IP-SoC 2006
IP Based SoC Design Conference
December 2006, Grenoble, France

MpNoC Design: Modeling and Simulation

Simon DUQUENNOY Sébastien LE BEUX Philippe MARQUET Samy MEFTALI
Jean-Luc DEKEYSER

LIFL and INRIA-Futurs
University of Lille
France

Abstract

MppSoC is a SIMD architecture composed of a grid of extended MIPS R3000 processors, called Processing Elements (PEs). This embedded system gives interesting performances in several modern applications based on parallel algorithms. Communication is clearly a key issue in such a system. In fact, regular communication between the PEs are assumed by a X-Net network, while point to point connections seem to be very tedious to realize using such a network.

We present in this paper a model and an implementation of a communication network called mpNoC. This IP permits non-regular communications between PEs in an efficient way. MpNoC is integrated in the mppSoC platform.

1. Introduction

Now-a-days, a significant class of SoCs has emerged for several modern applications. In fact, we see more and more application-specific SoCs for mobile applications, audio- and video-based multimedia products, wireless communications and imaging systems, signal processing...

These applications need high execution speed and often real-time processing capabilities. But, unfortunately their performances are more and more limited by lifetime of batteries, because of their growing power consumption. Thus, new design methodologies producing more power-efficient and higher throughput designs are greatly desired for future SoC architectures design.

In the nineteenths, designers focus on SIMD architectures for classical on-board systems. This allows them to speed-up significantly execution times of complex applications. But, until the last few years, silicon integration technology doesn't allow us to such systems to the SoC context. Since recently, we are able to integrate billions transistors in a single chip (giga and tera-scale integration 'GSI/TSI'). This permits to put together hundreds of simple processing elements and memories on-chip. This great evolution allows us so to imagine SIMD systems on-chip.

Interconnects are considered as one of the most important challenges in GSI/TSI are facing today. At the chip level, interconnects have become the major component in the delay of critical paths, are the largest source of power dissipation, generate crosstalk and power supply noise, and cause reliability problems due to electromigration and fragile low-k materials.

Our contribution to SIMD on-chip design domain consists in the implementation at a cycle accurate bit accurate level (CABA) of a system called mppSoC. It looks like an on-chip version of the famous MasPar [3]. MppSoC contains a configurable number of processors, having local memories. The whole system is mastered by a particular processor called ACU. The processors communicate using a X-Net network.

The X-Net network permits regular and systematic communications between processors, but is not really efficient when communications become irregular. Thus, to improve the performances of the mppSoC, especially in terms of communication we need an additional and novel communication network.

The objective of this work is to provide designers with a flexible NoC implemented as a SystemC IP. Our NoC called mpNoC is furnished and integrated in an SIMD SystemC platform. This later contains processing elements (extended MIPS 3000), memories and regular X-Net interconnection network.

In order to allow mpNoC to interact with its environment, we also furnish a special protocol permitting to manage the Global Router and the PEs. Thus, some new instructions and new wires have been added to the processing elements.

The rest of this paper is organized as follows: a NoC state of the art is proposed in the next section. Section 3

briefly presents the mppSoC platform. The general operating mode of the mpNoC is explained in Section 4. Section 5 details the mpNoC IP possible configuration and use. In Section 6, we propose an efficient way to integrate the mpNoC IP with the mppSoC processors and devices. Section 7 presents an application using our proposed network for non-regular communications. Finally, Section 8 concludes the paper.

2. Related Works

Recent years have seen a proliferation of research in Network-on-Chip (NoC). There has been a growing interest in designing novel NoC alternatives, both in academia as well as in industry. These research projects usually target specific application domains, such as multimedia and network processing. Thus, the evaluation of these proposals needs a detailed understanding of the application and its underlying PE architectures and NoCs. Additionally, a novel NoC design is usually proposed as an integral part of a novel multiprocessing computing platform. Thus, we present in this section some relatively recent and significant efforts in this domain in a non-exhaustive manner.

The Philips Nostrum Architectures [7] is integrated in a Digital Video Platform (DVP). It is based on a 2D mesh topology, and used for a specific heterogeneous SoC. The Intel IXP 2800 network processor is a programmable network processor which targets OC-48 and OC-192 processing for network applications [1]. The Sonics Silicon Backplane Micro Network provides a time-division multiple access (TDMA) protocol [9], it has been successfully applied to a number of embedded system applications. The SPIN micro network is an on-chip micro network based on deterministic wormhole routing [2], its topology is a fat tree. The Octagon on-chip communication architecture targets the bandwidth need for high performance network processors [4]. In Octagon architecture, 8 nodes are connected through 12 bi-directional links.

All these NoC topologies are efficient for their specific platforms. But, to our knowledge they are not enough flexible to be used in a large SIMD system such the mppSoC.

3. MppSoC

This section details some technical aspects of the mppSoC in order to facilitate the understanding of the following sections. A full description of the mppSoC is out of the scope of this paper.

Classical MIPS R3000 processors have a special STATUS register that stores some flags and reflects the current processor state. In mppSoC, both the ACU and the PEs are an evolution of this MIPS 3000 and need some new flags. Some available spaces in this register are used for four additional flags. This design choice permits to keep the initial complexity of the processors. The added flags are the following:

- The ACU decodes the instructions and synchronously sends micro-instructions to all the PEs. All the PEs with their *E bit*, "enabled" bit, set realize the instruction while the other PEs do nothing. This basic control of the PE activity allows to implement advance control and high-level algorithms [8].
- the *T bit*, "transmission" bit, identifies if the PEs will participate or not in a send operation, either this operation involves the X-Net or the Global Router.
- the *R bit*, "reception" bit, determines if the PE has received a data during the last communication (X-Net or Global Router communication).
- the *F bit*, "fetched" bit, is only used for the Global Router fetch operations. Only PEs which have really fetched data set their F bit.

For an easier programming, we added some new instructions manipulating directly those bits (copy, logical operations, move to or from a register).

Like the MasPar, the mppSoC has a Or-Tree: a binary tree with PEs as leafs and the ACU as root. This tree systematically produces a logical OR of all the PEs E bit. A special ACU instruction can read the current Or-Tree value at any time, allowing to determine if at least one PE is still active. The newly introduced bit manipulation instructions permits to use the Or-Tree to compute any other status bit.

4. MpNoC Utilization

The mpNoC IP is basically used to connect N PEs to each other: roughly speaking, the mpNoC may be considered as a Global Router of the mppSoC.

The mpNoC has N input ports and N output ports. It is implemented by an internal network described in the next section. It usage relies on an internal protocol which does not depend on the kind of network used. As in the MasPar, the network works using circuit switched and serializes the data.

Six new instructions have been added to realized any kind of communication. The three T, R, and F bits of the PEs status register are used by the Global Router. Table 1 sums up this instructions and their arguments.

Communication Steps Using the Global Router to realize a communication needs several steps. We decided to use at least one instruction per step (instead of regrouping them) to offer more flexibility and efficiency.

Opening Channels (`r_open`) The first communication step consists in opening the channels: every PE with its T bit activated sends its destination PE, contained in one of its registers, to the Global Router. The internal network configures itself to open some communication channels. If a non-full network, such a delta network, is used, there can be some conflicts between the channels. In this case, the incompatible channel subset is not opened. At the end of the opening, opened channels destination PEs have their R bit set to 1 while the other PEs have their R bit set to 0.

Sending Data (`r_send`) Once channels have been opened, PEs with their T bit set will send their data to the Global Router. No destination is required because the channels have already been opened. Only PEs with R bit set to 1 will receive a valid data and store it in a register. Several send instructions can be done while the channel are open.

The `r_osend` instruction combines an opening of channels and an emission. Its implementation is more efficient that a `r_open` followed by a `r_send`.

Fetching Data (`r_fetch`) Emitters can also fetch data from their destination with the `r_fetch` instruction. Every PE with their R bit set sends a data to the Global Router. The data goes back into the internal network up to the source PE (thanks to the circuit switching). In case of a non-full network, it is possible that some PEs with their T bit set cannot fetch any data because of a conflict. After a fetch request, the F bit is set to 1 for each PE that successfully fetched a data, 0 for the others.

Closing Channels (`r_close`) After a communication, the channels must be closed. This operation sets to 0 the T bit of each of the PEs that was the source of an opened channel and successfully communicated with their destination. For the other PEs, a new communication (open and send) phase is required.

It is possible to close the channels just after having fetched data using the `r_fetchc` instruction that is faster than `r_fetch` followed by a `r_close`.

Communication Realization The realization of a communication, is based on the following pattern: First, the destination PE for every sender (PEs with T bit set) must be stored in a register of the sender. Then, the channel can be opened and some data can be send or fetched. Finally, the connection must be close. Figure 1 sums up those steps.

In few situations, some connections are incompatibles. In this case, it must be checked if all the channels have correctly been opened. There are two possibilities:

- No fetch is needed in the communication, so a close will set the T bit of every successfully sender to 0. While a PE has its T bit equals to 1 (which is checked with the Or-Tree), a new communication phase is needed.
- A fetch is needed in the communication. After fetching a data, every sender which successfully fetch a data from its destination set its F bit to 1. So T can be updated depending of this result. If a `fetchc` instruction can be used as the last data exchange, it will update both F and T bits.

Knowing that, it becomes very easy to realize any kinds of communications. A classical usage of the Global Router is the accumulating send, where each PE which is destination of multiple others PEs accumulates the received data. Such a communication is described in Algorithm 1. Those kind of basic communications are directly implemented in an higher level language than the extended mppSoC assembly language.

Table 1. MpNoC instruction set

`r_open` PRt *Open a Global Router communication channel. PRt contains the destination PE. Set R to 1 for each destination PE.*
`r_send` PRd,PRs *Send data via the open channels. PEs with R bit set to 1 receive a data. Dst(PRd) ← Src(PRs)*
`r_osend` PRt,PRd,PRs *Open then send data on the router.*
`r_fetch` PRd,PRs *Fetch data from the open channels. PEs with T bit set to 1 can receive a data. Src(PRd) ← Dst(PRs). Set F bit to 1 for fetching PEs.*
`r_close` *Set T bit to 0 for open channels PEs.*
`r_fetchc` PRd,PRs *Fetch data from the open channels then close channels.*
`r_mode` SRh *Set the mpNoC mode (PEs to PEs, devices to PEs, PEs to devices.*

PRx: the PE Rx register
SRx: the ACU, thus scalar, Rx register

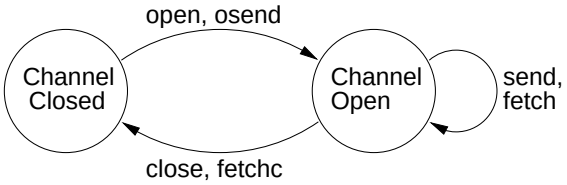

Figure 1. Communication automaton

$oldE \leftarrow E\ bit$;
$T\ bit \leftarrow E\ bit$;
$acc \leftarrow 0$;
while *At least one PE is active* **do**
  open channels;
  send snd to rcv of the destination PE;
  close channels;
  $E\ bit \leftarrow R\ bit$;
  $acc \leftarrow acc + rcv$;
  $E\ bit \leftarrow T\ bit$;
end
$E\ bit \leftarrow oldE$;

Algorithm 1: Accumulative Send

5. MpNoC IP Architecture

As detailed in Section 4, the Global Router includes an internal network which transfers data from sources to destinations. This network is the key point of mpNoC. In order to allow an efficient and a realistic IP integration of this network, its interface must be generic enough to support a configurable size (4×4, 32×32 for example). While targeting a mpNoC integration into mppSoC, the number of mpNoC sources and destinations is equal to the number of PEs used in the mppSoC grid. A second key-point for an efficient mpNoC IP integration is the configurable used of the internal network, according to the application requirement. Allowing designer to choose the internal network increases run-time performances. In the following, we present two internal networks, with their advantages and disadvantages: a full crossbar and a Extended Delta Network (EDN) [6]:

Full Crossbar This kind of network fully connects each device of the system to each other, allowing heterogeneous and one passe communications. However, its silicon cost is high and grows quadratically with the network size. Thus, this network is only suitable for little networks.

Extended Delta Network While targeting a huge network size, it is necessary to deal between efficiency (expressed in average number of passes) and the silicon space. The Extended Delta Network (EDN) is a particular delta network with good properties [5]. Many delta networks uses crossbars whereas the EDN uses hyperbars. The main idea in an EDN is to use two kinds of wires: *thin wires* and *thick wires*. *Thin wires* carries a single data while *thick wires* can carry up to K data. A hyperbar is a crossbar which uses *thick wires* instead of *thin wires*. In our EDN, *thick wires* can carry up to 2 data. We used interconnected 2×2 hyperbars to build this EDN. Finally, an EDN needs *thin-to-thick* and *thick-to-thin* converters in its extremities to use hyperbars. The right-hand side of the figure 2 illustrates an EDN.

In this section, we have proposed a mpNoC IP generic enough (in terms of size and internal network used) to offer powerful integration in various systems: from little and regular system till a huge and heterogeneous one.

6. MpNoC Integration in MppSoC

The mpNoC IP fulfils a triple function in the mppSoC architecture. Firstly, the mpNoC is used as a Global Router connecting, in parallel, any PE with another one. Secondly, the mpNoC is able to connect the PEs to the mppSoC devices, and as a special case of this, the mppSoC is able to connect the ACU to any PE of the mppSoC.

Connecting each PE of a mppSoC to one input and one output of a mpNoC allows to exchange data between any couple of PE in parallel: The mpNoC acts as a Global Router. Nevertheless, if any communication between PEs may be realized by a set of X-Net communications, as performances and flexibility are concerned, the usage of such a Global Router has an advantage over the X-Net usage for many algorithms. Consequently, including or not a Global Router in a given mppSoC design is a trade-off between the cost in term of silicon and the advantage in term of performance and flexibility, specially in the case of a design targeting a configurable hardware such as a FPGA. The nature of the targeted applications may the decisive element in this design choice.

Communication between the ACU and the PEs may take advantage of a common access to the PE memories, specially for initialization and finalization phases of algorithms. In our first design of the mppSoC, this access to the PE memory was realized by dedicated wires and hardware elements connecting the ACU to the memory of any PE. This memory being a double access memory with one great advantage: by construction, only one access may occur at a time, an instruction being either sequential or parallel. The mppSoC may be used to connect the ACU on one side with all the PEs on the other side. It thus suppresses the need of an access to the PE memories from the ACU.

A major problem of several massively architecture is their inability to respond the need of a high bandwidth of input/output. Many number cruncher applications require high amount of data. A sole sequential access to a flow of data inhibits the parallelism of a massively parallel machine and is a major drawback when performances are concerned.

The mpNoC may be used to connect, in parallel, the PEs of a mppSoC to devices. Getting data requires the input devices to be connected to the input ports of the mpNoC while the PEs are connected to the output ports of the mpNoC. Producing data needing opposite connections. While the connections are established, parallel input/output operations may deliver data from the pins of the chip to all the PE in an efficient way.

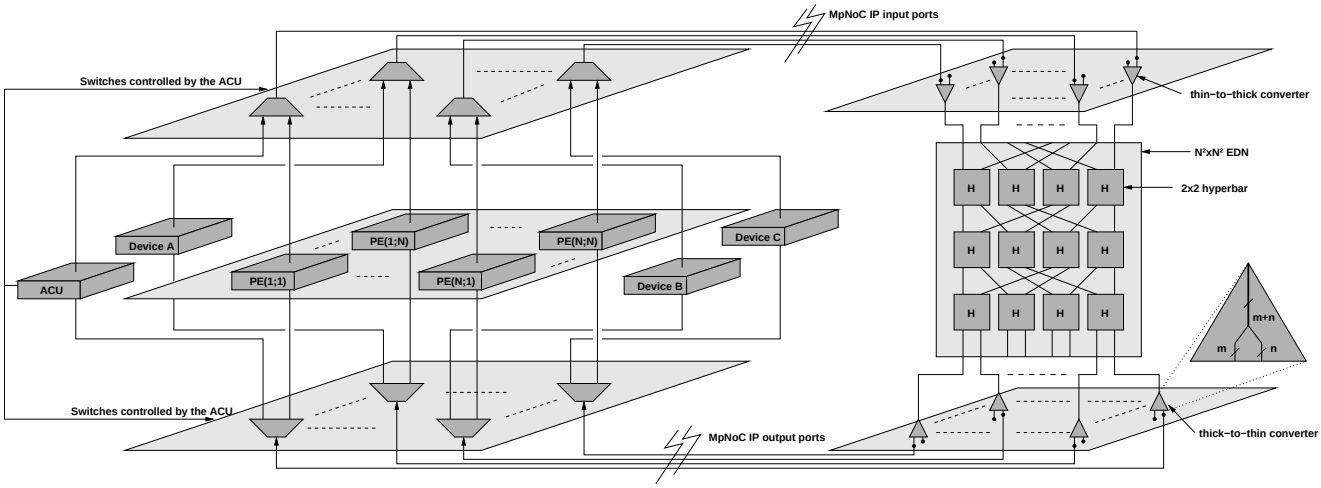

Figure 2. MpNoC integration into mppSoC The mpNoC IP and its input/output ports are shown on the right-hand side of the figure. It includes *thin-to-thick* and *thick-to-thin* converters and the EDN. The mpNoC IP is connected to mppSoC and their input/output devices via controlled switches (top left and bottom left-hand side). The mppSoC (including the PE grid and the ACU) and input/output devices are situated between those switches.

Three usages of a mpNoC IP are then required to build a mppSoC. Nevertheless, only one instance of a mpNoC is enough, the synchronous execution model of the architecture ensuring that at a given instant at most one usage of a mpNoC is necessary. Furthermore, the current usage of the mpNoC may be deduce from the current instruction issued by the ACU. Thus the mppSoC integrates an only mpNoC but this mpNoC input and output ports are connected to switches controlled by the ACU. Theses switches allow to connect either the PEs or the input/output devices and the ACU to the mpNoC, depending on a global control issued from the ACU. The figure 2 illustrates the integration of a mpNoC into mppSoC.

7. Global Router Applications

Using mpNoC as a Global Router can be very interesting to realize several kinds of communications. However, the choice of using X-Net or mpNoC in mppSoC platform for a given application is not at all obvious. In fact, using X-Net in non-regular communications is very tedious, while using mpNoC in regular ones is time and energy consuming. Thus, the designer has to make the right choice between the two networks, depending of the application, in order to optimize the whole system's performances.

Picture rotation algorithms seem to be simple and good examples to use the Global Router. In fact, in this situation, communications are very irregular: PEs need to communicate using several different directions and lengths. Obviously, it is possible to realize that using the X-Net network, but this needs several communication steps (as much as the number of PEs). We tried this rotation using the Global Router on a picture. With a PE grid of 32×32 PEs, we realized 1024-pixel picture rotations. The Global Router was using an EDN as internal network in which our rotation needs only sixteen communication steps to finish.

Figure 3. Picture rotation

This result shows that using the Global Router instead of the X-Net can be really efficient in terms of performances and easier to program. The resulting pictures of Figure 3 were provided by an execution of binary program on our SystemC mppSoC simulator.

8. Conclusion

Having an efficient communication network in modern multiprocessor systems on-chip is certainly one of the biggest challenges for designers. This is particularly true

for SIMD architectures. The mppSoC platform uses X-Net network for inter-processor communications. This network is very efficient for regular communications. However, it seems to be time and energy consuming when dealing with point-to-point and non-regular communications.

This paper presents an implementation of an efficient communication network IP called mpNoC. MpNoC can be used alone or with X-Net in a mppSoC platform. The internal architecture of mpNoC IP is very flexible. In fact, its behaviour can be implemented with a full crossbar as well as with a multistage or other kinds of networks. To make as easier and systematic as possible the integration of mpNoC, we furnish an implementation of a specific protocol, and also an extension of the MIPS 3000 processor used in our platform.

The current design of mppSoC has been implemented at the RTL level targeting a FPGA. Nevertheless the mpNoC IP is not include in this version. Thus, a RTL description of the mpNoC IP and its integration in the FPGA platform constitute a significant issue of our future works. This will enable to perform more accurate performance estimations, specially in terms of silicon area or energy consuming.

References

[1] M. Adiletta, M. Rosenbluth, D. Bernstein, G. Wolrich, and H. Wilkinson. The next generation of Intel IXP network processors. *Intel Technology Journal*, 6(3), Aug. 2002.

[2] A. Adriahantenaina, H. Charlery, A. Greiner, L. Mortiez, and C. A. Zeferino. SPIN: A scalable, packet switched, on-chip micro-network. In *Design, Automation and Test in Europe Conference and Exhibition (DATE'2003)*, pages 70–73, 2003.

[3] T. Blank. The MasPar MP-1 architecture. In *Proceedings of the IEEE Compcon Spring 1990*, pages 20–24, San Francisco, CA, Feb. 1990. IEEE Society Press.

[4] F. Karim, A. Nguyen, S. Dey, and R. Rao. On-chip communication architecture for OC-768 network processors. In *Design Automation Conference (DAC'01)*, pages 678–683, 2001.

[5] C. P. Kruskal and M. Snir. The performance of multistage interconnection networks for multiprocessors. *IEEE Transactions on Computers*, C-32(12):1091– 1098, Dec. 1983.

[6] S. Meftali, J.-L. Dekeyser, and I. D. Scherson. Scalable multistage networks for multiprocessor system-on-chip design. In *Eighth International Symposium on Parallel Architectures, Algorithms, and Networks (I-SPAN 2005)*, pages 352–357, Los Alamitos, CA, Dec. 2005.

[7] M. Millberg, E. Nilsson, R. Thid, and A. Jantsch. Guaranteed bandwidth using looped containers in temporally disjoint networks within the Nostrum network on chip. In *Design, Automation and Test in Europe Conference and Exhibition (DATE'04)*, pages 890–895 Vol.2, Feb. 2004.

[8] G.-R. Perrin and A. Darte, editors. *The Data-Parallel Programming Model*. Lecture Notes in Computer Science, Tutorial Serie, vol. 1132, 1996.

[9] Sonics Inc. Product brief: SiliconBackplane III MiroNetwork IP, May 2003. `http://www.sonicsinc.com/sonics/products/siliconbackplaneIII/productinfo/docs/siliconbackplaneIII.pdf`.