Impact of Interconnection Networks in a Massively Parallel FPGA Architecture on a Parallel Reduction Algorithm

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

Abstract

As the size, hardware complexity, and programming diversity of parallel systems continue to evolve, the range of alternatives for implementing a task on these systems grows. Choosing a parallel algorithm and implementation becomes an important decision, and the choice has a significant impact on the execution time of the application. This paper focuses on the implementation of a SIMD parallel reduction algorithm in a massively parallel architecture on FPGA. In fact, parallel reduction is a common and important data parallel primitive. The impact of the interconnection network topology on the number of data transfers required to perform the computations is studied. This paper introduces also two flexible and parametric communication networks, integrated in a SIMD SoC architecture, to manage both regular and irregular communications. The programmer can choose one or both networks when configuring his architecture in order to choose the most appropriate one for a given application. The performance of executing the reduction algorithm on the proposed architecture is finally evaluated. The goal of this work is to highlight some implementation decisions that influence the overall performance of a parallel algorithm. We conclude that the massively parallel interconnection network used has a great impact on the performance of a data parallel algorithm.

1. Introduction

The last decade has seen a revolution in large scale computation. The massively parallel processing (MPP) paradigm, originally seen as an outsider in the supercomputer race, is widely recognized as the technology of the future. Since recently, we are also able to integrate billions transistors in a single chip (giga and tera-scale integration {\it GSI/TSI}). FPGAs, for example, are well suited as building block for special purpose hardware in the field of compute intensive applications.

In this work, we focus on SIMD architectures and data-parallel algorithms. We call these algorithms data parallel algorithms because their parallelism comes from simultaneous operations across large sets of data, rather than from multiple threads of control. The SIMD parallel algorithm discussed in this paper is the reduction algorithm. Many problems require that all the elements of a data set be combined in some fashion, e.g., the sum or product of all elements of an array. These operations are known as reduction operations, i.e., a single value is computed as the result of an operation on the data set. Parallel reduction operations can only be applied to associative operations, e.g., sum, product, min, and max. SIMD architectures are powerful executers especially for data intensive calculations. One of the most debatable aspects of a SIMD architecture however, is its communication infrastructure. In fact, communication between the processing elements (PEs) in a SIMD processor has remained a cause of inefficiency. In the design of parallel algorithms, especially those involving various collective communication patterns, the underlying communication technology plays a vital role in determining the communication efficiency of these algorithms. There are two classes of communication architectures that existing SIMD processors employ. The first one is the locally connected SIMD architecture. Each PE can receive data from its direct neighbors. However, many image processing algorithms for example need to get data not only from direct neighbors, but also from more distant PEs. This is performed through successive shifts which severely reduce performance. So, this communication architecture is cheap but rather inefficient. To overcome this inefficiency, a fully connected SIMD architecture is implemented. Each PE can send/receive data to/from all other PEs by means of a fully connected crossbar switch. The problem of this

architecture is that when the number of PEs increases too much (e.g., more than 64 PEs), the area of this communication network dominates the whole chip area [3]. This solution leads sometimes to an excessive communication area cost and scalability problems. So this is not feasible for massively parallel SIMD architectures. This network is also not efficient to manage direct neighbour communications.

In this paper, we introduce a SIMD parallel architecture, called massively parallel processor System on Chip, mppSoC, with two types of communication networks, making the communication more efficient. MppSoC looks like an on-chip version of the famous MasPar. It 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 are connected by a regular neighbourhood network and a general purpose global network. These two types of networks perform regular and irregular communications respectively.

The objective of this work is to show the impact of parallel interconnection networks when executing a data parallel algorithm, in our case a reduction algorithm. Providing designers with flexible parametric parallel networks, able to be configured in different topologies, makes them more efficient and well suited for multiple data parallel applications needs. The programmer has also the choice to use the network needed for a given application. Indeed, the choice of the communication network is very important in SIMD architectures. Different data parallel applications require different interconnection networks topologies for maximum performance. It is, therefore, necessary that the network is both effective and easy to modify to suit different applications.

The rest of the paper is organized as follows: In Section 2, we give an overview of related work. The mppSoC communication networks are introduced in Section 3. The parallel reduction algorithm is described in Section 4. Experimental results of executing this algorithm on mppSoC is then discussed. Finally, Section 5 concludes this paper with a brief outlook on future works.

2. Related works

Several commercial SIMD machines were introduced in the 1970s [10], but they were not widely used. Interest in this class of machines was renewed in the early 1980s with the introduction of the ILLIAC IV [8], the Connection Machine (CM-1) [15], and the MasPar MP-1 [14]. The ILLIAC IV was a SIMD computer for array processing. It included 64 PEs. An 8 by 8 grid interconnect joined each PE to 4 neighbors. Non-neighbor communication requires extra shifts. There is no another network to manage point to point communications. In the MasPar MP-1, there are two separate communication systems, and programmers can alternate between them to choose the best performance for different parts of their algorithms. One interconnection network is known as X-net. It connects each processor to its 8 nearest neighbors in a 2D torus. The other connection is a global router, which provides point-to-point communication between each two PEs. The router is implemented by a three-stage switching network, where each stage in a 1024-processor machine contains a crossbar, together; the three stages comprise a crossbar. XETAL [1] and IMAP [5] are 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 neighbors (left and right) and when it wants to get some data from its n^{th} neighbor ($n > 1$), 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. Imagine [9] is a SIMD architecture which consists of 8 PEs (each PE is a VLIW). Each PE has the ability to get data from all PEs using a fully connected network. This architecture is not scalable. If the number of PEs is increased beyond 64, the area related to this communication network will dominate the total area. The Morphosys [11] proposed dynamically reconfigurable SoC architecture. It includes an array of reconfigurable cells working in SIMD fashion and contains a sophisticated programmable tri-level interconnect network. This gives efficient regular applications, but unfortunately non-neighbours communications seem to be tedious and time consuming. Programmable parallel processor architecture in FPGAs for image processing sensors is also described in [12]. This work presents parallel processor SIMD architecture. It is applicable for multiple object detection in industrial image processing. The processors are 1bit PEs connected by a NEWS network, so that each PE has four inputs and four outputs, and controlled by a central unit.

It seems that a SIMD processor needs either many cycles to perform non neighbour communication, or a very rich interconnection structure with a high cost. In contrast, we propose a flexible architecture with parametric and scalable networks in order to perform different communication types and to suit diverse application needs.

3. Communication networks in mppSoC

In this section, we explain briefly the global mppSoC architecture. We detail then the communication networks integrated. MppSoC is a SIMD massively parallel processor architecture composed of a number of processor elements (the PEs), working in a perfect synchronization. A small amount of local and private memory is attached to each PE. The latter is potentially connected to its neighbours via a regular network. Furthermore, each PE is connected to an entry of mpNoC, a massively parallel Network on Chip that potentially connects each PE to one another, performing efficient irregular communications. The system operating is organized by a control processor (the ACU, Array Control Unit) that synchronously controls the whole mppSoC architecture [2].

Figure 1 shows the mppSoC architecture that can overcome the communication bottleneck explained in the previous sections. In this architecture, there are two communication networks in order to assure neighbour as well as point to point communications.

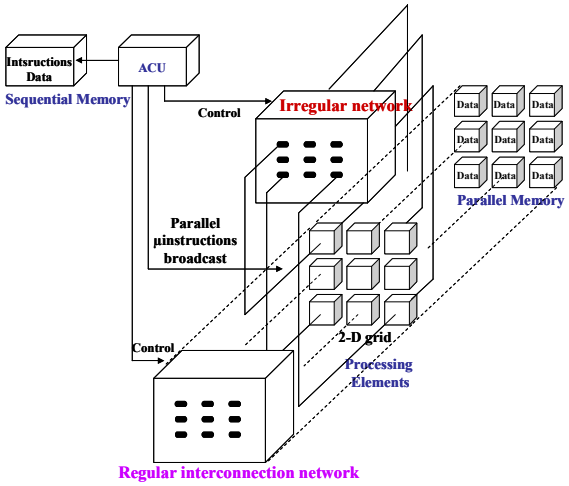

Figure 1. The mppSoC architecture

One desired mppSoC configuration is set at compile time. In fact, our prototype generated an FPGA configuration parameterized by the number of PEs, the amount of local memory attached to each PE, the type of the communication network and its topology. This parameterization makes it possible to tailor the architecture for a specific application and thereby increasing its effectiveness.

In order to improve the performances of a massively parallel architecture, and to satisfy the requirements of different data parallel applications we need scalable and parametric communication networks. In fact, the choice of the interconnection network is of great importance in SIMD architectures. When constructing a SIMD system, a large part revolves around the interconnection network, and specifically its topology. Different applications require different interconnection networks for maximum performance. It is, therefore, necessary that the network is both effective and easy to modify to suit different applications.

3.1. The nearest neighbourhood network

This network permits regular and systematic communications between processors. In massively parallel architectures, there are different regular network topologies (mesh, linear, ring, Torus, Xnet, etc.). The neighbourhood network introduced is parametric in terms of topology and the number of PEs. It is also scalable from a very basic topology such as a mesh, to an eight nearest neighbourhood network X-Net.

The nearest neighbourhood network is different form the network performing point to point communications, since it is faster with a less significant communication overhead. In this case, all the communications take place in the same direction so we don't have to deal with messages contesting for the same output port. This makes the router logic simpler. One PE can communicate, not only to his direct neighbour, but also to more distant PEs. This is achieved by indicating the distance, presenting the number of paths, between the sender and the receiver in the instruction field when programming regular communication.

3.2. The mpNoC

Point-to-point communications are more tedious and difficult. In fact, data transfers to and from the array are often a bottleneck in parallel systems. Consequently an irregular network, called massively parallel network on chip, is also integrated on the mppSoC. It permits non-regular communications between PEs in an efficient way.

The mpNoC can perform different communication modes independently of the interconnection network used. In fact, the mpNoC contains an interconnection network that can be configured in various types, responsible of routing data. The mpNoC input and output ports are connected to switches controlled by the ACU. Theses switches allow connecting either the PEs or the input/output devices and the ACU to the mpNoC, depending on a global control issued from the ACU.

The mpNoC is scalable from a very basic topology such as a bus, to a complete all-to-all network like a

crossbar. It is also parameterized in terms of the number of PEs connected to the network and the communication mode chosen. In fact, the mpNoC is basically used to connect N PEs to each other. It has N input ports and N output ports. It is implemented by an internal network and uses an internal protocol which does not depend on the kind of the network used. The internal network transfers data from sources to destinations. This network is the key point of mpNoC. A second key-point for efficient mpNoC 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.

For both types of networks, communication is implemented via SEND and RECEIVE instructions which derive from the processor instructions used, namely store and load instructions. So, we have proposed a mpNoC 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. Including or not a mpNoC 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. The nature of the targeted applications may be the decisive element in this design choice. In the next section, we describe the parallel reduction algorithm tested. The impact of a particular interconnection network on the time complexity of the parallel algorithm is also discussed.

4. Experimental results on reduction algorithm

In this section we describe the SIMD reduction algorithm implemented and present experimental results when executing it with different mppSoC configurations.

4.1. Reduction algorithm description

The reduction application [13] is used for testing the functionality and performance of the PEs and the communication networks. It presents one basic image processing operations. When reduction computation is conducted in parallel, it is known that the computation can be completed with the minimum number of steps using a binary-tree representation as shown in Figure 2.

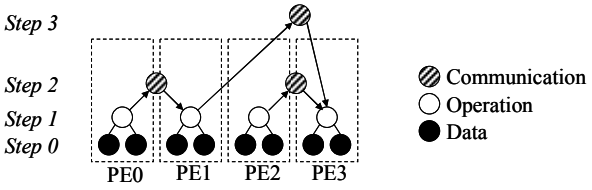

Figure 2. Reduction computation using four Processing Elements

To implement the reduction algorithm we use the recursive doubling procedure, sometimes also called tree summing. It is an algorithm that combines a set of operands distributed across PEs [14]. Consider the example of finding the sum of M numbers. Sequentially, this requires one load and M-1 additions, or approximately M additions. However, if these M numbers are distributed across N = M PEs, the parallel summing procedure requires $\log_2(N)$ transfer-add steps, where a transfer-add is composed of the transfer of a partial sum to a PE and the addition of that partial sum to the PE's local sum.

We execute the parallel reduction algorithm [6] on different sized mppSoC configurations in order to find the best mppSoC architecture for this algorithm.

4.2. Evaluation results

In this work, we focus in the manner of communication between PEs in order to communicate partial results. We have constructed different sized architectures (4 and 16 PEs), arranged in different topologies (2D and linear), and based on different networks (mesh neighbourhood network, array neighbourhood network and mpNoC with a crossbar interconnection network). In fact, the same data distribution across the PEs is used to compare the performance of the reduction algorithm on three parallel machines with topologically distinct interconnection networks. Performances of executing the reduction algorithm on these architectures are then compared. Application results describe the performance and run time of the implemented reduction algorithm. Shown in Figure 3 are the cycles needed when running the application. The X-axis shows the type of the interconnection used and the Y-axis represents the time execution.

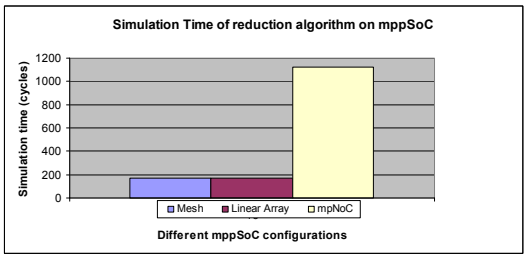

Figure 3. Simulation results on different mppSoC configurations

When implementing an algorithm on a parallel machine, the goal is to minimize the execution time. Factors that, in general, help to do this include decreasing inter-PE communications and balancing the workload among the PEs so that as many PEs as possible are concurrently executing the algorithm. We note also that the architecture of the parallel machine used has a great impact on the speedup of a given algorithm.

We notice that the architecture based on the regular network is the most effective for this type of application. In the case of a completely-connected topology, the speedup is six times lower than when using a mesh inter-processor network. The two regular topologies, mesh as well as linear array, give approximately the same simulation time. Indeed, the simulation time obtained with a linear router (170 cycles) is slightly lower than with a mesh router (172 cycles). This is due to the additional communication overhead introduced by the mesh router. These experiments demonstrate that the reduction algorithm needs one mppSoC system with only a neighbourhood network, preferably configured in a linear topology. These different results show also the flexibility of the mppSoC architecture described and facilitate the architectural exploration in order to choose the best mppSoC configuration for a given application.

5. Conclusion

In this paper, we illustrate the impact of different interconnection network topologies while holding all other factors identical, by executing the same SIMD reduction algorithm on a SIMD massively parallel architecture called mppSoC. It presents also an implementation of efficient communication networks in the mppSoC architecture. 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 irregular network is efficient for point to point communications. The two networks designed are scalable and parametric supporting various topologies needed to satisfy different data parallel application requirements. 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.

To evaluate the overall architecture, a reduction algorithm is implemented on different sized architectures with three configurations depending on the network type. According to the performances obtained, we can choose the best configuration for the application tested. The designer can choose one or both networks in a mppSoC platform. These results show that in designing algorithms for large-scale parallel machines, many issues must be addressed to devise an effective implementation, in particular the parallel architecture used. They present, in addition, a first step towards building a flexible architecture that can satisfy different application requirements.

Future work will involve the development of a complete tool chain facilitating the mppSoC architectural exploration.

6. References

[1] A. Abbo, and R. Kleihorst, "Smart Cameras: Architectural Challenges", *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS)*, September 2002, pp. 6-13.

[2] 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", *In First International Conference on Embedded Systems and Critical Applications ICESCA*, Tunis, Tunisia, 2008.

[3] H. Fatemi, H. Corporaal, T. Basten, R. Kleihorst, and P. Jonker, "Designing area and performance constrained SIMD/VLIW image processing architectures", *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS)*, September 2005, pp. 689-696.

[4] J. Fischer, and J. Dorband, "Applications of the MasPar MP-1 at NASA/Goddard", *Proceedings of COMPCON,* 1991, pp. 278-282.

[5] 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)*, October 1997, pp. 22-32.

[6] M. Harrist, "NVIDIA: Optimizing parallel reduction in CUDA", Available at http://astro.pas.rochester.edu/aquillen/ gpuworkshop/ reduction.pdf.

[7] R. S. Hogg, D. W. Llyod, and W. I. Hughes, "Communication techniques for a self-timed massively parallel architecture", *In First International Conference on Massively Parallel Computing Systems,* Ischia, Italy, May 1994.

[8] R. M. Hord, “The ILLIAC IV, the first supercomputer”, *Computer Science Press*, 1982.

[9] B. Khailany,W. J. Dally, S. Rixner, U. J. Kapasi, P. Mattson, J. Namkoong, J. D. Owens, B. Towles, and A. Chang, “Imagine: Media processing with streams”, IEEE Micro, April 2001, pp. 35-46.

[10] D. J. Kuck, “A survey of parallel machine organization and programming”, *ACM Comput. Surv*, 1977, pp. 29–59.

[11] M.-H. Lee, H. Singh, G. Lu, N. Bagherzadeh, F. J. Kurdahi, E. M. C. Filho, and V. C. Alves, “Design and implementation of the MorphoSys reconfigurable computing processor”, *The Journal of VLSI Signal Processing*, 2000, pp. 147–164.

[12] F. Schurz, and D. Fey, “A programmable parallel processor architecture in FPGAs for image processing sensors”, *Integrated Design and Process Technology (IDPT)*, 2007.

[13] H. J. Siege1, L. Wang, J. E. So, and M. Maheswaran, “Data parallel algorithms”, *ECE Technical Reports*, 1994.

[14] H. S. Stone, “Parallel computers”, *Introduction to Computer Architecture*, 1980.

[15] S. A. Zenios, and R. A. Lasken, “The Connection Machines CM-1 and CM-2: solving nonlinear network problems”, *Proceedings of the 2nd International Conference on Supercomputing (ICS)*, 1988, pp. 648–658.