Study and Integration of a Parametric Neighbouring Interconnection Network in a Massively Parallel Architecture on FPGA

Mouna Baklouti[1,2], Mohamed Abid[1], Philippe Marquet[2], Jean Luc Dekeyser[2]
[1]*National Engineering School of Sfax, CES Laboratory, Sfax, Tunisia*
[2]*LIFL, INRIA Lille North Europe, University of Lille, Lille, France*
mouna.baklouti@lifl.fr

Abstract

Single Instruction Multiple Data processors are increasingly used in embedded systems for multimedia applications because of their area and energy-efficiency. Neighboring communications between the processing elements are a key issue in SIMD processors. They are present in most data parallel applications. However, the lack of flexibility in major parallel architectures is its main shortcoming. In order to improve the performances of a massively parallel architecture, especially in term of neighboring communication we need a flexible and parametric communication network. This paper focuses on the problems with the design of a parametric nearest neighborhood interconnection network in a SIMD architecture in System on Chip (SoC). This network can be configured in multiple topologies making it flexible and parametric in order to suit different application needs. The proposed architecture is evaluated in terms of area (cost) and performance (execution time), which are deduced respectively from synthesis and simulation results. Experiments are performed on different architectures with various topologies. In order to evaluate the performance of the proposed architecture, a FIR application is finally implemented.

1. Introduction

The complexity of embedded multimedia signal processing applications has reached a point where the performance requirements of these applications can no longer be supported by embedded system platforms based on a single processor. In this context, Single Instruction Multiple Data architectures are increasingly used. They are effective for applications that are highly parallelizable and require execution of the same operation over and over again. By using VLSI technology and replication of components effectively, massively parallel processors can achieve high performance at low cost. Key issues are how the processor and memory are partitioned and replicated, and how inter-processor communication and IO are accomplished. In this work, we emphasize on the architecture and performances of the nearest neighborhood interconnection network, essential for most SIMD processors, in order to provide an efficient and flexible one. We define also in this paper a SIMD architecture, called massively parallel processor System on Chip, which is parameterized; since its component parts are adjustable in different parameters e.g. network topology and memory size. MppSoC contains a number of processing elements (PEs) having local memories. The whole system is mastered by a particular processor called Array Controller Unit (ACU). The processors communicate using a neighborhood interconnection network. In this work, we focus on describing the nearest neighborhood interconnection network used since it presents the new component integrated on FPGA [6]. It permits regular and systematic communications between processors. The objective of this work is to provide designers with a flexible parametric neighborhood interconnection network since it can be configured in many topologies making it more efficient and well suited for multiple data parallel applications needs. In order to assure its functioning in the whole architecture, we also furnish communication instructions permitting to manage the network.

The rest of the paper is organized as follows. Section 2 gives an overview of related works. The mppSoC architecture is explained briefly in Section 3. This Section emphasizes on the integration of the neighborhood interconnection network and describes its global features. Section 4 discusses implementation results and shows an example of a representative application mapped to mppSoC. Since SIMD is exceptionally suited for image processing, the intension is to use a FIR-filter as test application on the

resulting system. Finally, conclusion and future works are presented.

2. Related works

The major SIMD machines appeared contain a neighborhood interconnection network allowing regular data communications. In [2], [4] and [5], a processor array SIMD architecture, called Massively Parallel Computer (MasPar), is presented. It has X-Net as a neighbourhood interconnection network, in a form of a 2D-mesh with wraparound. The PEs are connected via a three-way switch, meaning that every PE has 8 nearest neighbours that it can reach in only one clock cycle. In [8], a SIMD computer for array processing, ILLIAC IV, is described. It included 64 PEs where each PE is connected to four neighbours by an 8 by 8 grid interconnect. In this architecture, non neighbour communication requires extra shifts. XETAL [1] and IMAP [10] are also more recent and interesting SIMD processor examples for video processing. They consist of 320 and 256 PEs, respectively, arranged as a 1-dim, linear array. Each PE has only the ability to access its neighbours (left and right) and when it wants to get some data from its n^{th} neighbour (n>1), the corresponding data should be shifted to become accessible. In this case, the communication bottleneck may cause a significant increase in the cycle count of the program. The Real-Time, Embedded, Modular, Adaptive, Massively Parallel Processor Project (REMAP), was also a long-time project in SIMD research [7]. There are several prototypes of the REMAP. All of them are SIMD systems with some differences between versions. In REMAP-β the communication network enables two different connections: nearest-neighbour ring communication and broadcast. In the ring network each PE can read,

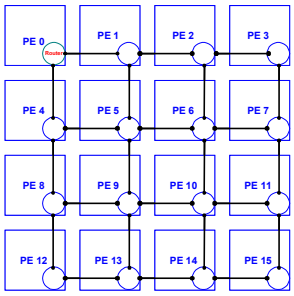

Figure 1. Mesh network topology

but not write to, its neighbouring PEs memories (both left and right). In [3], a reconfigurable parallel computer using SoC design, Morphosys, consists of a hardware implemented external processor connected to a Reconfigurable Cell (RC) array via a system bus. The RC array consists of four blocks of 4x4 matrixes making a 8x8 matrix of reconfigurable cells. The RCs are connected via a three layer network, where the first layer is a 2D-mesh with nearest-neighbour connection, and the second and third are column and row broadcasts respectively [3]. Thus, each RC can access data from any of its row/column neighbours with the nearest neighbour layer.

According to all these works, the neighbourhood interconnection networks of parallel architectures could be configured in only one topology making them no flexible and parametric in order to suit different application needs. Major of these networks allow only communication between direct neighbours which makes the communication with a non-direct nearest PE a bottleneck. Other architectures allow PEs to communicate in only one access in writing or reading. All these parallel architectures mentioned are also static in term of PE arrangement topology. Thus, an inter-PE communication network is proposed in this paper to face these limits and will be more efficient and parametric to satisfy various data parallel applications requirements.

In the following Section, we will present the mppSoC architecture and we will focus on specific issues related to the neighbourhood network integration with an impact on its performance.

3. Integration of the neighborhood interconnect in mppSoC

3.1. Basic mppSoC architecture

In this section, the mppSoC design with its all components is uncovered. In fact, mppSoC is a SIMD architecture composed of a number of processing elements (the PEs) and memories connected by a regular neighbourhood network and a general purpose global network [6]. These two types of networks perform regular and irregular communications respectively, making the communication more efficient. All the system is controlled by an Array Controller Unit (ACU) which is responsible for fetching and interpreting instructions. The ACU processor issues arithmetic and data processing instructions to the PEs, and handles any control flow or serial computation that cannot be parallelized [6]. In this architecture, multiple PEs, working in parallel, use an interconnection network for the exchange of data and control information. Efficient exchange of information across the network is crucial to overall system performance. Indeed, the choice of the interconnection network is very important in SIMD

architectures. Different data parallel applications require different interconnection networks topologies for maximum performance, particularly in term of regular communications. It is, therefore, necessary that the neighbourhood network is both effective and easy to modify to suit different applications. The work proposed emphasizes most features of this router performing regular communications.

3.2. The mppSoC neighborhood interconnection network

The idea is to form an inter PE communication network of different topologies, based on IP blocks. The network is scalable from a very basic topology such as a mesh (Figure 1), to an eight nearest neighbourhood network X-Net.

The inter-PE network is different from a general on chip NoC which allows each processor to send data to any other processor in the system. This makes it more flexible than the neighbourhood network, but slower with a significant communication overhead. Based on a neighbourhood network in SIMD architecture, all the communications take place in the same direction so we

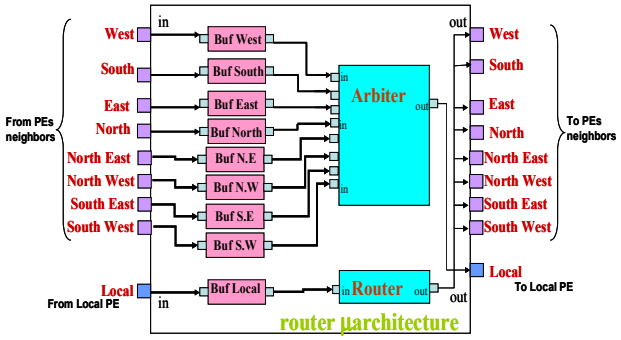

Figure 2. Router element architecture

don't have to deal with messages contesting for the same output port. This makes the router logic simpler.

The nearest neighbourhood interconnect is clocked synchronously with the PEs. Its primary function is to move data between PEs, but also to load the PEs memories with data. To achieve the high reusability needed in this work, the interconnection network consists of a controller and a number of smaller blocks, called routing elements, a number that is equal to the number of PEs. These components are connected with each other in a network topology and are responsible of routing the data. So, the routing element is the essential component of the interconnection network. It is designed as an IP that contains a switch able to communicate data to another PE in a specific direction, depending on whether the destination is, according to the network topology. This looks different depending on the kind of topology that is used but each has a connection to its PE, one or more connections to another routing elements and a controller signal. A router element used in an array network has only two connections to other routing elements, whereas one in an X-Net network can have up to ten connections. The functionality of the routing element is also independent of the network type. It receives data from its PE and another routing element and forwards it to a second routing element and the PE respectively. The way it forwards the data depends on the control signal, which is specific for each kind of topology. In fact, the control signal transmitted via the ACU is responsible of configuring the network in one desired topology. Thus, the network depends mostly on its topology and the number of PEs.

The routing element's architecture is presented in the Figure 2. This router has 8 input interfaces and 8 output interfaces dealing with synchronized communication between routing elements and the network interaction with processors. The number eight presents the maximum number of directions that one PE may need to communicate with another PE in a 2D topology and which presents the case of an Xnet network. Each comes from a different PE in a different direction (North, South, East, West, North East, North West, South East and South West). There is also an input from the local PE itself and an output to the latter. Each input is stored in a buffer which can temporarily store data packets. An arbiter manages the priorities between the different requests from neighbouring PEs and communicates them one by one to the local PE. A round-robin scheme is employed to arbitrate these requests. A router handles simultaneous input requests from the local PE and transfers them to the PE in the direction specified.

Each routing element performs wormhole routing with a packet size of 32 bits. This number is chosen to match a 32-bit embedded microprocessor. In each PE, data received from other routing elements are transferred to a PE communication block which assures storing data in memory. Each node has access to its own communication memory and to the communication memories of the neighbouring nodes depending on the network topology.

The PE communication IP allows regular data communications and their storage in memory in defined addresses. It has two inputs: the request and the data transmitted by the PE sender. Its role is to verify the destination address: in the case of a regular communication address the data is transmitted and stored in the memory at that address (Figure 3). In fact, the PE memory is partitioned into two parts: one dedicated to store local PE data, and another to store

data provided by the routing element through the neighbourhood interconnection network. These parts are delimited by a START_IO address pointer defined in the configuration file of the system and that can be modified as needed by the programmer. In order to allow sending and receiving data by the network, we use different communication instructions that will be described in the following subsection.

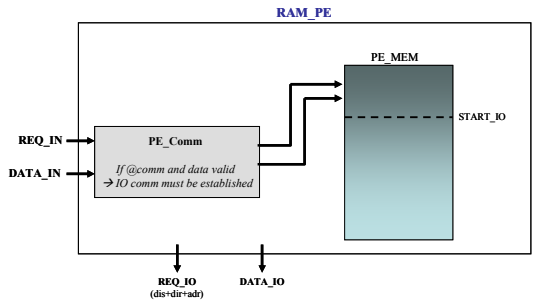

Figure 3. PE memory unit architecture

3.3. Communication instructions

The communication across the neighbourhood network is implemented via communication instructions. In fact, we use two main instructions, which are based on the processor instructions:

- SEND instruction: it consists on writing the data transmitted by communication across the network in a specific memory address.
- RECEIVE instruction: it is simply a reading of the data transferred from memory.

In our case, SEND instruction has two operands: one dedicated to the data to be transmitted and another

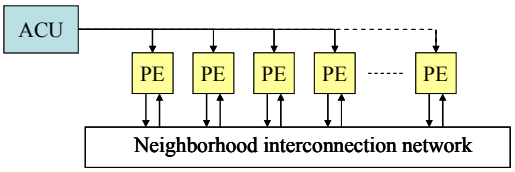

Figure 4. Massively parallel architecture mapped on FPGA

containing the distance between the PE sender and the receiver, the direction to which the data will be transmitted and the destination address in the processor receiver. The general format of the SEND instruction is the following: p_sw data, @comm_reg, where: @comm_reg=distance & direction & address.

The direction is defined depending on the regular network used, for example for a 2D torus it can be north, east, west or south. The distance defines the number of paths needed to achieve the communication between the PE sender and the other receiver. For example if one PE needs to communicate with another PE that is two blocks away (i.e, one intermediate PE) to its right, it can directly communicate by specifying a distance equal to two in the SEND instruction. In this way communication will be faster. This feature allows that each PE can get data not only from direct neighbour but also from more distant PE. In the configuration file, eight directions (the maximum neighbourhood communication links), are defined by eight constants to be used in the program and when configuring the mppSoC system.

Below, we will present implementation results obtained by mapping the mppSoC architecture described on FPGA.

4. Experimental results

4.1. Implementation results

The architecture implemented on FPGA is presented by Figure 4. The logic on FPGA is programmable, this has made it possible to adapt the hardware design to a specific application, which makes testing and debugging the system a relatively easy task. Performance and size are instantly available when running the implementation, thus comparison between different implementations becomes less complicated.

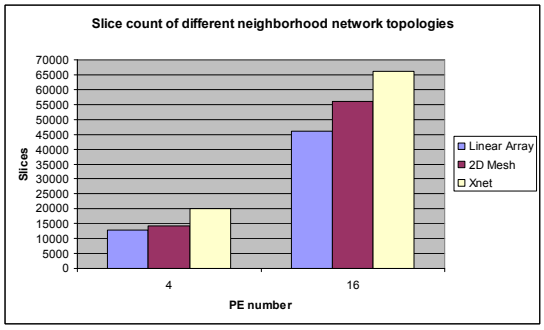

Figure 5. Synthesis results

An evaluation has been performed to measure the size and complexity of the system when configuring three types of neighbourhood networks. The topologies tested are: a linear array, with the ability to shift data left or right, a 2D mesh topology, where data can be transmitted left, right, up and down and finally an Xnet topology where data can be transmitted in eight different directions.

These topologies were implemented with 4 and 16 routing elements. As shown in Figure 5, the Xnet

network is considerably larger, especially in the implementation with 16 PEs, followed by the 2D mesh then the linear array network. The gap between the different surfaces rates presented in the Figure 5 is more significant with the increasing number of PEs. In fact, each interconnection network implemented with 16 PEs occupies an area three times larger than when implemented with 4 PEs. It is therefore likely that this difference in size will increase further when implementing larger networks with e.g. 64 PEs.

4.2. FIR application

Finite Impulse Response (FIR) filter application [9] is used for testing the functionality and performance of the PEs and the inter-PE router. In fact, a large FIR filter is well-suited for SIMD execution since it requires many identical multiply-accumulate operations that can be run in parallel. In this work an adapted version of the difference equation called the Direct Form Structure (Figure 6), is implemented. To run the FIR-filter application, a system containing an ACU, a neighbourhood interconnection network and 4 PEs were implemented with 4KB memory per PE. We have constructed different architectures with three networks topologies in order to find the best architecture for the FIR algorithm.

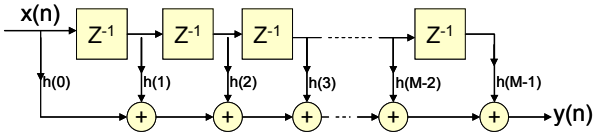

Figure 6. Direct form of FIR filter

Application results describe the performance and run time of the implemented FIR-filter. Shown in Figure 7 are the cycles needed when running the FIR filter application. The X-axis shows the type of the interconnection used and the Y-axis represents the time execution. The result is based on an impulse response with a length of four which in the chosen implementation requires four multiplications and three additions per output signal. The rest of the instructions are overhead in form of communication and memory instructions.

As expected, the linear architecture based on an array neighbourhood interconnection topology is the most effective for the FIR application. We deduced also that the 2D architecture based on an Xnet interconnection topology is better than with a mesh network. These results show that, based on the flexible neighbourhood network in the mppSoC system, the programmer would use the best interconnect suited to his application. This work presents a first step towards the mppSoC architecture exploration.

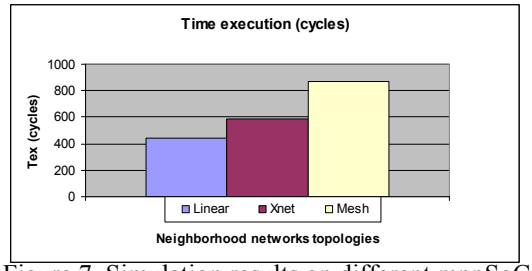

Figure 7. Simulation results on different mppSoC configurations

5. Conclusion

In this paper, we introduced briefly the mppSoC system which is a SIMD massively parallel architecture on chip. Among its important features, we emphasize on the flexible neighbourhood interconnection network. In fact, communication networks are essential for a multi-processor architecture. The regular network is important to manage regular communications present in most data parallel applications. The network designed is scalable and parametric supporting various topologies needed to satisfy different data parallel application requirements. Communication instructions are also defined to assure the functioning of the inter PE network.

The mppSoC architecture can be configured to use different sizes and network topologies. This configurability makes it possible to tailor the architecture for a specific application and thereby increasing its effectiveness. To evaluate the proposed neighbourhood network designed we have implemented different architectures with various configurations for a FIR application. According to the performances obtained for each architecture, we can choose the most appropriate one for the application tested.

More extensive testing with several different applications is ongoing. Future works aim at providing a methodology to help the designer to perform an architecture exploration for a given application.

6. References

[1] A. Abbo, and R. Kleihorst, “Smart Cameras: Architectural Challenges,” Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS), Gent, Belgium, September 2002, pp. 6–13.

[2] E. Zehendner, “Simulating systolic arrays on maspar machines,” EUROMICRO, 1997, pp. 394–401.

[3] G. Lu, H. Singh, M.-H. Lee, F. F. E. Bagherzadeh, N.and Kurdahi, and V. Castro-Alves, "The morphosys dynamically reconfigurable system-on-chip," Evolvable Hard, 1999, pp. 152–160.

[4] J. Nickolls, "The design of the maspar MP-1: a cost effective massively parallel computer," 1990, pp. 25–28.

[5] J. N. Kalamatianos, and E. Manolakos, "Parallel computation of higher order moments on the maspar-1 machine," International Conference on Acoustics, Speech, and Signal Processing, ICASSP-95. Vol 3, May 1995, pp. 1832–1835.

[6] M. Baklouti, S. Le Beux, P. Marquet, M. Abid, and J-L. Dekeyser, "Implementation of a Simple Massively Parallel Processor System on FPGA: Case Study," ICESCA, Tunisia, 2008.

[7] M. Taveniku, and A. Linde, "A reconfigurable SIMD computer for artificial neural networks," Chalmers University of technology, Tech. Rep, -ISBN 0-13-373762- 4, 1995.

[8] R. Michael Hord, "The ILLIAC IV, the first supercomputer," Computer Science Press, 1982.

[9] S. M. Kuo, and W. S. Gan, "Digital Signal Processors Architectures," Implementations and Applications. Prentice Hall, 2005.

[10] Y. Fujita, S. Kyo, N. Yamashita, and S. Okazaki, "A 10 GIPS SIMD Processor for PC-based Real-Time Vision Applications Architecture, Algorithm Implementation and Language Support," Proceedings of the Computer Architectures for Machine Perception (CAMP), Washington, DC, USA, October 1997, pp. 22–32.