IP based configurable SIMD massively parallel SoC

Mouna Baklouti* and Mohamed Abid
CES Laboratory
ENIS School
Sfax - Tunisia
Email: mohamed.abid@enis.rnu.tn

Philippe Marquet and Jean Luc Dekeyser
*Univ. Lille, F-59044, Villeneuve dascq, France
LIFL, Univ. Lille 1, F-59650, Villeneuve dascq, France
INRIA Lille Nord Europe, F-59650, Villeneuve dascq, France
UMR 8022, CNRS, F-59650, Villeneuve dascq, France
fmouna.baklouti,philippe.marquet,jean-luc.dekeyserg@lifl.fr

***Abstract*—Significant advances in the field of configurable computing have enabled parallel processing within a single Field-Programmable Gate Array (FPGA) chip. This paper presents the implementation of a flexible and programmable Single Instruction Multiple Data (SIMD) processing system on FPGA that can be adapted to the application. Its implementation is based on an IP (Intellectual Property) assembling approach making its design fast and easy. A generation tool is also developed to generate the SIMD configuration depending on the application requirements. The proposed parallel processing system on chip is portable, scalable and flexible since it can be customized to match the needs of a data parallel application. Based on FPGA, different SIMD configurations have been evaluated in terms of performance and area trade-offs. The proposed parametric system shows good results executing some signal processing applications such as parallel matrices multiplication, FIR filter and RGB to YIQ image color conversion.**

I. Introduction

Intensive signal or image processing covers a variety of applications where speed and accuracy matters a lot. Massively parallel architectures, in particular Single Instruction Multiple Data (SIMD) architectures are proposed to be good executers. In fact, the SIMD architectures are widely recognized as being well-suited for media-centric applications since they can efficiently exploit massive data parallelism available with minimal energy [2].

However in the nineties, this class of processors has faced many challenges due to its increasing fabrication cost and design complexity [1]. Nowadays, the recent great evolution of silicon integration technology on the one hand, and the wide usage of reusable Intellectual Property (IP) cores on the other hand, are more and more adopted to rise these challenges and reduce the time to market. This context substantially alleviates the design cost of a dedicated processor for a SIMD system. In addition with IP reuse we can easily and rapidly design a parallel architecture that can be tuned according to the application requirements.

Several SIMD architectures have been proposed to exploit data intensive parallelism. They can be partitioned into hardware specific solutions [11], [12], or reconfigurable solutions [7]. However, they are hard to implement and they are optimized for specific applications. This normally results in good performance for the targeted application; however the performance of other applications may not be so good due to the diversity of parallel application requirements. Thus, current developments of parallel architectures often fail to deliver the needed performance across different parallel applications.

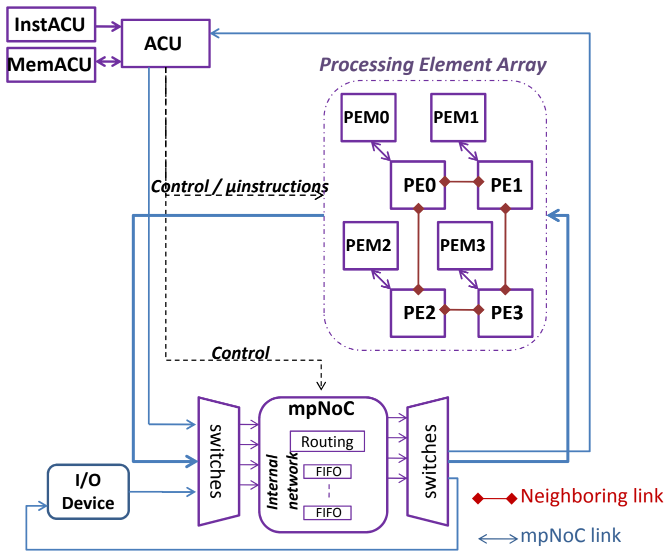


Fig. 1. MppSoC system.

For all those reasons, our goal is to design a programmable and flexible parallel SIMD processing on chip architecture, named mppSoC (massively parallel processing System on Chip), based on an IP reuse approach. MppSoC is extensible, configurable and can be customized to match the needs of a data parallel application.

We target FPGA devices to implement and test our proposed design. Compared to ASIC, FPGA requires much less development costs which is a very important issue. FPGA devices are characterized by their increased capacity, smaller NRE (non-recurring engineering) costs, and programmability. They can also be re-wired and remotely reconfigured at any time helping us to easily test different mppSoC configurations.

By providing tools to guarantee the architectural mppSoC flexibility, the design can be used to a variety of data intensive signal processing applications. The advantages of this approach are to reduce development costs and time to market and to be able to rapidly implement a SIMD on chip system adapted to the application. Both speed and flexibility are of paramount importance for parallel competitive systems. The next section describes the mppSoC system and focuses on

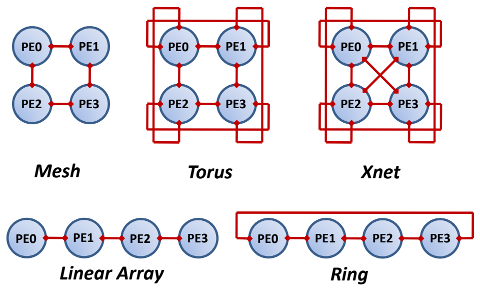


Fig. 2. Neighbourhood network topologies.

the proposed IP based assembling methodology to implement one mppSoC configuration on FPGA. It also introduces the developed mppSoC tool. Section 3 presents the FPGA based synthesis results of some mppSoC configurations and discusses the experimental results showing the performance of the mppSoC system. Finally, section 4 concludes this paper with a brief outlook on future work.

II. Flexible mppSoC design

As a basic model design [4], the SIMD FPGA architecture (Fig 1) is composed of a main controller ACU (Array Controller Unit) connected to its sequential instruction and data memories named InstACU and MemACU respectively, and a number of elementary processors (PEs). Each PE is connected to its local private data memory. We distinguish two mppSoC networks: a neighbourhood interconnection network to assure neighbour communications between PEs and a global router, called a massively parallel Network on Chip, mpNoC, to perform point to point communications. The designer can use none, one or both routers to generate the needed mppSoC configuration.

A particularly important feature of the mppSoC is the ability to target diverse applications by customizing the basic architecture. This customization is achieved with its parametrization as well as its extensibility and reconfigurability. In fact, the mppSoC is parametric in terms of the number of PEs as well as the memory size. In addition, the neighbourhood network can be configured at compile time to have one topology among five (2D (mesh, torus, Xnet) and 1D (linear, array)) as illustrated in the figure 2.

Furthermore, the mpNoC is designed as an IP which synchronously connects a set of inputs to a set of outputs. It is based on an internal interconnection network that can be chosen at compile time (shared bus, crossbar, multi-stages, etc.) according to the applications requirements. Allowing designer to choose the internal network increases run-time performances. The mpNoC assures irregular communications between processors and also performs parallel I/O data transfer which is clearly a key issue in a SIMD system. The mpNoC can be configured at run-time to support one of the three different bidirectional communication modes (PE-PE, ACU-PE, I/O Device-PE) making it powerful and suitable for parallel systems [6].

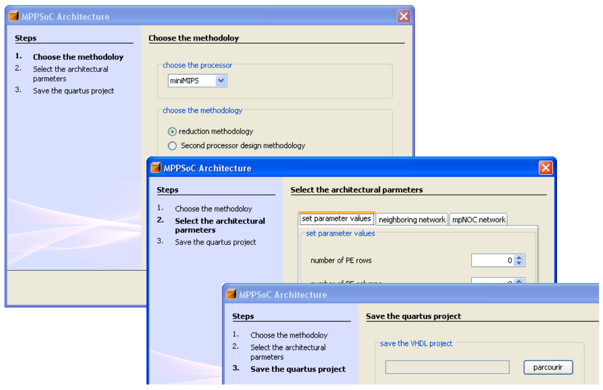
Programmability of the architecture is ensured in the design by performing a tool chain to generate binary data parallel programs that ensures multiple applications can be mapped onto the generated hardware. An mppSoC generation tool is also developed to generate the VHDL synthesizable code of the chosen mppSoC configuration. This tool is based on an mppSoC IP library and uses a VHDL configuration file that instantiates the mppSoC parameters. This file can be modified as required by the designer.

The architecture is designed as an assembling of various components or IPs, including processors (ACU + PE), memories and networks (neighborhood network and mpNoC). These IPs are partitioned into standard IPs such as processor IP, memory IP and interconnection network IP, and mppSoC dedicated IPs provided by the mppSoC tool. Some standard IPs are furnished in a HW library to alleviate their design. The designer can also choose his own IP. For that purpose, a descriptive manual can help him to connect one new IP to other mppSoC components. MppSoC dedicated IPs, named IP glues or ad-hoc IPs, must always be used to construct the architecture such as IP controller integrated with the use of a global router to assure the synchronization of the architecture functioning. These IPs are automatically generated by the mppSoC tool when needed.

To build the mppSoC processors, namely ACU and PE, two assembling methodologies are proposed. The first one, called *reduction methodology*, is based on the reduction of the main processor in order to have a small reduced one [5]. This significant gain allows integrating a large number of PE on a single chip. In the second methodology, called *replication methodology*, the PE is constructed by the same processor as the ACU to reduce the design time and make the architecture assembling faster. This methodology offers a large gain in the development time since it is easy. However, we are unable in this case to integrate many PEs on a single chip. The criterion for using this methodology is to choose a smaller processor that can be fitted in large quantities in one FPGA. So, there is a compromise between the two processor assembling methodologies and the designer has to choose the most appropriate one satisfying his needs.

In order to generate one mppSoC configuration, an mppSoC generation tool has been developed. It helps the designer choosing a SIMD configuration at a high abstraction level and automatically generating its source code. The generated system can be then directly prototyped or simulated using synthesis and simulation tools (such as the Quartus synthesis tool for Altera devices and Modelsim tool for simulation).

To validate the proposed mppSoC IP assembling design, different data parallel algorithms have been tested on mppSoC. Using the mppSoC tool, the designer can easily implement various mppSoC configurations in order to choose the most appropriate and efficient one for a given application. These experiments are evaluated with regard to their performance and cost.



choose the processor: miniMIPS

choose the methodology: [✓ reduction methodology] Second processor design methodology

number of PE rows: 0

Fig. 3. mppSoC generation tool interface.

III. Experimentation on FPGA

Different data parallel applications have been executed on mppSoC to test its performance. They are written in MIPS assembly code in the case of the minimips processor and C code when using the NIOS processor. Each mppSoC configuration is generated, based on the mppSoC generation tool, in synthesizable VHDL code. It is then prototyped on the Altera Stratix 2S180 FPGA with 179k logic elements. The mppSoC is also portable and can be efficiently implemented on any other FPGA architecture. The next section describes the generation tool and enumerates the steps needed to generate an mppSoC configuration.

A. mppSoC generation tool

The mppSoC generation tool is based on an mppSoC IP library that contains some IPs to be directly integrated in the mppSoC. The library is composed of processors, memories and interconnection networks. Different IPs are provided in order to alleviate their design and to accelerate the generation process. The tool has a simple GUI interface, as illustrated in the figure 3, enabling the user to select an mppSoC configuration and generate its VHDL code. It contains different steps in which the designer has to make various choices:

1) select the processor IP from the mppSoC IP library (we can actually choose among three available processors: minimips [14], MIPS [15] and NIOS [16]);
2) choose the processor assembling methodology (reduction/replication);
3) choose the memory IP from the IP library;
4) choose the mppSoC architectural parameters namely the number of PEs and their arrangement (1D or 2D grid) as well as the memory size;
5) select/no the neighbourhood network and its topology. In this step the user can choose among linear or ring topologies if he has chosen a 1D PE arrangement or among torus, mesh or Xnet if the PEs are placed in a 2D grid;
6) select/no the mpNoC interconnection network (shared bus, crossbar or Delta multi-stage network (omega, baseline or butterfly)).

By this way, the designer can generate the configuration satisfying his needs. We notice that the developed generation tool facilitates the mppSoC design and that any modification in the configuration could be easily done.

B. Synthesis results

Using the mppSoC tool, it was easy to prototype different configurations on the FPGA Stratix 2S180 device which includes 143520 ALUTs for hardware logic [13]. The table I shows some synthesis results varying the mppSoC parameters as well as its integrated components. The processor used in these designs is the minimips [14].

We clearly notice that the processor replication consumes more FPGA area than the processor reduction. With the latter methodology, we can put a large number of PEs on a single chip (up to 84 PEs on the Stratix 2S180). We also deduce the impact of the integrated network in the FPGA resources. Comparing for example between the first and the second configurations, it is clearly shown that the crossbar consumes more FPGA logic than the omega network. The neighbourhood network also consumes fewer FPGA resources than the mpNoC. Thus, depending on his needs the designer can integrate the needed components in the selected mppSoC configuration. The mppSoC generation tool significantly facilitates the mppSoC design and rapidly allows the modification of the SIMD configuration. It just takes fewer seconds to generate an mppSoC configuration.

C. Application experimental results

This section gives better insight on the performance of the proposed flexible mppSoC system. The table II presents the execution time results running different data parallel applications on mppSoC configurations and compared to other systems. The tested mppSoC configurations are obtained varying the parameters and the used networks in order to find the most adequate configuration.

From these experiments we demonstrate the effectiveness of mppSoC to compute data processing applications. In fact, the mppSoC has shown better results running a parallel matrices multiplication compared to an FPGA based parallel accelerator for matrix multiplication [8]. This proves the efficiency of the parallel proposed SoC. The mppSoC performance also compares favourably to that of a multi-FPGA (eleven Virtex II) based custom hardware design [10]. So, the mppSoC system not only achieves better execution time than dedicated hardware systems but also presents the advantage of programmability to address a wide range of data intensive processing applications. We notice comparable results between the SIMD execution and the DSP [9] execution. This is explained by the fact that the DSP is realized on an ASIC (Application Specific Integrated Circuit) and presents a higher frequency than the FPGA based mppSoC. Nevertheless, the DSP is slightly faster than our proposed SoC.

This analysis demonstrates the efficiency of mppSoC to compute data intensive processing applications. Moreover, the

TABLE I
SYNTHESIS RESULTS ON STRATIX 2S180

Number PEs	Processor Assembling Methodo.	Neighbourhood network	mpNoC interconnection network	Logic Utilization ALUTs	registers	%	Total memory ACU (bytes)	PE (bytes)	%
8	replication	–	crossbar	42059	16863	27	4096	1024	3
16	reduction	–	omega	27268	11481	21	4096	4096	6
32	reduction	2D Torus	crossbar	81796	22478	45	2048	600	4
64	reduction	Xnet	–	89154	24479	49	4096	2048	8
84	reduction	–	omega	130603	41620	96	2048	2048	10

TABLE II
SYNTHESIS RESULTS ON STRATIX 2S180

Application	mppSoC configuration	Execution Time (μs)	System : Execution Time (μs)
Parallel matrices of size 200x200 multiplication	processor (minimips/reduction) + 64 PEs + 2D Torus	6000	H-SIMD [8] : 7000
64-tap FIR on an impulse response with a length of 128	processor (minimips/reduction) + 64 PEs + linear neighbouring network	2.06	MD32 DSP without BPU [9] : 1.26
RGB to YIQ color conversion on 320x240 color image	processor (NIOS/replication) + 32PEs + crossbar based mpNoC	769	multi-FPGA [10] : 10000

mppSoC is flexible, programmable and can be easily tuned according to the application requirements.

IV. CONCLUSION

This work presents a rapid IP assembling approach to design a flexible massively parallel processing on chip system named mppSoC. This approach leads to the design large and complex parallel architectures with less costs and higher performances. An mppSoC generation tool is also developed to automatically generate the mppSoC configuration at an RTL abstraction level as specified by the designer at a high abstraction level.

This work opens an interesting topic for future research and development on parallel architectures. Our long term is to promote a more complete framework to make mppSoC exploration.

REFERENCES

[1] E. Strohmaier, J. J. Dongarra, H. W. Meuer and H. D. Simon, *The Marketplace of High Performance Computing*. Parallel Computing, Vol. 25, Elsevier, 1999, pp. 1517-1544.

[2] A. Gentile and D. Scott Wills, *Portable Video Supercomputing*. IEEE Transactions on Computers, Vol. 53, No. 8, August 2004.

[3] W. C. Meilander, J. W. Baker and M. Jin, *Importance of SIMD Computation Reconsidered*, in: Proceedings of International Parallel and Distributed Processing Symposium, 2003.

[4] Ph. Marquet, S. Duquennoy, S. Le Beux, S. Meftali and JL. Dekeyser, *Massively parallel processing on a chip*, in: Proceedings of the 4^{th} International Conf. Computing Frontiers (FPL), 2007.

[5] M. Baklouti, Ph. Marquet, JL. Dekeyser and M. Abid, *A design and an implementation of a parallel based SIMD architecture for SoC on FPGA*, in: Proceedings of Conference on Design and Architectures for Signal and Image processing (DASIP), 2008.

[6] M. Baklouti, Ph. Marquet, JL. Dekeyser and M. Abid, *Reconfigurable Communication Networks in a Parametric SIMD Parallel System on Chip*, in: International Symposium on Applied Reconfigurable Computing (ARC), 2010.

[7] H. Fatemi, B. Mesman, H. Corporaal, T. Basten and R. Kleihorst, *RC-SIMD: Reconfigurable Communication SIMD Architecture for Image Processing Applications*, Journal of Embedded Computing, Vol. 2, 2006, pp. 167-179.

[8] X. Xu, S. G. Ziavras, T. G. Chang, *An FPGA-Based Parallel Accelerator for Matrix Multiplications in the Newton-Raphson Method*, in: Lecture Notes in Computer Science, Embedded and Ubiquitous Computing, Springer, Berlin, 2005, pp. 458-468.

[9] C. Xiaoyi, Y. Qingdong and L. Peng, *Data Bypassing Architecture and Circuit Design for 32-bit Digital Signal Processor*, in: Journal of Electronics, Vol. 22, No.6, 2005.

[10] M. Z. Hasan and S. G. Sotirios, *Customized kernel execution on reconfigurable hardware for embedded applications*, in: J. Microprocessors and Microsystems, 2009, pp. 211220.

[11] M. Sayed, W. Badawy and G. Jullien, *Towards an H.264/AVC HW/SW Integrated Solution: An Efficient VBSME Architecture*, IEEE Transactions on Circuits and SystemsII, Vol. 55, No. 9, September 2008.

[12] R. Lopez Rosas, A. de Luca and F. Barbosa Santillan, *SIMD Architecture for Image Segmentation using Sobel Operators Implemented in FPGA Technology*, 2nd International Conference on Electrical and Electronics Engineering (ICEEE) and XI Conference on Electrical Engineering (CIE 2005) , Mexico City, Mexico, 2005.

[13] Altera Corporation, *Stratix II Device Handbook*, 2004.

[14] Opencores, *minimips overview*, http://www.opencores.org/?do=project &who=minimips.

[15] S. Connors, T. Brown, J. Hansen and C. Kief, *Mips source code*, http://www.eece.unm.edu/ivpcl/classes/mips microprocessor/mips microprocessor.html.

[16] Altera, *Nios II Processor Reference Handbook*. http://www.altera.com/literature/hb/nios2/n2cpu nii5v1.pdf., 2009.