NOE : A PROGRAMMABLE NETWORK CONTROLLER

J.L. DESBARBIEUX, A. ZERROUKI, F. WAJSBÜRT, C. SPASEVSKI, A. GREINER
University P. & M. Curie, Laboratoire LIP6, ASIM Team
4 Place Jussieu 75252 Paris Cedex 05 France
Tel : (+331) 44 27 52 53 Fax : (+331) 44 27 72 80
E-mail : Jean-lou.Desbarbieux, Amal.Zerrouki, Franck.Wajsburt, Cyril.Spasevski, Alain.Greiner }@lip6.fr
Web : http://mpc.lip6.fr

ABSTRACT

This paper presents NOE, a programmable network controller designed to allow the implementation and the evaluation of a variety of message passing protocols. Such a component is adapted to build PC-clusters since it interfaces any CPU board offering a connection to the PCI bus to the high speed HSL network defined by the IEEE 1355 standard. NOE offers a 32/64 bit, 33/66 MHz PCI interface allowing a throughput of up to 520Mbytes/s. NOE includes also a set of programmable, parallel and asynchronous operators controlled by an external micro-controller. The paper describes also the micro-controller firmware named "BEE executive". This micro kernel provides the programmer with communication primitives as well as synchronization mechanisms necessary when implementing a specific message passing protocol on NOE.

NOE is a VLSI ASIC. It contains 1,3M transistors and its area is 45,3mm^2. It has been fabricated in a 0.25mm process. It is 33/66 MHz PCI compliant and can operate internally at up to 160 Mhz.

KEYWORDS : *network controller, VLSI, HSL networks, low-level protocols, parallel processing.*

1. INTRODUCTION

NOE has been designed in the frame of the MPC project whose objective is the design of low cost and high performance parallel computers. The MPC parallel computer[2] is currently available and used by several universities. It offers 16 processing nodes connected through HSL links. A specific board has been designed to connect the nodes to the HSL network. It includes the RCUBE router[3] which realizes the routing function in the HSL network and PCI-DDC a dedicated network controller implementing the Direct Deposit State Less Receiver Protocol (DDSLRP) [4]. This specific protocol uses the remote-write primitive : at the emitter end, the processor adds a message descriptor (start address, size, remote address, …) in the list of messages to be sent. Then, through DMA reads, PCI-DDC fetches data to be transmitted from the local memory, and handles data transfers to the remote node memory without any intervention of the processor. At the receiver end, PCI-DDC writes the incoming data in memory through DMA accesses before notifying the processor.

Such a protocol is efficient and reduces significantly the processor overhead and transfer latency. Classical data transfer protocols usually require several data copies in intermediate buffers before and after transmission through the network. However, PCI-DDC introduces some limitations that forbids any hardware optimization. As an example, PCI-DDC requires physical addresses. Virtual to physical address translation has still to be performed by the processor. Moreover, no programmability is available since the DDSLR protocol is hardwired in the PCI-DDC chip.

A new network controller, named NOE, has been designed to address theses limitations. NOE brings programmability and allows thus protocol optimizations at the hardware level and new low-level protocol evaluation.

2. THE NOE-HSL BOARD

Figure 1 presents the NOE-HSL board built using the NOE network controller. It consists of NOE, a micro-controller, and some memory (RAM and ROM). The board is connected to the processing node (e.g.: a PC) through the PCI bus. NOE offers a 32/64 bit, 33/66 MHz PCI interface. The PCI registers of NOE can be mapped into the processor memory space, allowing thus the configuration of the network controller directly by the processes.

To enable load-balancing on the network, two bi-directional HSL serial ports are provided by NOE. Each link allows a throughput of 1.5Gbit/s.

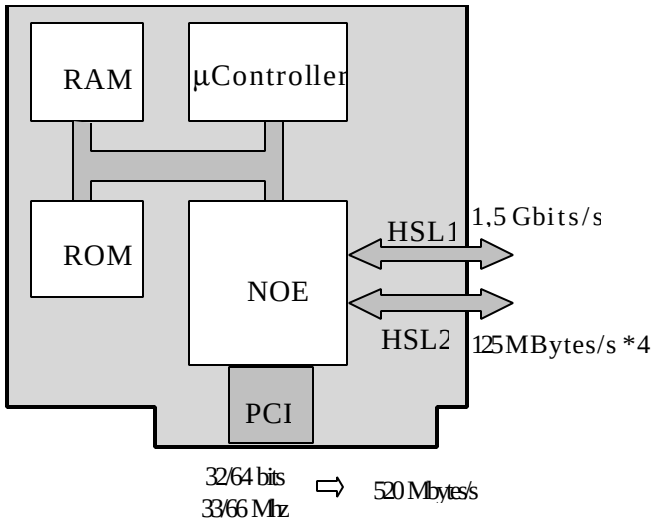

Figure 1 : NOE-HSL board

In this architecture, versatility is partly brought by the on-board micro-controller. NOE handles data transfers, DMA accesses to and from the host memory, and packet emission (and reception) on (and from) the HSL network. The communication protocol is programmed by the on-board micro-controller. The Command and Status registers of NOE are mapped into the micro-controller memory space. By reading the status and writing commands in these registers, the micro-controller controls the NOE operation to implement a specific communication protocol.

3. NOE ARCHITECTURE OVERVIEW

Figure 2 presents the internal architecture of NOE[5]. It includes 12 operators, 14 fifos, 2 bi-directional HSL ports and a 32/64bit, 33/66 MHz PCI interface.

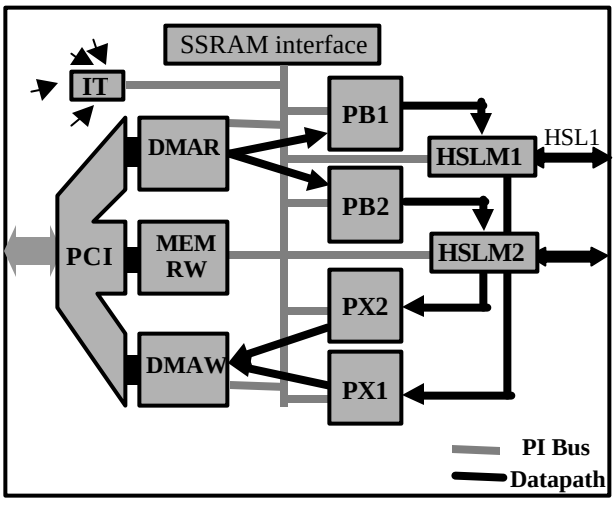

Figure 2 : NOE internal architecture

NOE operators are connected together through a PI-BUS[6]. This bus is connected to the micro-controller through an SSRAM interface. NOE operators can be seen as independent blocs. Each of them includes Command and Status registers. Setting the command register of an operator starts the operator. As soon as finished, the operator sets its status register to notify the operation completion to the micro-controller.

4. PCI OPERATORS

4.1 PCI controller

The PCIC block implements the PCI protocol and is not controlled by the micro-controller. It manages the PCI accesses in both master and target modes.

4.2 PCI target

MEMRW handles the PCI target requests. Through the PCI bus, NOE offers two distinct memory spaces:

- NOE command/Status registers (4Kbytes) : this space includes two 32-bit interrupt registers to generate interrupts from/to micro-controller/processor and vice-versa, two register files of 32 registers each, and the NOE configuration registers.

- Micro-controller space (4KBytes to 4GBytes) : this space is provided to receive the processor requests whose destination is the micro-controller. The size of this space is programmed by the micro-controller at boot time.

Thanks to its internal fifo, MEMRW allows up to 16 read/write posted requests targeting the micro-controller. A single register is provided within MEMRW to return the result of each posted read targeting the micro-controller space.

4.3 DMA operators

DMAR and DMAW allow NOE to act as a master on the PCI bus.

DMAR reads the host memory through DMA accesses. This operator accepts up to two requests posted by the micro-controller. This last indicates to DMAR the DMA start address in host memory, the data size, the PCI command, and the destination fifo of NOE in which data should be written.

Three destination fifos are possible. Each packet builder is connected to DMAR through a fifo to solve synchronization discrepancies. These fifos are used to store temporarily data coming from the PCI bus and going to the HSL network. A third fifo connects DMAR to the micro-controller. This fifo stores data coming from the host processor and going to the micro-controller, such as a new descriptor of a message to be sent.

If any fifo gets full during a PCI transaction, DMAR can stop the transaction and resume it as soon as the destination fifo gets ready. It also handles misalignment and is 32/64 bit capable.

DMAW writes data into the host memory through DMA accesses. It accepts up to 2 DMA requests posted by the micro-controller thanks to its internal request fifo. A micro-controller request consists of the start address in the host memory at which data should be written, the data size, the PCI command, and the source fifo from which data should be read.

Here again, three source fifos in NOE are possible. Each packet extractor writes data extracted from the HSL link into a fifo. A third fifo connects the micro-controller to DMAW. As an example, when a message has been fully received, the micro-controller pushes the message descriptor so that DMAW writes the descriptor into the host memory to notify the message reception to the host processor.

As for the DMAR operator, if any source fifo gets empty during a PCI transaction, DMAW can stop the transaction. It will resume the transaction as soon as new data is detected in the fifo. It also handles misalignment and is 32/64 bit capable.

5. PACKET HANDLING OPERATORS

Two independent and parallel data-paths can be identified in this architecture, one per HSL port. Each path includes a packet builder (PB), a packet extractor (PX), and an HSL interface (HSLM). These operators are interlaced with buffering resources made of fifos to solve synchronization discrepancies. Both data-paths share the same and unique PCI resource.

5.1 Packet builder

The packet builder, PB, assembles data and control characters to generate packets to be transmitted on the HSL link. Basically, PB reads the header provided by the micro-controller, and the data brought by DMAR from the related fifos. A packet consists of one or more sections separated by the ES special characters (*End of Section*) to allow building various packet formats.

The packet builder is programmable. It reads instructions posted by the micro-controller in the PB instruction fifo. Assigning the micro-controller the charge of providing headers and building instructions to PB brings the flexibility necessary to adapt the packet format to the protocol under evaluation. The only constraint on the NOE packets is that they must consist of sections separated by the special character ES so that headers and/or data can be properly extracted by the receiver packet extractor.

5.2 Packet extractor

The packet extractor, PX, handles packets coming from the network and identifies the data in an incoming packet. PX operates independently. Interaction with the micro-controller is not necessary when receiving packets delivered by the network. Thanks to the use of ES characters, PX can identify different packet formats.

Each section of a packet corresponds to a destination fifo. There are two possible destination fifos. As a general rule, the first section of a packet contains always the header. This last is pushed by PX into a fifo (PX2LP) read by the micro-controller that will interpret the header to program the DMAW operator. The second section starts when an ES character is identified. This section generally corresponds to the data itself. It is stored in a second fifo (PX2DMAW) to be read by the DMAW operator. Each new ES character defines an new section, and the destination fifo switches alternatively from PX2LP to PX2DMAW.

5.3 HSL operators

The HSLM block is dedicated to the communication on an HSL link. It is specific to the HSL network. NOE includes 2 instances of this block, one per link. It computes CRC to guaranty packet integrity, detects errors in incoming packets and notifies errors to the packet extractor.

NOE offers 2 serial ports allowing the connection to the HSL network. Each serial port is actually controlled by a specific HSL core[1] communicating with HSLM. This core has been developed at UPMC in collaboration with BULL. The core performs serialization/de-serialization and symbol coding of data according to the 1355 HSL standard, clock recovery, and parity generation and comparison.

6. SIGNALLING HANDLING

Given that the micro-controller can implement various protocols, NOE offers two different signaling policies: polling and interrupts. Although they are rarely used because of their cost, NOE can generate hardware interrupts that might be necessary in protocols having real time constraints.

Polling is preferred to interrupts because in this architecture, the micro-controller is dedicated to NOE control. It issues commands to NOE operators and waits for their completion while polling the NOE status register.

The IT block in NOE contains the general status register which indicates the status of each operator. It also implements a programmable priority mechanism to speed up the interpretation of the status register by the micro-

controller. A mask register is also provided to select or disable interrupts.

7. NOE-HSL FIRMWARE

As described above, NOE consists of a set of programmable operators. As a consequence, the performances of this architecture depend directly on the micro-controller firmware. Such a firmware must be capable of reacting to the various hardware events generated by NOE (arrival of a new packet header on an HSL link, detection of a host processor request in MEMRW, …).

Basically, the firmware should be able to handle simultaneously four activities : two emissions (one per HSL link) and two receptions. This firmware must be optimized as much as possible to take advantage of the parallel feature of the operators of NOE.

To satisfy these requirements, a micro-kernel has been developed. BEE (Best Effort Executive) allows the execution of several tasks simultaneously.

It is non pre-emptive, and all the tasks share the same address space. Each NOE activity (message emission, message reception,…) is implemented as a task. All the tasks are statically allocated at boot time. Synchronization among tasks uses queued semaphores. A specific scheduler implementing circular priority has been developed. It assigns the micro-controller to one of the ready-to-run tasks.

In order to evaluate the performance of a specific protocol on the NOE-HSL board, a software environment has been developed on a Linux PC-platform. It allows the description of protocols under the forms of tasks and offers a simulation environment and debugging facilities.

The DDSLR protocol hardwired in the PCIDDC [4] network controller has been implemented on this plate-form and encouraging results have been obtained : less than 6µs latency and a full PCI throughput. This implementation demonstrates BEE efficiency : only 26 micro-controller cycles are necessary for a context switch. When two emissions and two receptions are performed simultaneously, penalty on latency du to BEE is of 20%.

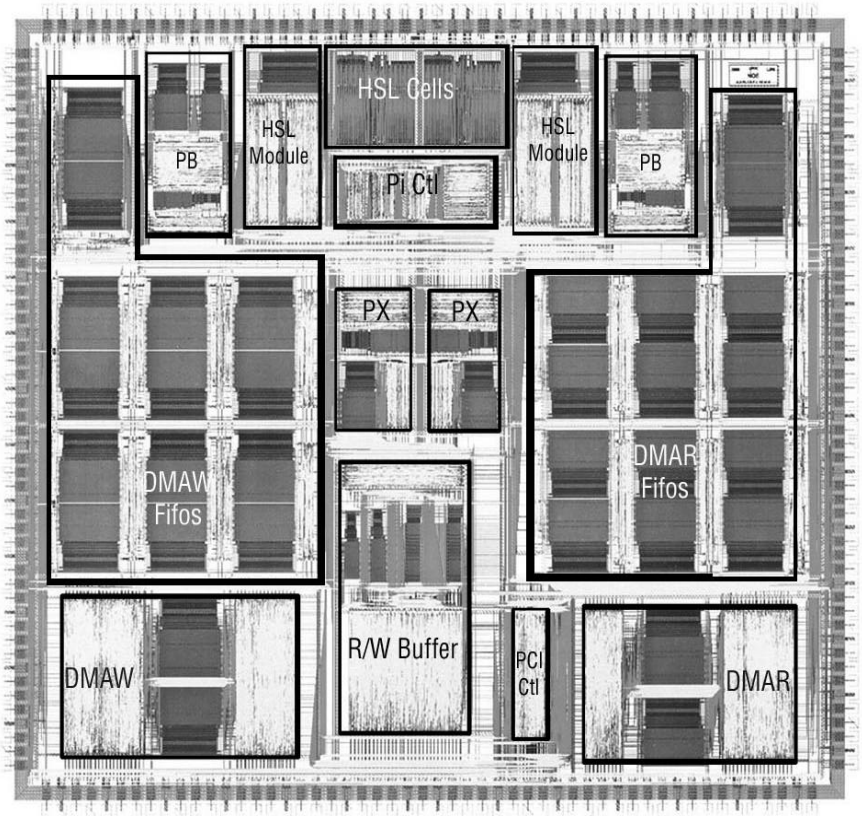

Figure 3 : NOE layout

8. CONCLUSION

This paper presented the architecture of NOE, a programmable network controller designed to interface a standard PC CPU board to the HSL network. NOE contains 12 operators for data transfer, 14 buffering fifos to solve synchronization discrepancies, 2 1,5 Gbits/s full-duplex and bi-directional serial HSL ports and a 32/64bit, 33/66 MHz PCI interface.

NOE is an ASIC. It contains 1.3 M transistors and the chip area is of 45,3 mm². It has been fabricated in the ST microelectronics 0.25 µm process and is encapsulated in a QFP 208 package.

A micro-kernel has also been presented in this paper. BEE (Best Effort Executive) allows the execution of several tasks by the micro-controller. Although it has been designed in the frame of NOE project, this micro-kernel can still be used in any embedded system using a processor core.

The differentiating feature of this architecture resides in fact that NOE operators are programmable. A micro-controller is connected to NOE and is assigned the charge of programming the operators to implement a specific low-level communication protocol. Buffering resources included in NOE added to autonomous operators reduce considerably timing requirements on the micro-controller firmware. BEE takes advantage of this feature to provide protocol programmers with a set of primitives for sending and receiving messages using NOE.

As a conclusion, programming a new protocol on the micro-controller using BEE and NOE does not require specific knowledge of the internal architecture of the network controller. Programmability is made compatible with performances.

9. REFERENCES

[1] IEEE 1355, *IEEE1355 Standard for Heterogeneous Interconnect (HIC) Low Cost Low Latency Scalable Serial Interconnect for Parallel System Construct*ion, (IEEE Standards Department, Aug. 1994).

[2] A. Zerrouki, O. Gluck, J.L. Desbarbieux, A. Fenyö, A. Greiner, C. Spasevski, F. Wajsbürt, F. Silva, E. Dreyfus, The MPC parallel computer : hardware, low-level protocols and performances, *in Proc of IASTED Parallel and Distributed Computing and Systems (PDCS 2000*), Las Vegas, Nov. 2000, pp. 87-92.

[3] V. Reibaldi, Conception et réalisation d'un router de paquets à hautes performances, *PhD thesis of University Pierre et Marie Curie*, France, 1997.

[4] F. Wajsbürt, J.L. Desbarbieux, A. Greiner, C. Spasevski, S. Penain, An Integrated PCI component for IEEE 1355 Networks, *In Proc. of EMMSEC'97*, Florence, Italy, 1997.

[5] JL. Desbarbieux, Conception et réalisation d'un contrôleur réseau programmable pour machine parallèle de type "grappe de PC", *PhD thesis of University Pierre et Marie Curie*, France, 2000.

[6] Open Microprocessor systems initiative, *OMI 324 : PI-Bus*, April-May 1996. Available at www.omimo.be/public/data/_indstan.htm#OMI324.