NoCRay, an FPGA Network-on-Chip Based MP-SoC for Graphics Ray Tracing Applications

Sergio Tota, Mario R. Casu, Paolo Motto, Massimo Ruo Roch, Maurizio Zamboni

sergio.tota@polito.it

VLSI Laboratory – Department of Electronics - Politecnico di Torino

http://www.vlsilab.polito.it

Abstract

A NoC-based MP-SoC has been developed and tested on a FPGA prototype board. The case-study, a graphics ray tracing rendering engine, demonstrates the integration of a scalable communication infrastructure, a configurable microprocessor design, a programming model and an automated design flow.

1. Introduction

Network-on-Chip (NoC) is seen as the interconnection methodology for future homogeneous and heterogeneous Multiprocessor Systems-on-Chip (MP-SoC) [1]. The motivations for this choice are the better scalability of performance with the increasing number of processing elements (PE) compared to standard on-chip busses, the high regularity which improves layout and particularly the allocation of wiring resources and the layered design approach which enables tackling the SoC complexity.
MP-SoC design is a multi-disciplinary research activity that encompasses on-chip communication infrastructures, microprocessor architectures, programming models, co-design/co-simulation flows and EDA tools for automated system generation.
So far the need for higher performance had led to an increase of the architectures complexity in a monolithic way; of which the microprocessors evolution is the most evident example. From the various x86 architectures to the more recent superscalar ones the trend was to implement always more complex solutions, driven by a need for an ever increasing system frequency and instructions per cycle. Recently, this trend has started to slow-down even if the number of transistors is expected to continue to double every two years. Power-thermal issues as well as design complexity have begun to limit the performance growth-rate compared with the increasing number of transistors available in a single die [3].
One way to cope with this “productivity gap” is the “tile-design” concept which underlies a simple yet effective paradigm: parallelization through replication of many identical blocks placed each in a tile of a regular array fabric. Instead of focusing on improving the complexity of a single block, the solution aims at delivering performance through several replicas of the same basic blocks. The final aggregate-performance will only be limited by the number of transistors available in the same silicon. This approach has the terrific consequence of making systems design a matter of instantiation capability instead of architecture complexity, an objective which has to be pursued through innovative scalable hardware/software solutions.
Our research activity in this field focused on the various aspects that this new paradigm coalesces. The integration of a scalable communication infrastructure, a configurable microprocessor design, a programming model and an automated design flow is demonstrated through a MP-SoC case study: a graphics ray tracer based rendering engine that lively runs on a FPGA prototype board.

2. Research Framework

The following summarizes the many facets of this work.

2.1 Network-on-Chip

Providing a scalable communication infrastructure is the basis of tile-design. The key component of NoC is the *switch*. This IP has been designed with the goal of keeping the area as small as possible and thus making it possible to achieve high clock frequency. Another important aspect is the high configurability of such IP depending on application requirements. To fully use the whole capacity of such infrastructure, a basic packet-switching algorithm based on *deflection-routing* has been used.

2.2 Microprocessor

Current microprocessors follow a classic approach where a bus interface is used to communicate with peripherals. High performance channels are indeed required to fully use the capacity of the NoC. For this purpose custom ad-hoc ports are required as well as an instruction-set support. The use of configurable processors plays a fundamental role in this scenario. State-of-the-art Xtensa Tensilica configurable processors have been used for this purpose providing a full instruction-set support for NoC I/O.

2.3 Programming model

The classic shared-memory approach poses several limits on the scalability of such systems in terms of software deployment. In order to fully exploit concurrency, a distributed programming model is required to help programmers think *distributed*. The consolidated Message Passing Interface (MPI) methodology has been used. A subset of the basic primitives used for on-chip programming, which we called embedded MPI (eMPI), has been developed.

2.4 System Co-Design/Co-Simulation

The possibility of having highly accurate as well as fast models of all the system components is a fundamental aspect of the design flow. Cycle-accurate Instruction Set Simulators (ISSs) combined with powerful tools for system analysis speed up design time of orders of magnitude. For this task the Mentor Graphics *Seamless* co-design tool has been used, thus allowing us to develop the software layer in parallel with the hardware implementation.

2.5 Design Automation

The regularity offered by the tile-based design approach enables powerful automated design methodologies to quickly generate the system architecture. An EDA tool has been developed, *RapidBUILDER*, based on the *Eclipse* framework. Given the NoC size, packet size, network topology and routing algorithm, RapidBuilder quickly generates the whole system composed by the processors and the NoC, as well as co-simulation scripts and models.

3. NoCRay, an eMPI based Raytracer

As a case-study of the proposed design flow an MP-SoC (Fig. 1) with a parallel graphic application, the SPLASH-2 Raytracer, has been mapped on a FPGA demonstrator. The original version was intended for shared-memory architectures, thus a MPI version has been used [4] as a starting point and ported to the MP-SoC environment using the eMPI subset. The Instruction and Data memory footprints have been reduced to 59KB and 16KB respectively for each processor. In order to save FPGA resources, each processor couple shares the same Instruction Code using a dual-port memory (Fig.2**).** A monitor processor dispatches the workload among the others ensuring a load balance: When one ends a local computation, the monitor assigns a new job to it. The monitor also sends computed pixels to a remote host via an Ethernet controller. Since each RGB pixel use a 24 bit representation, each NoC packet is 32 bits wide inclusive of 8 bits reserved for addressing, validation and other services.

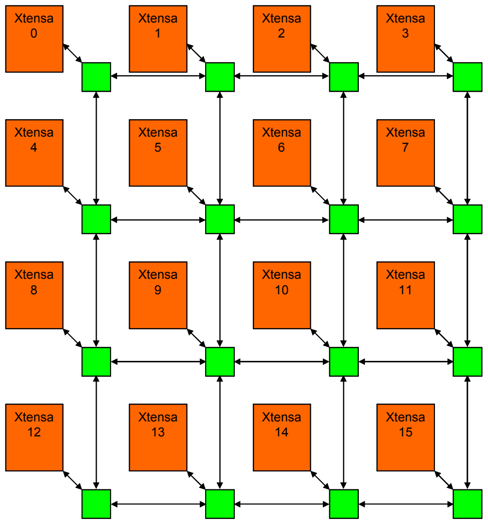

Figure 1: MP-SoC block scheme.

A 16 processor system mapped on a Xilinx Virtex-4LX has a clock frequency of 85MHz and an equivalent gate count of 20 Million.

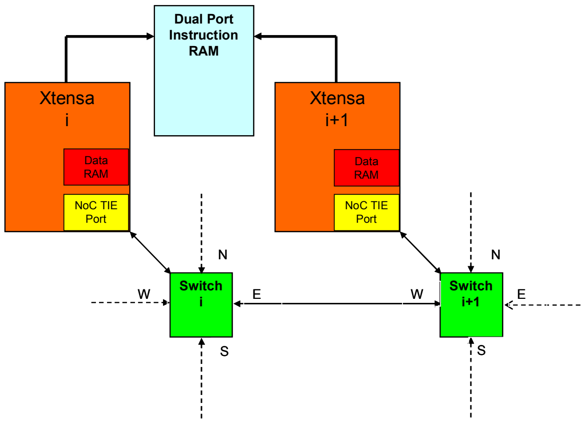

Figure 2: Detail of processors and memory.

The whole system has been generated with RapiBUILDER, Tensilica Xplorer and simulated with Mentor Graphics Seamless prior to FPGA download.

4. Conclusion

The main goal of this research activity was to propose a set of highly integrated hardware and software components for MP-SoC design. This new approach has been verified with a real-life application, a multiprocessor graphic ray tracer engine, NoCRay. What clearly emerges is that with a tiled approach it is possible to design complex architecture with a drastic reduction of time and resources with respect to a classic approach. The next step of this research activity will be to provide a full *application-driven* MP-SoC design methodology as well as Operating System support for MPI applications.

5. Acknowledgement

We would like to thank Tensilica, Mentor Graphics and Xilinx for their precious support.

5. References

[1] Jerraya A. and Wolf W. (editors), Multiprocessor Systems-on-Chip, Elsevier Morgan Kaufmann, San Francisco, California, 2005.
[3] Held J., Bautista J., Koehl S., “From a Few Cores to Many: A Tera-scale Computing Research Overview”, Intel Development Forum, September 2006.
[4] Fatahalian K, Hui J., “A Parallel Stochastic Photon Mapping Ray Tracer using MPI” [Online]. Available: http://www.eecs.berkeley.edu/~jwhui/research/photon_mapping/