Modular Asynchronous Network-on-Chip: Application to GALS Systems Rapid Prototyping

Jérôme Quartana[1], Laurent Fesquet[2], Marc Renaudin[2]
[1]CMP-GC Centre Microélectronique de Provence Georges Charpak,
Avenue des Anémones, 13120 Gardanne, France quartana@emse.fr
[2]TIMA Laboratory, 46 av. Felix Viallet 38031 Grenoble Cedex, France
{laurent.fesquet,marc.renaudin}@imag.fr

Abstract. This paper presents an innovating methodology for fast and easy design of Asynchronous Network-on-Chips (ANoCs) dedicated to GALS systems. A topology-independent building-block approach permits to design modular and scalable ANoCs with low-power and low-complexity requirements. A crossbar generator is added to the existing design flow for fast system architecture exploration. A multi-clock FPGA allows a fast prototyping of complex ANoC-centric GALS systems. A demonstrative platform is implemented onto an Altera Stratix FPGA. It includes synchronous standard IP cores and asynchronous modules connected through an asynchronous 6x6 crossbar. Results about communication costs across the Asynchronous NoC and synchronous/asynchronous interfaces are reported.

1 Introduction

GALS paradigm is to partition a system design in decoupled clock-independent modules [1]. Design parameters of each block can be adjusted independently (performance, power consumption or clock-tree management to name but a few). Another benefit of GALS paradigm is to separate the design of communication from functionality by using handshake protocol synchronization (amongst other techniques).

Asynchronous NoCs (ANoCs) strongly benefit to such a globally asynchronous design methodology. Clockless interconnect networks improve reliability by removing clock-domain crossing synchronizations and by using delay-insensitive arbiters for solving routing conflicts [2, 3]. Global design constraints are released. They also offer robust communications thanks to an automatic data transfer regulation (elastic pipeline): no data item can be lost or duplicated. Moreover, regular distributed network topologies (any topology based on point-to-point links,

such as meshes, tores or crossbars), built of independent routing nodes, fully exploit modularity and locality design properties of asynchronous circuits. To illustrate these benefits of using ANoCs for GALS systems, several publications bringing major research contributions can be cited.

In [4] a stoppable clock methodology, based on asynchronous wrappers around synchronous blocks, is used to compare topology performances by using ad-hoc synchronous peripherals adapted to the asynchronous networks. Such techniques need training sessions and suffer from PVT sensitivity [21] and from penalties in restarting the clocks.

Beigne et al. present in [3] an asynchronous mesh topology providing a high Quality-of-Service (QoS), using a multi-level design flow. This very efficient ad-hoc architecture is dedicated to a specific application and has a high complexity cost. Bolotin and al. use in [5] a generic architecture to evaluate four classes of packet services. After a training session on every class, the most appropriate service is implemented onto a point-to-point link between two components, according to the communication requirements. This NoC architecture is more modular than [3] but for a higher complexity cost.

In [6] and [7], Bainbridge and Lovett develop a modular and low-complexity ANoC design methodology, using simple one-to-two and two-to-one switches to build regular topology networks. In such structures, arbiters are very simple and so efficient for packet routers with few channels to drive. However, assembling these switches will heavily increase latency and area costs for large multi-inputs/outputs routers.

Compared to these works, our purpose is to provide a simple and flexible generic structure which allows fast design of a large spectrum of ANoC topologies for GALS systems requiring efficient communications at a low complexity cost. According to this motivation, this paper presents in section 2 a topology-independent structure which is strongly modular, scalable and robust and which permits by using accurate-function building blocks to design ANoCs for high-reliability, low-power and low-complexity requirements. Section 3 gives some details of the self-timed FIFO structure to interface synchronous and asynchronous domains. Section 4 details the design flow methodology. A crossbar generator has been developed for fast system architecture exploration. As such a flexible ANoC structure is well-suited for rapid GALS system prototyping [9], we remind a special methodology [8] to synthesize asynchronous modules onto FPGA, with an extension for non-deterministic arbiter circuits [9]. In section 5, this methodology is applied to implement an ANoC-centric GALS system onto a multi-clock Altera Stratix FPGA. It includes synchronous standard IP cores and asynchronous modules connected through an asynchronous 6x6 crossbar. Results about performances of the Asynchronous NoC are reported as well as a peripheral-to-peripheral communication cost (across both the ANoC and the synchronous/asynchronous interfaces).

2 Asynchronous NoC design

Our methodology fully exploits the modularity of asynchronous circuits. We provide a basic layered structure of ANoC with no predefined topology, by using a building-block approach. Each block or layer has been accurately defined to efficiently deliver one of the major functions of an interconnect network (these functions are detailed in section 2.2 with the description of each block):

- service-level communication protocols,
- synchronization interfaces at mixed-timing domains,
- signal-level information transport,
- packet arbitration and routing in interconnect nodes.

Moreover, the basic blocks have been designed with an objective of reliability improvement (section 2.1) and with respect to low-complexity, "easy-plug" and scalability features. The result is a simple and flexible structure having efficient latency and throughput and a wide variety of high-level services at low-complexity and low-power costs. Such structure allows fast design of any ANoC regular distributed topology.

2.1 Focus on synchronization bolts

Our methodology for designing ANoCs is focused in part on solving synchronization problems. The two major synchronization bolts for a GALS system are: synchronization at clock domain boundaries and arbitration between concurrent requests [14]. Such circuits have a non-deterministic behavior. We put special invest to improve reliability/performance tradeoffs of these synchronizer circuits.

Clocked synchronization. As discussed in the introduction, using an ANoC is in itself a reliability improvement by removing clock-domain crossing synchronizations through the interconnect network. However clocked synchronizers are still required between Synchronous peripheral Blocks (SB) and the ANoC. Discussion on this synchronous/asynchronous interface is developed in section 2.2 and structural details are given in section 3.

Delay-insensitive arbiters. Arbitration circuits, or simply arbiters, are required where a restricted number of resources are allocated to different user or client processes. Packet routers are such cases. In the case of an ANoC, delay-insensitive arbiters have this main advantage of being hundred-percent reliable (enough time is given to resolve metastability). Reliability of on-chip communication systems is becoming a major issue since the increase transaction rates are drastically reducing the so-called Mean Time Between Failure characterizing clocked synchronizers. In [2] we present a class of delay-insensitive arbiters which decouple the sampling of incoming requests from the arbitration process in a strong modular and reliable structure. Such arbiters use a Parallel-Request-Sampling structure and are used in [3, 9] and in the following ANoC structure.

2.2 Modular ANoC structure

We cut out the construction of ANoCs in five basic components or layers, as illustrated in Fig. 1.

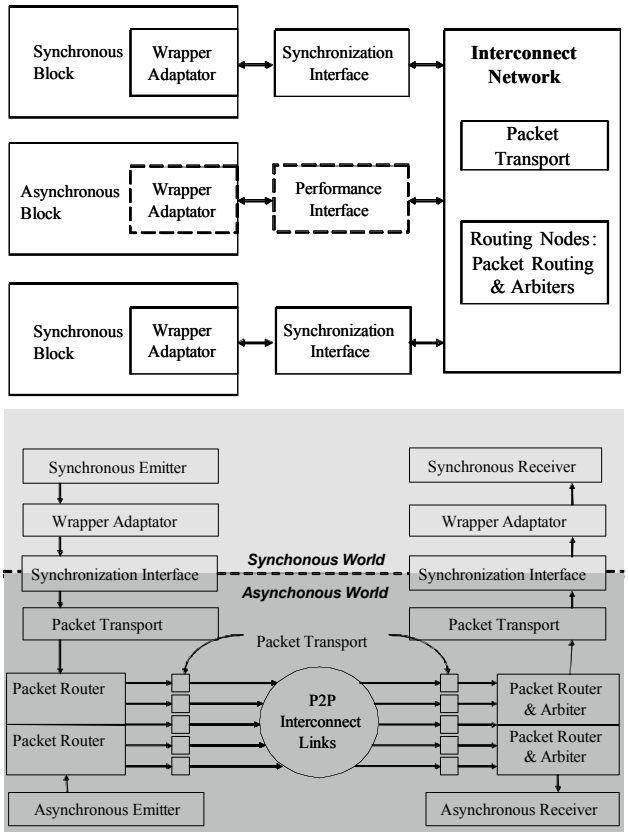

Fig. 1. ANoC-centric GALS architecture: a) abstract structure b) layered structure

1. Wrapper Adaptor (WA). This resource is required to translate between the communication protocols used by a synchronous or asynchronous peripheral and the interconnect network. The WA component adapts both flit and packet levels of the communication protocols. The details of these protocols are beyond the scope of this paper [3].

2. Synchronization & Performance Interface (SPI). This component binds the SB clock domain with the ANoC using a FIFO decoupling method. The SPI consists of a standard double flip-flop (DFF) synchronizer and of an asynchronous FIFO. Such simple synchronization interface facilitates plugging of standard synchronous IP cores.

The DFF resynchronizes asynchronous signals with the SB clock. The DFF offers actually a very sufficient reliability/latency tradeoff (two clock cycles per input signal sampling) [15], compared to numerous clocked synchronizer's improvements [16].

The asynchronous FIFO transforms the synchronous protocol in the corresponding asynchronous protocol, adapting relative speeds between the SB and the ANoC. For AB, such a FIFO is optional and can be used for pipeline performance optimization. In this case we call it Performance Interface (PI) (Fig. 1). Details of the asynchronous FIFO structure are presented in section 3. This architecture is based on an existing asynchronous FIFO [17]. The level of parallelism between data and control flows is improved and two versions are delivered: a low-latency version or a low-power consumption version, according to design requirements.

3. Packet Transport (PT). This resource adapts the physical level (or signal-level) of the communication protocol. The PT component provides successive protocol conversions from SPI component to delay-insensitive NoC core for best power consumption and robustness. Between SPI and PR layers, bundle data protocols are converted in delay-insensitive protocols for better robustness. Between the packet routers (PR layer), the four-phase protocols can be converted in 2-phase protocols for long interconnect links for lower power consumption and higher speed [18].

4. Parallel-Request-Sampling Priority-Arbiter (PRS-PA). This resource provides a self-timed arbiter with a decoupled arbitration process and a 100% reliable request sampling structure based on delay-insensitive parallel synchronizers [2, 19] (section 2.1).

5. Packet Routing (PR). This resource offers a modular routing of data items for transaction services (packet level services such as burst mode or split transactions). PRS-PA and PR resources are parts of ANoC routing nodes, as detailed in section 2.3.

2.3 Switches architecture for ANoC routing nodes

Packet router is the core component of an interconnect network. The packet routers are assembled with modular elementary blocks, as shown in Fig. 2, with the same objectives of low-complexity, easy "plug-and-play" and scalability as for the complete ANoC.

Emitter module. Fig. 2 illustrates two switch instances. The n-to-1 switch, or Emitter, is built around the PR (Packet Router) and PRS-PA (Priority Arbiter) components, as previously presented in section 2.2. The PR resource is decomposed in three modules: Packet Analyzer (PA), Data Path Controller (DPC) and MUX module. The Emitter component delivers two major classes of packet level services: arbitration service and transaction service. The PA block decodes $Channel_i_ctrl$ message in order to extract arbitration and transaction information parts and to drive it respectively to the PRS-PA and DPC modules. Arbitration information is composed of *Request* and *Priority_level* (optional) channels, used by the PRS-PA module to arbiter incoming requests. Once a $Channel_i_data$ is elected, PRS-PA

informs the datapath controller module (DPC) through *Selected_Channel*. DPC exploits it and the *Transfert_mode* channel to control data flow on the elected $Channel_i_data$ and to drive the switch output (MUX module). Through *Transfert_mode* channel, transaction information delivers packet status, such as single flit packet or for burst mode: start-packet flit, body flit, end-of-packet flit. Once the packet transfer is achieved, DPC module informs the sleeping arbiter module PRS-PA through *Sampling* channel that a new transaction can start.

Receiver module. The 1-to-m Switch, or Receiver, is a PR component which realizes the dual operation by driving the input (*Packet_Ctrl* and *Packet_Data* channels) to the selected *Target_Address*. No arbitration is needed here. By composing these switches we can build in short design time fast and efficient routing nodes (sections 4.2 & 5.1).

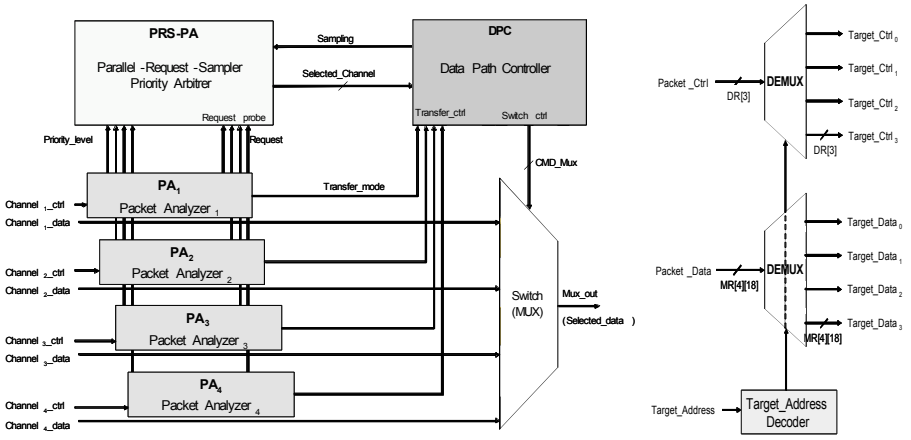

Fig. 2. Switch components: a) Emitter (n-to-1 switch) b) Receiver (1-to-m switch)

3 Asynchronous FIFO for mixed domain interfaces

3.1 Reference work

Chelcea and Nowick present in [17] several mixed-timing FIFO designs. The designs are implemented as a core of micropipeline-style circular arrays of identical cells connected to common data buses. Data items are not moved around the array once they are enqueued, preserving power consumption. Control is made with two tokens: the first one allows enqueuing data whereas the second one allows dequeuing data. This asynchronous array is scalable and modular and offers very low latency.

The core of these asynchronous FIFO cells are used to design instances of double-clock FIFOs and in our concern mixed synchronous/asynchronous FIFOs.

But we decide not to use this mixed version of the FIFO. Indeed, our GALS architectures integrate heterogeneous Synchronous peripheral Blocks (SBs) communicating across the ANoC. These synchronous and asynchronous domains will present very different working speeds. In such a situation, the mixed-timing

version of the FIFO (interfacing a SB with an ANoC routing node) can not guarantee one write or read operation per cycle on its synchronous part (SB side). The FIFO will often be empty or full and speed performances will be degraded by a global three clock cycles latency cost, due to complex FIFO-state detector interfaces. Preliminary result analysis on the FPGA platform confirms large different speeds between SBs and high-speed ANoC (section **Error! Reference source not found.**).

To avoid the use of such latency-penalizing interfaces, an improved version (section 3.2) of the fully asynchronous FIFO is provided to interface synchronous and asynchronous working domains. The FIFO is connected to a standard DFF synchronizer which reduces the latency to two cycles. This solution is robust (section 2.1) and efficient to adapt domains with large difference in working speeds. The next section briefly describes how we improved the self-timed FIFO architecture.

3.2 Improved asynchronous FIFO

The architecture of the fully asynchronous FIFO is transformed in two ways to improve its performances.

1. Improved level of parallelism. This architecture has a limited degree of parallelism between control and data paths (token passing and data enqueuing/dequeuing operations). We use the TAST tool suite (see section 4) features to improve it, and consequently to improve the speed of the FIFO. A FSM modeling of the FIFO in CHP language allows a decoupling of token passing and data enqueuing/dequeuing operations. TAST synthesizer options allow to parameter the synchronization point between these operations and therefore ensure the correctness of the FIFO. Both delay-insensitive and micropipeline versions of a FIFO can be synthesized.

2. Low-power and fast architecture exploration. The common data buses give increasing power consumption penalties for deep FIFOs. Moreover, the bus buffers have to be re-designed for each new FIFO size. We replace these high-loaded buses with two components called *One-to-Two Sequential* switch (*OTS*) and *Two-to-One Sequential* switch (*TOS*). These components are bonded in a vertical binary tree of switches as shown in Fig. 3.

Fig. 3 shows the horizontal array of FIFO cells (*FC*) with the distributed right-to-left token passing control path [17]. Data items move vertically across a path of *OTS*, *FC* and *TOS* components. Each *OTS* component is a 1-to-2 demultiplexer with automatic toggle. Each data item is alternatively driven to one of both output paths, starting on the right path. The *TOS* components are the reciprocal 2-to-1 multiplexers, receiving the first data item on the right input path and then automatically switching from one input to the other. A version with one-to-three and three-to-one switches can be provided to extend the available size of the FIFO.

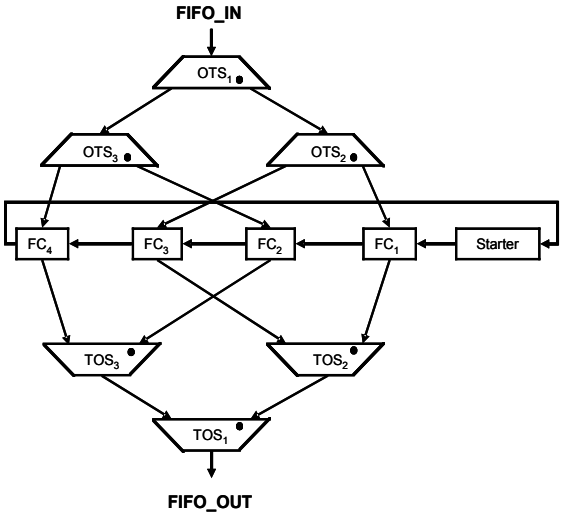

Fig. 3. FIFO structure with mux/demux trees

This architecture ensures the correctness deadlock-free operations of the FIFO. *OTS* and *TOS* components work as supplementary data memorization cells. Moreover, the cell structure for data paths is identical for *FC*, *OTS* and *TOS* components, i.e. a data latch added to a Muller gate which controls channel request signal. Consequently, the input and output loads of each cell are balanced. Compared to the common bus solution, the mux/demux binary trees solution provide the following features: design of the FIFO is simplified, scalability and power consumption are improved and latency is degraded (but throughput is identical).

3. Conclusion. A high-throughput self-timed (either QDI or µP style) FIFO with a high degree of parallelism is delivered to robustly interface SB and ANoC modules in an ANoC-centric GALS system. Two versions are available: a mux/demux binary-tree version for fast system architecture exploration (especially for optimal FIFO size) and low-power; and a common-bus version for low-latency requirements.

4 Design flow

4.1 Design methodology

We specify and model asynchronous circuits in CHP (for Communicating Hardware Processes), a high-level description language based on communicating processes [10, 11]. The processes are synthesized using TAST, a suite tool [12] dedicated to asynchronous circuit synthesis. The TAST tool enables to map the CHP specification onto a standard-cell library and/or a specific cell library [13] when targeting ASICs, or to map onto FPGA for rapid system prototyping [8, 9].

4.2 Automatic crossbar generation

We use an automatic crossbar topology generation tool to implement the 6x6 crossbar ANoC. The tool controls adjustable design parameters for some of the five ANoC modular blocks/layers. It supports fully-interconnect or Octagon [20] topology generation and modular routing node cores generation, which can be hand-adapted and assembled in more complex regular interconnect topologies, such as meshes. The choice of crossbar or fully-interconnect topologies ensures a fast, flexible and low-complexity system architecture exploration. It allows implementing efficient Emitter and Receiver components in terms of routing complexity, latency and throughput and in terms of control cost. The Receiver component supports high packet service extensions thanks to its high modularity.

So far, the adjustable parameters are:

1. *Crossbar size*. It depends of the number of the system's components.
2. *Point-to-point (p2p) interconnects width*. The width of each interconnect path is defined according to the required bandwidth of each p2p linked SB or AB.
3. *Priority algorithm*. The priority solving function can be programmed. Available policies are round-robin, FIFG and non-interruptible two-level priority policies. The FIFG policy can be programmed independently for each routing node.
4. *Transaction services*. DPC module can be programmed to support data transaction services. For the time being, only the burst mode is available. All routing nodes must support the same transaction services.

4.3 Synthesis of QDI circuits onto FPGAs

This section presents an ANoC-centric GALS architecture implemented onto a multiclock Stratix Altera FPGA. We give in [8] a generic synthesis methodology to properly place and route asynchronous elements or mixed synchronous/asynchronous circuits onto a FPGA, respecting the specific timing assumptions of either QDI or micropipeline (μP) asynchronous design techniques. This methodology is extended in [9] to synthesize arbiter circuits with non-deterministic behavior, due to their synchronizer elements. A special circuit mapping is presented for delay-insensitive synchronizers devoted to asynchronous arbiters.

This FPGA-prototyping methodology is applied to the clock-less modules of the following architecture (ANoC and DES). The ANoC is designed according to the modular building method of sections 2 and 3.

5 Validation platform

5.1 PACMAN platform

We demonstrate our network-centric GALS building methodology with a case-study implemented on a Stratix Altera FPGA. This system is a first prototype version of a generic GALS platform called PACMAN, for Programmable And Configurable Multiprocessor Asynchronous Network.

The PACMAN first-version architecture is shown in Fig. 4. It includes an ANoC interconnecting four processing elements.

The *asynchronous NoC* is a 6x6 crossbar, but it is used in fact as a 5x5 crossbar, with for processing elements and a direct output parallel communication link. There is no pipelining in this version of the ANoC even though higher throughput could be easily obtained by applying asynchronous pipelining techniques. The ANoC delivers both arbitration and transaction services (section 2.4). The arbitration policy is a non-interruptible two-level priority policy. When concurrent incoming requests need arbitration, a request with the high-priority level is selected and low-priority level requests are suspended. For equal priority-level concurrent requests, a First-In First-Granted (FIFG) policy is used. A former selected channel can not be interrupted by an incoming higher priority-level request during a burst mode data transfer. The high-priority level is assigned to the MIPS processors. The transaction service delivers burst mode or simple on-flit packet transfer modes, plus a special service called Indirect-Response (IR). In IR mode, a peripheral A, initiator of a communication, notify the receiver B not to answer to A, but to a third peripheral C.

The four processing elements are:

- *Two independently clocked MIPS* with local RAM banks and serial communication links. One MIPS is running at 45MHz for interfacing purposes whereas the other MIPS is running at 50MHz for number crunching applications.
- *A self-timed DES module* (Data Encryption Standard).
- *A shared RAM bank*.

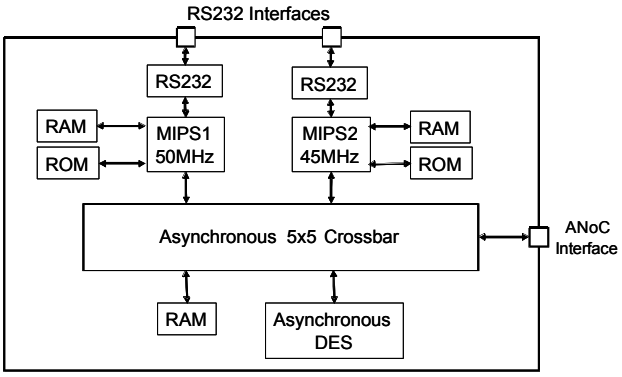

Fig. 4. Structure of PACMAN case-study version for FPGA implementation

5.2 Performance of the communications

The Stratix Altera FPGA platform we have been using successfully supports the PACMAN architecture implementation. Characteristics of the FPGA are the following:

- device EP1S40F780C5 (40k gates),
- pin count 780,
- speed grade 5

Implementing a 6x6 crossbar (used in fact as a 5x5) Asynchronous NoC onto the FPGA involves:

- 13458 LUTs and 0 registers for packet router modules (see section 2.3)
- for communication between peripherals, interfaces including WA, SPI and PT modules are involved (see section 2.2). Each interface involves 218 LUTs and 90 registers.

Table 1 shows latencies and throughput of the ANoC without interfaces. Cycle time is the direct flit latency from one packet router (Emitter module) to another packet router (Receiver module) plus the backward acknowledge propagation time. Table 2 shows latencies and throughput between MIPS1 (50MHz) and MIPS2 (45MHz) across ANoC and interfaces.

As mentioned before, these data transmission rates can easily be improved with pipelining.

Table 1. Latencies and throughput from packet router to packet router in the ANoC

	Direct latency (ns)	Cycle time (ns) (delay between flits)	Throughput (Mflit/s)	Throughput (MBps)
Burst Mode	43,3	57,2	17,5	630
Simple Mode	45,9	61,7	16,2	583,2

Table 2. Latencies and throughput between MIPS1 and MIPS2 across ANoC and interfaces

Interface clock frequency (MHz)	Data transfer mode and packet	Direct latency (ns)		Cycle time (ns) (delay between flits)	Throughput (Mflit /s)	Throughput (MBps)
		start-paquet flit	body or end-of-packet flit			
50	Burst	50	31	120	8,3	266,6
50	Single	76	57	141	7,1	226,9
66	Burst	60	53	105	9,5	304,7
66	Single	72	64	121	8,2	264,5
90,9	Burst	43	31	77	12,9	415,6
90,9	Single	68	62	100,4	9,9	318,7

Conclusion

In this paper we provide a simple and flexible structure of Asynchronous NoC for GALS systems requiring efficient communications at low-complexity and low-power costs. Such a structure is modular, robust and scalable. The interconnect topology generator delivers several configurable interconnect topologies which facilitate the system architecture exploration, helped by a scalable and easy-to-plug (flexible?) self-timed FIFO. Then a low-latency FIFO version can be instantiated in the final architecture. Using a multi-clock FPGA allows a fast prototyping of a complex ANoC-centric GALS system with mixed synchronous and asynchronous components. First result analysis gives promising ANoC abilities to deliver fast and robust communications. Another PACMAN version has been successfully prototyped onto the Altera Stratix FPGA. This is a distributed architecture implementing four independently clocked MIPS interconnected by the AnoC. Closely analyses of the FPGA platform are currently performed to extract complete results from these two PACMAN implementations, in order to improve both ANoC and GALS system design.

Prospective works will be to extend the topology generator to the other regular distributed topologies, with a large variety of arbitration policies and transaction services. Another work will be to integrate formal verification methods into the design flow. The aim is to deliver a dedicated synthesis tool for asynchronous interconnect networks generation.

References

[1] F. K. Gürkaynak, S. Oetiker, N. Felber, H. Kaeslin et W. Fichtner, Is there hope for GALS in the future ?, proceedings of the 4th Asynchronous Circuit Design Workshop (ACID 2004), Turku, Finland, June 28-29, 2004.

[2]J. B. Rigaud, J. Quartana, L. Fesquet et M. Renaudin, Modeling and design of asynchronous priority arbiters for on-chip communication systems, proceedings of the VLSI-SOC'01 Conference on Very Large Scale Integration Systems.

[3] E. Beigné, F. Clermidy, P. Vivet, A. Clouard, M. Renaudin, An Asynchronous NOC Architecture Providing Low Latency Service and its Multi-Level Design Flow, 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC), March 14-16, New York, USA, 2005.

[4] T. Villiger, H. Kaeslin, F. Gurkaynak, S. Oetiker et W. Fichtner, Self-timed Ring for Globally-Asynchronous Locally-Synchronous Systems, Ninth International Symposium on Advanced Research in Asynchronous Circuits and Systems, ASYNC'03, Vancouver, Canada, May 12-16 , 2003.

[5]E. Bolotin, E. Cidon, R. Ginosar et A. Kolodny, QNoC : QoS architecture and design process for network on chip, *Journal of Systems Architecture*, no. June 2003.

[6]W. J. Bainbridge et S. Furber, CHAIN: A Delay Insensitive CHip Area INterconnect, *IEEE Micro*, vol. 22, no. 5, pp. 16-23, September/October 2002.

[7]W. O. Lovett, CHip Area Network Simulation, Master of Science, University of Manchester, 2002.

[8]T. Q. Ho, J. B. Rigaud, M. Renaudin, L. Fesquet et R. Rolland, Implementing Asynchronous Circuits on LUT Based FPGAs, Proceedings of the Field-Programmable Logic and Applications, Reconfigurable Computing Is Going Mainstream, 12th International Conference on Field-Programmable Logic and Applications Conference, FPL 2002, Montpellier, France, September 2-4, 2002.
[9]J. Quartana, S. Renane, A. Baixas, L. Fesquet, M. Renaudin, GALS Systems Prototyping using Multiclock FPGAs and Asynchronous Network-on-Chips, 15th Field-Programmable Logic and Applications Conference (FPL'05), August 24-26, Tampere, Finland.
[10] A.J. Martin, Programming in VLSI: from communicating processes to delay-insensitive circuits, in C.A.R. Hoare, editor, *Developments in Concurrency and Communication*, UT Year of Programming Series, 1990, Addison-Wesley, p. 1-64.
[11] Anh Vu Dinh Duc, Laurent Fesquet, Marc Renaudin, Synthesis of QDI Asynchronous Circuits from DTL-style Petri-Net IWLS-02, 11th IEEE/ACM International Workshop on Logic & Synthesis, New Orleans, Louisiana, 2002.
[12] A.V. Dinh Duc, J.B. Rigaud, A. Rezzag, A. Sirianni, J. Fragoso, L. Fesquet, M. Renaudin, TAST CAD Tools: Tutorial, tutorial given at the International Symposium on Advanced Research in Asynchronous Circuits and Systems ASYNC'02, Manchester, UK, April 8-11, 2002, and at the ACiD Summer School on "Asynchronous circuits design", Grenoble, France, July 15-19, 2002. TIMA internal report ISRN:TIMA-RR-02/07/01—FR, http://tima.imag.fr/cis.
[13] Ph. Maurine, J.B. Rigaud, F. Bouesse, G. Sicard, M. Renaudin, Static Implementation of QDI asynchronous primitives, PATMOS'03 - 13th International Workshop on Power and Timing Modeling, Optimization and Simulation. Torino, Italy, September 10-12, 2003.
[14] R. Ginosar, Synchronization and Arbitration, Proceedings of the ACiD Summer School on Asynchronous Circuit Design, Grenoble, France, July 15-19 2002.
[15] Y. Semiat et R. Ginosar, Timing Measurements of Synchronization Circuits, Ninth International Symposium on Advanced Research in Asynchronous Circuits and Systems, ASYNC'03, Vancouver, Canada, May 12-16 , 2003.
[16] R. Ginosar, Fourteen ways to fool your Synchronizer, Proceedings of the Ninth International Symposium on Advanced Research in Asynchronous Circuits and Systems, ASYNC'03, Vancouver, Canada, May 12-16 , 2003.
[17] T. Chelcea et S. M. Nowick, Robust Interfaces for Mixed-Timing Systems, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 12, no. 8, 2004.
[18] R. Ho, J. Gainsley et R. Drost, Long wires and asynchronous control, Proceedings of the Asynch'04, 2004.
[19] A. Bystrov, D. J. Kinniment, A. Yakovlev, Priority Arbiters, in *International Symposium on Advanced Research in Asynchronous Circuits and Systems* (ASYNC), Eilat, Israel, April 2000, pp. 128-137.
[20] F. Karim, A. Nguyen et S. Dey, An Interconnect Architecture for Networking Systems on Chips, *IEEE Micro*, vol. 22, no. 5, pp. 36-45, September/October 2002. Integration, Montpellier, France, 3-5 Dec. 2001.
[21] C. Piguet, M. Renaudin, T. Omnés Low-power systems on chips (SOCs), Proceedings of the DATE Conference, Munich, Germany, 2001.