Crossbus: a Design Flow and a NoC for MPSoPC

Gabriel Oshiro Zardo, Dominique Houzet, Sylvain Huet

Crossbus: a Design Flow and a NoC for MPSoPC

Gabriel Oshiro Zardo
gabrieloshiro@gmail.com
UFRGS
Brazil

Dominique Houzet
dominique.houzet@gipsa-lab.inpg.fr
INPG GIPSA-Lab CNRS
France

Sylvain Huet
sylvain.huet@gipsa-lab.inpg.fr
INPG GIPSA-Lab CNRS
France

Abstract

In this paper we present Crossbus: a NoC and a design flow developed at the GIPSA-lab which targets the implementation of dataflow applications, e.g. signal, image, video processing, on Xilinx FPGAs.

1. Introduction

The number of Processing Elements (PE) on MPSoC is increasing rapidly: the 2007 ITRS roadmap predicts that the number of PE in consumer portable design will evolve from 60 in 2009 to 1400 in 2022. The Network on Chip (NoC) appeared to be solution to achieve an efficient on-chip communication for such complex designs. It offers a better performance/scalability/reliability compromise than classical communication topologies such as shared buses, point to point links… Many NoCs have been developed since the 2000 either in academia or industry [1]. Many of them target ASICs. Nevertheless, the ever increasing density of integration makes the NoC a relevant communication design paradigm even for FPGAs [2][3].
Crossbus is a NoC and a design flow developed at the GIPSA-lab which targets the implementation of dataflow applications, e.g. signal, image, video processing, on Xilinx FPGAs. The multi FPGAs feature of Crossbus, which is totally transparent from the application designer point of view, is one of the novelties of this work. Besides, the NoC has been developed in conjunction with a dataflow programming model expressed with SystemC. This allows optimizing the NoC by implementing some primitives of the programming model in hardware.

2. Hardware Architecture

The Crossbus NoC allows interconnecting N FPGAs as a bidirectional full duplex ring network. Rocket IOs, high speed serial links from Xilinx, are used for these inter FPGAs links. Each FPGA embeds a 0xP 2 dimensional matrix of routers. Some places in the matrix can be left empty. The routers are linked has a 2D torus. The links are monodirectional either in the vertical or horizontal directions, except the first row which is bidirectional full duplex. The interconnections between routers, inside FPGAs, are handled by FSLs (Xilinx FIFOs). An example of the Crossbus topology is given in Figure 1. Each grey square inside a FPGA symbolizes a router, the FPGAs matrix size is 0=P=3.

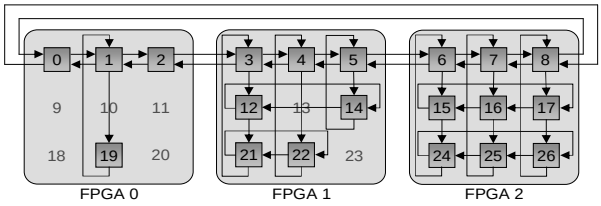

Figure 1: Crossbus Architecture Example

Each router can be connected with up to 4 hardware or software PEs. Crossbus uses an XY routing policy.

3. Programming model and design flow

Crossbus design flow is presented in figure 2. It relies on the SystemC programming model and allows generating both the software and hardware parts of the system.
The application is specified with a subset of SystemC which allows expressing the parallelism of the application. It is described as a set of `sc_modules` interconnected by `sc_signal` and `sc_fifo` communication channels. The synchronisation is expressed with the help of calls to the `wait` primitive on a clock signal. The application is validated and profiled through simulations with ModelSim.

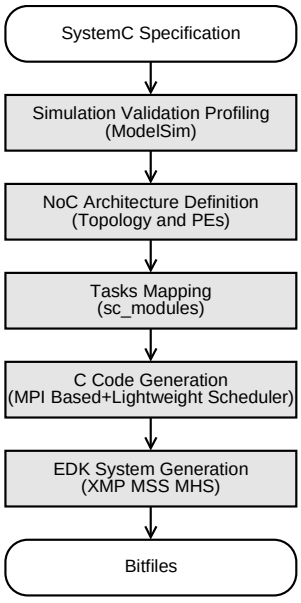

Figure 2: Design Flow

Thanks to its expertise and the profiling results, the designer defines (1) the hardware architecture, i.e. the NoC topology and the number of PEs and their kind (hardware or software) attached to each router (2) the mapping of the sc_modules on the PEs.

The software code embedded in the software PEs is generated with our SysCellC tool [5]. With the view to obtain a fast code with a small footprint, it transforms the SystemC specification in a C code which respects the SystemC semantics. This tool generates (1) a lightweight static application specific scheduler and (2) transforms the SystemC primitives' calls with MPI-2 RAM [4] calls. For example, `MPI_Init` is used to initialize the application, `MPI_Comm_Rank` is called during the initialization by the software PEs to obtain their identifier, the synchronization is expressed with `MPI_Barrier`, `MPI_Put` is used to express the data transfers and `MPI_Finalize` is called at the end of the program. This MPI based C code is generic and can be compiled on any kind of software PE as long as we dispose of the implementation of the mentioned above primitives, e.g. we use this C code to simulate applications on a Cell processor [5]. We developed an implementation of these primitives for the MicroBlaze processor from Xilinx, the processor we target with Crossbus. We hardwired the implementation of `MPI_Barrier` and `MPI_Comm_Rank` primitives. `MPI_Put` transfers information between PEs through DMA.

At last we generate the hardware description of the system in the MHS file format. This file format is used by the EDK Tool from Xilinx to describe the hardware architecture of a system [6]. All the hardware components composing the system (the NoC infrastructure, the hardware and software PEs connected to each router) are instantiated in this file. The hardware PEs can be either hand-written or instantiated from a library or generated with a High Level Synthesis (HLS) tool, e.g. we experience in the context of this work the generation of hardware PEs with ImpulseC [7]. We generated the software description of the System in the XMP and MSS files format. These files format is used by the EDK Tool from Xilinx to describe the software projects embedded in each software processors composing the system and to specify the drivers associated the hardware components [6]. Bitfiles are generated by EDK from these MHS and MSS and XMP files.

4. Results

The Crossbus NoC and design flow have been validated on two case studies. A producer/consumer case and a CDMA software radio case. Table 1 shows the hardware cost of a router in function of the number of connected PEs. It points out that the size of a router connected to 4 PE is comparable with the size of a MicroBlaze. Table 2 shows the execution time in clock cycles in function of the number of words in a message of several primitives for the producer/consumer case study. The system initialisation and the `write_signal` are proportional to the size of the message whereas `MPI_init` and the scheduling only depend on the number of communication channels. Since `MPI_Comm_Rank` and `MPI_barrier` are hardwired they only take 2 clock cycles.

Num. of PE	Ios	Funct. gen.	CLB Slices	DFFs or Latches	Freq. (Mhz)
1	497	2987	1494	628	173.3
2	582	3277	1639	609	178.6
4	743	3987	1994	668	203.3

Table1: Router costs

Primitive	4 words	8 words	16 words
System initialization	1385	2281	4057
MPI_init()	426	426	426
MPI_Comm_Rank()	2	2	2
Scheduling(wait())	60	60	60
write_signal	86	133	175
MPI_Barrier()	2	2	2

Table2: Execution Time in clock cycles in function of the number of words in a message

5. Conclusion

Our results show that it is possible to efficiently implement an application described at a high level of abstraction on a multi FPGAs platform with a multiprocessor plus NoC design paradigm. Although we initially target FPGAs, we plan to extend this work to ASICs solutions: we believe that a small footprint NoC is also pertinent for ASICs.

6. References

[1] Micheli, G.D. & Benini, L. Wolf, W. (ed.) Networks on Chips M. Kaufmann, 2006

[2] T. Marescaux, J-Y. Mignolet, A. Bartic, W. Moffat, D. Verkest, S. Vernalde, and R. Lauwereins. Networks on Chip as Hardware Components of an OS for Reconfigurable Systems. In Field-Programmable Logic and Applications, volume 2778/2003 of Lecture Notes in Computer Science, pages 595-605. Springer Berlin / Heidelberg, 2003.

[3] Riso, S.; Sassatelli, G.; Torres, L.; Robert, M. & Moraes, F. Réseau d'Interconnexion pour les Systèmes sur Puce : le Réseau HERMES SCS'04, 2004

[4] Gropp, W.; Lusk, E. & Skjellum, A. Using MPI (2nd ed.): portable parallel programming with the message-passing interface MIT Press, 1999.

[5] Kaouane, L.; Houzet, D.; Huet, S., "SysCellC: SystemC on Cell," Computational Sciences and Its Applications, 2008. ICCSA '08. International Conference on , vol., no., pp.234-244, June 30 2008-July 3 2008

[6] Xilinx, EDK Concepts, Tools, and Techniques A Hands-On Guide to Effective Embedded System Design, http://www.xilinx.com/ise/embedded/edk_docs.htm

[7] Impulse Accelerated Technologies, ImpulseC, http://www.impulsec.com/