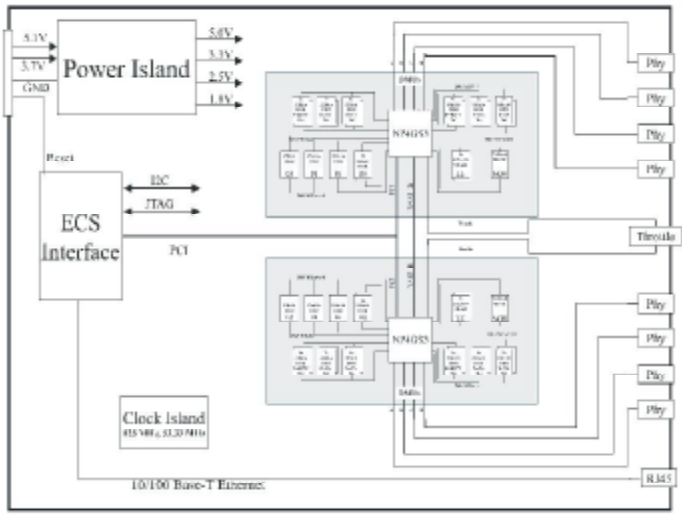


Figure 2: Mother-board for the NP4GS3 carriers. It provides power, clock and PCI to both NPs. Also shown are the 9 physical connectors (8 for the NPs, one for the CC-PC).

B. Piggy-back Board

This board will be comparatively complex, with ~ 12 layers and has rather stringent requirements on timing and distances. To have it on a small carrier board has therefore quite some advantages: the multi-layer board can be kept small, which eases production, and the flexibility to connect different physical layers is kept. An simplified block-diagram is shown in Figure 3:.

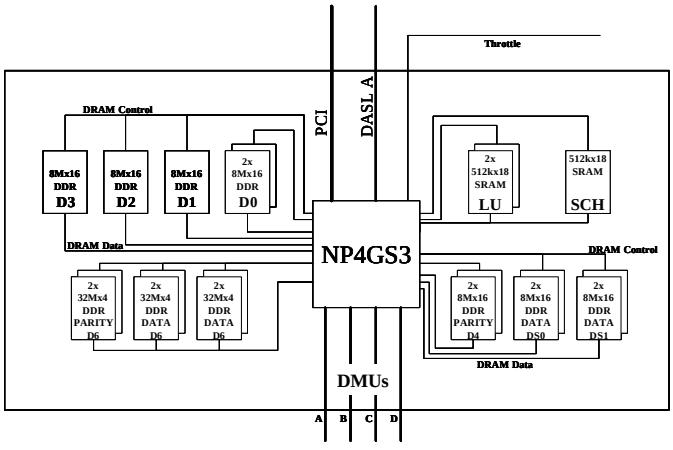


Figure 3: The piggy-back or carrier board will house the NP4GS3 processor and its associated memory-chips.

IV. APPLICATIONS IN LHCB

The flexibility of the module allows for a range of applications in the LHCb Data Acquisition system. They are

shortly discussed here. For details about the LHCb DAQ system see for example [4].

1) Readout Unit

The most obvious application is as a Readout Unit, which is interfacing/multiplexing front-end links to the Readout Network. The network processor is in this context as a fast sub-event merger, to assign destinations and function as a front-end to the event building switching network. The Readout Unit always has one (and only one) output to the Readout Network (RN), it can have several inputs (usually either 2 or 4). The NP-based Readout Unit will merge the sub-fragments and send them out to the network, using addresses determined by a pre-loaded address table. It will respect flow-control messages ("X-On/X-Off") from the network, to cope with local congestion, and it will itself have the possibility to throttle the trigger, when its buffers are about to overspill.

2) Front-end Multiplexer

The Front-end Multiplexer (FEM) application is basically the same as the Readout Unit. The multiplexing factor will be anywhere, between 7 and 2. For multiplexing factors smaller then 4, 2 FEMs can be implemented using a single, fully equipped module.

3) Main Event builder

The main event builders task is to collect all the fragments belonging to a specific event, originating from the RUs. This will be some 100 fragments, which have to be assembled into one contiguous event and sent to the Sub farm Controller (SFC). The fragments will arrive out of the order from the network, because the LHCb data acquisition does not have (nor does it want or need) any synchronisation after the Level 1 derandomisers. They have to be re-arranged into correct order and the possibly empty data and error blocks have to be merged.. Since the rates are low at this stage, one module could drive 4 event building streams, that is it can feed 4 SFCs.

4) Elementary Switching Module

The 8-port module can also be used as the building network for the switching network itself. Performance-wise this is definitely no problem, because this is the original domain of the network processors from their conception. The question is then more if such a switching network can be cost-effective on a price/port basis. Obviously one needs quite a lot of them, because one has to provide interconnections, which serve the purpose of the backplane in a conventional monolithic switch. There are however studies on how to reduce the number of modules, by making intelligent use of the traffic patterns in a data acquisition system see for example [4]. Additional advantages would be, that such a module would allow full control over switching process and functionality. Flow control, traffic shaping, check-summing could be implemented at will and customised for maximum performance in the DAQ. Furthermore such a switching network could do the final, main event building in its last stage.

V. Examples of Software for Applications

A. Sub-event merging in a Readout Unit

The task here is to collect up to 7 fragments arriving at a rate of at most 100 kHz. The fragments have an average size of a few 100 Bytes, which increases after successive levels of multiplexing. Sub-event building proceeds by analysing the fragment headers and waiting until all fragments belonging to an event have been received or a time-out condition has occurred. In any case event-building will start, in the latter case an error will be flagged in the error block. The frames will then be connected by adapting the link-pointers, moving as little data as possible. At the boundaries of original frames, it sometimes becomes necessary to actually copy some data to fill from the bottom. New frames are being built until a pre-defined maximum transfer unit has been reached. The frame is then dispatched, and the procedure is iterated until all data have been used up.

Care has to be taken to avoid corruption of the static data, due to multiple threads wanting to access them at the same time. The NP4GS3 provides a powerful semaphore mechanism to handle these situations.

Performance of egress event building according to simulation is shown in Figure 4:, only 16 out of 32 threads have been enabled, an improvement of at least 50% can still be expected.

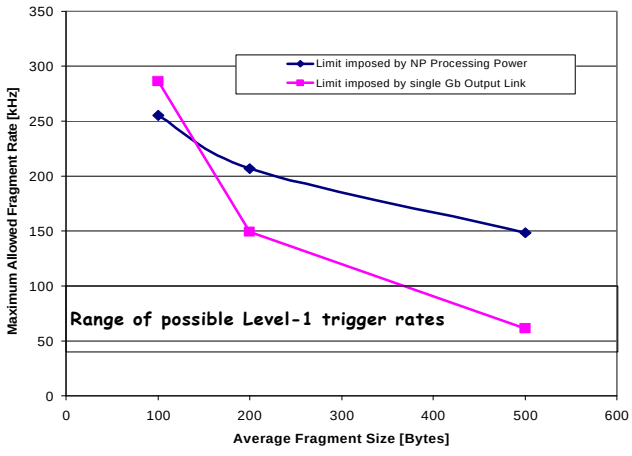


Figure 4: Performance of the Egress Event Building as a function of the average input fragment size. The green area shows the range of possible L1 trigger rates.

B. High rate event-building

Another application which could be of interest for the LHCb Level 1 trigger is sub-event merging at high rates of incoming fragments. Here very small fragments of some 30 to 50 Bytes are coming on 2 to 3 links at rates above 1 MHz.

The very high speed of the ingress memory, makes it ideal to perform event-building at high data rates and high trigger rates. The main idea is to store the data fragments waiting for all belonging to the same event having arrived and then copy the payload to form a new fragment to be sent towards the output ports. After stripping of the transport headers only part of the incoming data is transferred to the egress side after the event-building process, including the new transport structure for the outgoing event fragment. It has been shown that this strategy allows for event-building performances far beyond the capabilities of the output port. The performance is shown in Figure 5:.

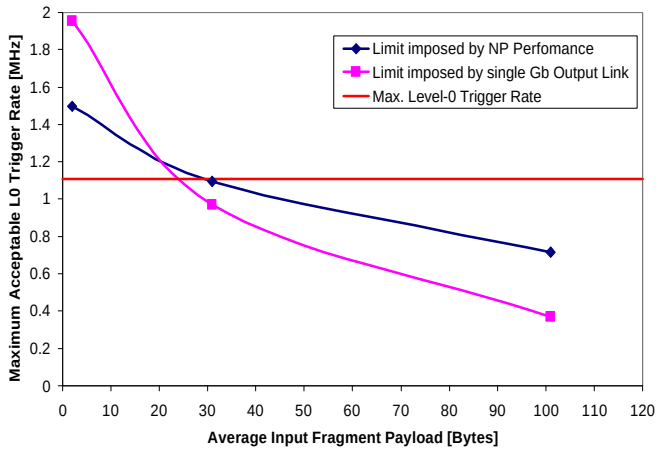


Figure 5: Performance of Ingress event-building from simulation as a function of the average incoming packet size. The straight line shows the fixed level 1 trigger rate of 1.1 MHz. The limitation on performance comes from the output link bandwidth only.

VI. Measurements with the IBM PowerNP Reference Platform

The results presented so far have been obtained using simulation. It is therefore interesting to assess how reliable the timing information of the simulation is. Since the latest version of the processor (revision 2.0) is not yet available on a reference platform, some parts of the sub-event building code cannot run un-modified on the reference design hardware. However, it has been possible to compare a representative set of algorithms on simulation and real hardware. The results agree well. They will be described in the following:

A. The IBM N4GS3 Reference Platform

The IBM NP4GS3 Reference Kit [6] aims at providing users with an implementation, which allows exploring most of the functions of the processor. The chassis with the important cards is shown in Figure 6. The configuration used for our measurements consisted of a chassis, a control processor (a PPC 705 based cPCI computer), two carrier boards with a NP4GS3 and 4 daughter cards, each with 2 Gigabit Ethernet SX optical ports. Furthermore we had a RiscWATCH ([3]) probe attached to the JTAG interface of one of the blades, which allows to access the network processor's internal registers directly from the NPSCOPE debugger (via Ethernet).

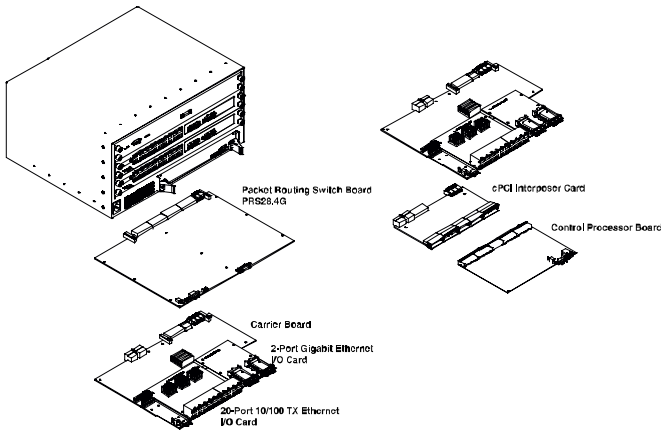
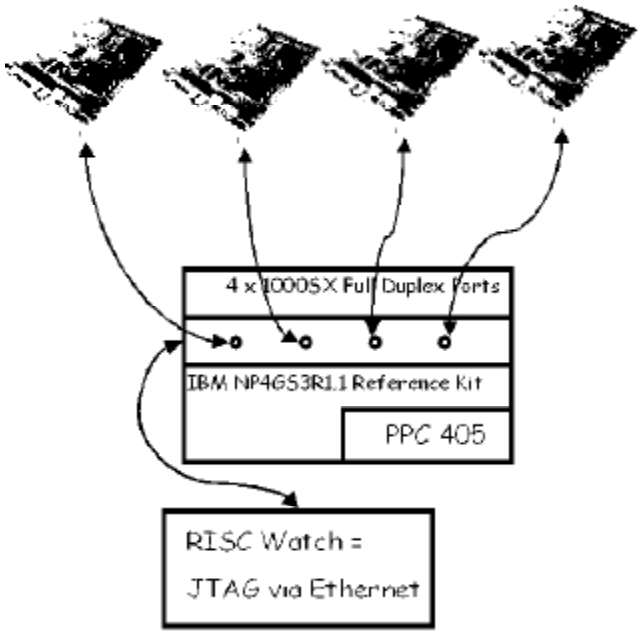


Figure 6: Main components of the NP4GS3 reference platform. The packet routing switch board is not included in our set-up.

B. Test set-up

The test set-up shown in Figure 7: consisted of 4 Netgear Gigabit Ethernet NICs, running a dedicated firmware, which made them traffic generators and sinks. The internal clock of the NICs allowed to measure latencies with approximately 1 µs precision. More information about these "smart NICs" can be found in [7].



.

Figure 7: Test-setup for the NP4GS3 reference kit. The data are fed and read back using four Tigon 2 based Gigabit Ethernet NICs, shown at the top of the figure. Download of NP software and monitoring of the NP is done via JTAG using the RISCWatch probe

The network processor is accessed remotely via the RiscWATCH. This configuration allowed to test 3 to 1 event-building. Each NIC has an internal clock which has been used to measure the latencies imposed by the event building code. This clock has an intrinsic resolution of 1 micro-second. The generation of frames and the evaluation of the time-differences in the receiving NICs, which are the same as the senders.To synchronise the NIC (the sources), which is necessary for the reasons outlined above, a special frame is send by one of the NICs to the NP, which then multi-casts it to all NICs. This triggers the collective sending of the packets, within 0.5 microseconds.

C. Results

The main aim of all measurements was to understand the accuracy of the simulation. The simulation is claimed to be cycle precise. It takes into account the contention between threads. However, it does not accurately simulate all external resources with their associated latencies. It was therefore especially interesting to see to what extend simulation results can be trusted. One problem with these measurements is that the version of the NP on these boards is not the latest. It lacks the semaphore coprocessor, a unit specifically designed for efficient resource protection to avoid race conditions in a multi-threaded application. Since our code heavily relies on this feature, it was necessary to tune the test conditions somewhat. This has been done by doing either a single thread measurement or by reducing the spread in the arrival time of the fragments by careful synchronisation. This does not change the run-time of the code, the simulations for both versions agree on that. It allows, however to avoid the occurrence of synchronisation problems which would otherwise be avoided by the semaphore coprocessor. We are confident that these measurements give a realistic impression of the performance of our sub-event building codes.

Measurements have been done only in the “high rate“ environment, which means short frames at high rates, since this is the more demanding and critical application.

Table 1: Comparison of measurements with simulation results. The handling time per fragment is shown in microseconds. The measurement times are shown once raw as measured, and second corrected, with the round-trip and handling time in the NICs subtracted.

	Measurement [µs/fragment]	**Simulation [µs/fragment]**
1 source 1 thread	6.6(4.9)	4.9
4 sources 1 thread	4.5(2.8)	3.2
1 source 16 threads	1.7 (0.0)	0.5

First the round-trip time of a packet has been measured. This is necessary to subtract any overheads coming from the transport over the DASL, the cables and especially the creation and time-stamping in the NICs, which are not included in the simulation. This time has been found to be 1.7 µs. It can be seen as an intrinsic resolution of the measurements. Since the whole system is pipe-lined (many threads working) time-intervals smaller than this intrinsic time cannot be accurately measured. This is the reason for the apparently strange value 0.0 in the last row of the following table.

Several scenarios have been tried, varying the number of active threads and active sources. The results are summarized in Table 1:.

VII. CONCLUSIONS

In this paper we have presented the use of a Network Processor for several applications in the LHCb Data Acquisition System. An integrated module has been described, whose function would be determined only by the software driving it, providing maximum flexibility and excellent debugging capabilities.

Two such sample software codes have been developed and benchmarked using a cycle-precise simulation .

The simulation results have been compared with measurements obtained with the reference platform of the IBM PowerNP. The results are in very good agreement, making us confident that we will have one, and only one, powerful, versatile module for the LHCb Data Acquisition.

VIII. REFERENCES

[1] Network Processor Central, [online] http://www.linleygroup.com/npu

[2] IBM PowerNP NP4GS3 Datasheet, [online] http://www3.ibm.com/chips/techlib/techlib.nsf/techdocs/852569B20050FF7785256983006A3809

[3] RISC Watch Debugger, [online] http://www3.ibm.com/chips/techlib/techlib.nsf/products/RISCWatch_Debugger

[4] B. Jost, “The LHCb DAQ system”, Presentation at the DAQ 2000 Workshop, Lyon

[5] J. P. Dufey et al., “Results from Readout Network Simulation” LHCb Note in preparation

[6] IBM PowerNP NP4GS3 Reference Platform, [online] http://www3.ibm.com/chips/techlib/techlib.nsf/techdocs/546B9AC56334EA0F872569F9005F7DA5

[7] LHCb Event-Building, [online] http://lhcb-comp.web.cern.ch/lhcb-comp/daq/Event-Building/default.htm