A Fully-Asynchronous Low-Power Framework for GALS NoC Integration

Yvain Thonnart, Pascal Vivet, Fabien Clermidy
CEA-LETI, MINATEC
Grenoble, France
{yvain.thonnart, pascal.vivet, fabien.clermidy}@cea.fr

***Abstract* — Requiring more bandwidth at reasonable power consumption, new communication infrastructures must provide adequate solutions to guarantee performance during physical integration. In this paper, we propose the design of a low-power asynchronous Network-on-Chip which is implemented in a bottom-up approach using optimized hard-macros. This architecture is fully testable and a new design flow is proposed to overcome CAD tools limitations regarding asynchronous logic. The proposed architecture has been successfully implemented in CMOS 65nm in a complete circuit. It achieves a 550Mflit/s throughput on silicon, and exhibits 86% power reduction compared to an equivalent synchronous NoC version.**

I. INTRODUCTION

With many hundreds of million transistors available on a single chip, the era of Manycores System-on-Chip (MPSoC) has become a reality. The available synchronous bus architectures cannot meet the increasing demands of communication between the processing elements, because the long-wire loads and resistances result in slow signal propagation. On the other hand, the advantages of Network-on-Chip (NoC) [1] are numerous: high scalability and versatility, high throughput with good power efficiency. A NoC can be implemented using synchronous logic [2] but designers still face the issue of generating a global clock tree over the chip. Multi synchronous design can help but with extra latency due to re-synchronization cost within each router along the paths.

The NoC distributed communication architecture is perfectly adapted to the Globally Asynchronous Locally Synchronous (GALS) paradigm [3] where ÏP units are implemented with standard synchronous design methodologies on an independent timing domain, while the NoC itself is implemented in asynchronous logic. The GALS paradigm also offers a natural way to implement Dynamic Voltage and Frequency Scaling (DVFS) thanks to fully decoupled synchronous islands. Even if asynchronous logic may bring additional benefits such as low power and low noise, asynchronous logic is yet not well supported by CAD tools. A recent asynchronous interconnect tool chain, CHAINworks™ [4], presents strong benefits but offer efficient point to point communications rather than a complete NoC protocol and regular framework. Several asynchronous NoC architectures have been designed and synthesized, but not yet fabricated [5][6].

In this paper, we present the design of a 550 *Mflit/s* asynchronous Network-on-Chip, fully implemented in Quasi – Delay – Insensitive (QDI) asynchronous logic [7]. The proposed ANoC architecture is fully testable. A new asynchronous logic timing optimization is proposed during Place & Route to overcome CAD tool limitations. The proposed architecture is implemented with optimized hard-macros for the NoC routers and NoC GALS interfaces, which are used at top level in a bottom-up design flow.

The outline of the paper is as follows: in section II is presented the architecture and details of its building elements. In section III is presented the proposed implementation flow, while in section IV are presented the area, speed and power results, as well as a comparison with two equivalent synchronous NoC versions.

II. ANOC ARCHITECTURE AND DESIGN

A. ANoC Main Principles

The proposed ANoC framework allows an easy assembly of a complex GALS SoC made of independent synchronous IP units, relying on asynchronous routers and links assembled in a mesh topology for communication as seen on Figure 1.

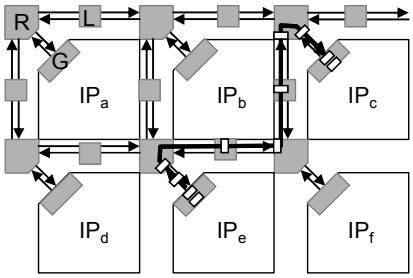

Figure 1. ANoC framework:
Routers (R), Links (L) and GALS interfaces (G)

Data is transmitted in packets made of several flits following the format of Figure 2, which transit in the NoC in a wormhole fashion, along the path specified in the header flit.

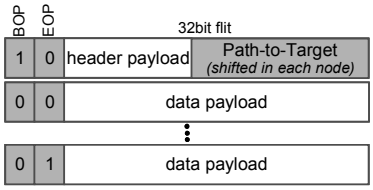

Figure 2. ANoC packet format

978-3-9810801-6-2/DATE10 © 2010 EDAA

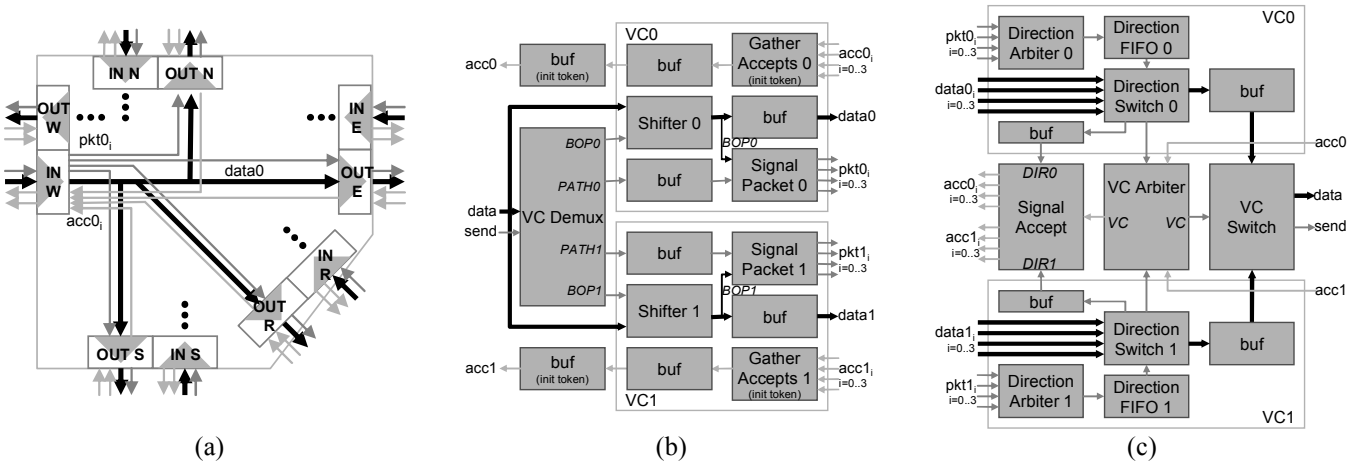

Figure 3. ANoC router architecture (a) and micro-architecture of input ports (b) and output ports (c)

B. ANoC Router

The router is in charge of routing and arbitrating the packets inside the NoC using the information contained in the packet format presented in Figure 2. Given the 2D-mesh topology of the NoC, the router is made of five input ports and five output ports; no central arbitration is needed. The input port routes the consecutive flits of a packet according to the path to target present in the header flit (marked with a BOP), which is shifted for use in the next router. The output port arbitrates and multiplexes the packets from all connected input ports starting from the BOP, and keeps a path open until the EOP is reached.

The router has two independent virtual channels for improved quality of service separating best effort traffic and priority out-of-band signaling. A backpressure mechanism is used for flow control in order to allow for at most two flits in each virtual channel of an input port, so that the shared NoC link is never congested. Figure 3 presents the micro-architecture of the router.

The input port is made of a first de-multiplexing stage (*VC Demux*), which routes flits to their corresponding VC queues. Then, the *Shifter* stage modifies the routing information as needed, and the flit is stored in a buffer stage waiting for consumption by one of the output ports. The notification from the input port to the good output port is done only once at the beginning of packet in *Signal Packet*.

The output port first performs arbitration between directions within each VC, and only then between VCs. *Direction Arbiter* performs fair arbitration of the possible new packet requests from the input ports, and generates a single command token to the *Direction Switch* that will be consumed only at the end of packet. Finally, the *VC Arbiter* arbitrates at flit level between the two VCs, and commands the *VC Switch*.

The send/accept backpressure mechanism is also initiated by the *VC Arbiter*: the downstream accept token is consumed, and an upstream accept token is generated, and forwarded, through *Signal Accept* in the output port and *Gather Accepts* in the input ports. The loop formed by two consecutive routers is initialized with as many tokens as there is room in the inputs queues.

C. ANoC Pipelined Links

Because the synchronous islands connected to the NoC may have an important area, the links between routers may be long. Due to the 4-phase handshake of the QDI 4-Phase asynchronous protocol (Figure 4), this length would have and impact on performance. In order to overcome this, asynchronous pipelining is added to the NoC links: it reduces the cycle time of the longest paths without adding a latency penalty compared to classical inverter/buffer wire buffering. Figure 5 presents a typical 4-Rail QDI interconnect, and the associated WCHB pipelining [4],[7].

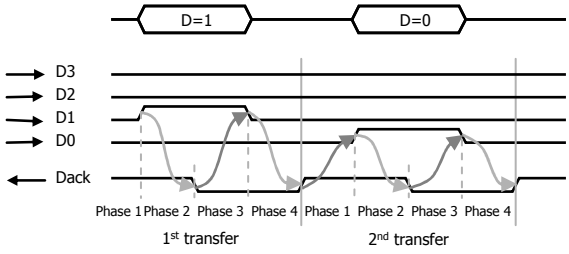

Figure 4. QDI 4-Rail reqest/acknowledge asynchronous protocol

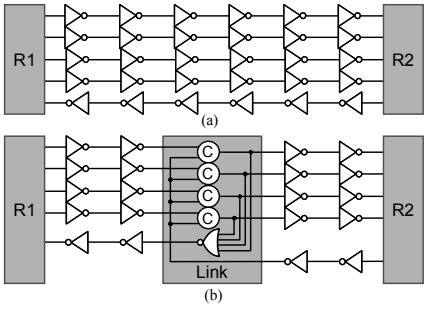

Figure 5. Buffered (a) and pipelined (b) QDI 4-rail links

The Muller C-elements induce a forward latency similar to that of two inverters, while the wire delay penalty is divided by two in the handshake loops. The links can be pipelined as presented, as often as needed to hide the impact of long wire delays on the NoC performance. A pipeline stage every millimeter reveals to be enough for 65 *nm* CMOS.

D. ANoC GALS Interface

The synchronous IP cores are connected to the routers by means of a GALS interface, which has two objectives: (a) re-synchronize the asynchronous NoC protocol with the synchronous domain; (b) generate a local clock with a programmable frequency.

In GALS design, the major challenge is to robustly interface different timing domains, while providing high throughput and low latency. Two kinds of solutions have been explored: pausable clocking scheme and FIFO based design. In pausable clocking, in order to avoid meta-stability, the idea is to generate locally the clock and pause or stretch this clock whenever a transfer from/to the asynchronous domain occurs through the interface [8]. This elegant design style is nevertheless limited in throughput, mainly due to the clock tree insertion time constraint [9], which can be severe with large synchronous IP blocks. On the other hand, in FIFO-based design, the idea is to use standard synchronizers to synchronize the synchronous and asynchronous domains (actually the read/write pointers) and build a FIFO to sustain the throughput by hiding the synchronizer latency [10].

For the proposed architecture, we propose to mix both types: the GALS interface is composed (Figure 6) of a new asynchronous ⇔ synchronous FIFO using Johnson Encoding in order to offer small and efficient FIFO, and of a locally generated clock using a pausable clocking scheme in order to offer a programmable frequency with a fast and robust programming interface (more detail is given in [11]).

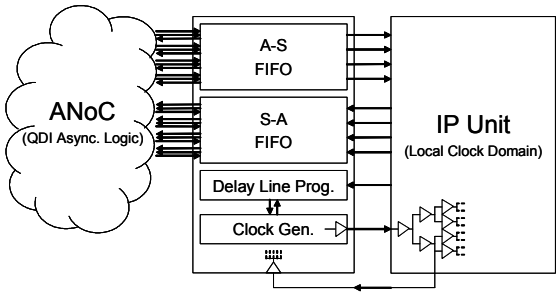

Figure 6. GALS adapter architecture

The GALS interface implements the NoC protocol: on the asynchronous side using the 4-phase / 4-rail protocol (Figure 4); on the synchronous side using a send/accept protocol. The two Virtual Channels (VC0 & VC1) are supported by the interface but without VC arbitration, to reduce FIFO number and logic complexity. The interface achieves one flit transfer per cycle, with a latency of 2 cycles. The two FIFOs have a size of 5 words to achieve maximum throughput while hiding the internal synchronization (2 Flip-Flops) of the Johnson Encoded read/write pointers.

The interface generates a local and programmable clock in the [400MHz - 1GHz] range with scaling factors. Its internal delay line is only made of standard-cell elements. The generated clock is used by the IP and is also fed back in the interface since used by its synchronous part. As a conclusion, the IP is a fully independent timing domain. A new frequency can be safely programmed by the IP itself in less than 3 cycles, without halting any IP computation or NoC communication.

E. ANoC Design-for-Test Architecture

Regarding testability of the proposed ANoC architecture, two issues need to be solved: the test of the *synchronous* IP units and the test of the *asynchronous* NoC itself.

For the test of IPs within the NoC, a test wrapper compatible with the IEEE 1500 standard [11] has been developed. Test data of each IP are encapsulated into packets that are transported at high bandwidth through the NoC. There is thus no extra Test Access Mechanism (TAM) hardware cost and the scalability of the test is improved [13]. Classical fault coverage of the synchronous IPs is then obtained using standard ATPG tools.

To test the asynchronous NoC itself, since asynchronous logic is difficult to test with standard design techniques, a novel asynchronous DfT architecture adapted to the asynchronous router has been developed [14]. The main idea is to encapsulate each router by an asynchronous test wrapper in order to improve its controllability and observability. The test wrappers are used: (a) to insert test vectors to the elements-under-test (routers and network links) and to get out the test results; (b) with network links, to establish high bandwidth asynchronous TAMs to transport test data.

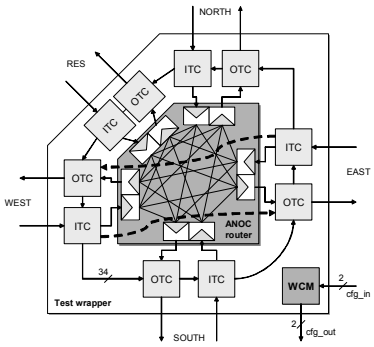

Figure 7. ANoC router with its test wrapper

For each router, the test wrapper (Figure 7) is composed of 5 input test cells (ITC), and 5 output test cells (OTC), plus the additional Wrapper Control Module (WCM) to control all test cells. The ITCs and OTCs are alternatively interconnected to establish a boundary-scan path around the router plus some additional bypass paths. The test wrappers are individually controlled through a configuration chain, while a centralized test controller performs test pattern generation and analysis using the configuration chain protocol. There is no change to the chip external interface except the additional 2-bit test configuration chain.

Using the proposed test protocol, it is then possible to test iteratively each link and each router within the NoC topology. By using successive data values for each router port, thanks to QDI logic stuck-at fault stall property, a minimal number of test vectors (320) is finally required per router. With a single-stuck-at fault model on both inputs and outputs, test coverage of 99.86% per router is achieved. In terms of test efficiency, 2 *ns* per configuration chain symbol is achieved, this provides a test time per router of 32 μs, and about 0.7 *ms* for a complete NoC topology with 20 routers.

F. ANoC Router with Automatic Power Down

Thanks to their local handshaking, asynchronous circuits are automatically in standby state when inactive. In order to further reduce the interconnect power consumption, each router can also integrate an automatic power down mechanism. Thanks to the robustness and locality of asynchronous logic, fast and reliable activity detection is done, which is used to power down unused or under-loaded NoC routers and thus save additional static or dynamic power. A detailed presentation of this proposal can be found in [15].

III. QDI Logic Implementation Methodology

To implement the targeted QDI asynchronous logic, a standard-cell based approach is preferred, and even mandatory for easy integration. The STMicroelectronics CMOS 65 *nm* standard-cell libraries have been enriched with a set of additional cells [14] (approximately 30 cells of Muller C-elements, Mutual exclusion cells and Synchronizers). These "asynchronous" cells are fully compatible with the standard-cell design flow (layout rules, .lib timing characterization, back-annotable Verilog model ...).

A. ANoC Tiles and Floor-Planning

In order to offer an efficient top level implementation, as well as in order to hide the inherent complexity of asynchronous logic, it is chosen to implement both the routers and GALS interfaces as hard macros. Due to the lack of asynchronous CAD design and verification tools, these two macros are implemented using an ad-hoc design style [7] and are fully validated by extensive simulations. As presented in Figure 8, the proposed architecture is implemented as a building-block design. During layout floor-planning, each NoC synchronous unit is reserved a square box, in which the top-left corner is reserved for the associated router, to which is connected the GALS interface.

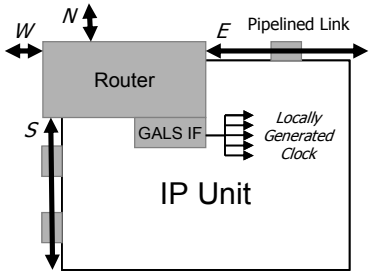

Figure 8. ANoC tile floor-planning

The pin-out of the two hard macros has been defined to allow direct pin-to-pin connections: between North & South ports, West & East ports for the router, and similarly for the GALS interface. Each link represents 92 signals, which defines 184 pins for a NoC bi-directional port, assigned in two metal layers (M2/M4 or M3/M5 according to the routing preferred directions). At top-level, in order to implement and route the pipelined links, minimal space is reserved according to available routing tracks using only two metal layers. In the case of non homogeneous NoC structure, a solution to help floor-planning consists in defining a constant height for NoC units on a given NoC row, and distinct widths according to each NoC unit area. The pipelined links are implemented as soft-macros to ease integration for these small elements and provide efficient buffering during timing optimization (see section III.B). This provides a fully GALS integration scheme, in which each NoC synchronous units has its own local clock: there is no synchronous timing constraint between distinct NoC units, only a few additional top level global signals may need synchronous constraints (chip I/Os). Hence, hierarchical floor-panning can be applied without derivation of complex timing constraints at top level.

B. QDI Logic Timing Modeling & Optimization

Even if asynchronous QDI logic is fully functional whatever the gates and wires delays [7], it is yet mandatory to optimize the logic to achieve performance. Proper asynchronous logic buffering is required, mainly for the long NoC link wires. Since asynchronous logic is not well supported by CAD tools, it has been chosen to model the asynchronous logic as a *fake* synchronous logic, using the global reset signal as a *dummy* clock to break the asynchronous logic timing loops. As seen in Figure 4 presented earlier, the timing loops that exist between forward and backward logic paths of the asynchronous handshaking can be broken in the C element of the pipeline stage. The C-element is then modeled for the CAD tools as a flip-flop using reset signal as a dummy clock. The associated clock period corresponds to only one phase of the 4-phase handshaking protocol, which thus requires constraining the dummy clock frequency to four times the target asynchronous period.

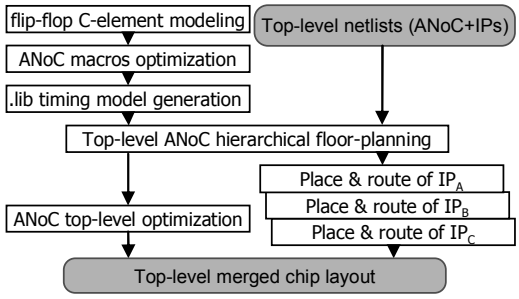

Figure 9. GALS NoC design flow

Throughout the whole design flow (Figure 9), the asynchronous logic reset is modeled as a clock, but which is always maintained with a virtual zero delay to remove any skew problem on this heavily loaded global signal, since its skew is clearly meaningless. Using this strategy, timing optimization is performed for implementing the two hard macros (router and GALS interface); abstract timing models of the macros are then generated using standard CAD tool features: the generated ".lib" models contain asynchronous timing paths of the I/Os related to the reset signal. Using these models, top-level optimization is achieved on the NoC link long wires, through proper buffering under timing constraints. The obtained asynchronous handshake cycle time is at least four times the *dummy* clock period. Post-layout back-annotated simulations allow checking the maximum sustainable performance. To conclude, even if CAD tool does not support properly asynchronous logic, by using optimized hard-macros with a fake synchronous modeling, hierarchical floor-planning offers a divide and conquer GALS scheme, while timing optimization at top level is still under control.

IV. IMPLEMENTATION & BENCHMARKING WITH A SYNCHRONOUS NOC

A. Asynchronous NoC Implementation

The architecture presented in Section II has been implemented using the methodology described in Section III, and the complete proposed framework has been deployed and fabricated using the STMicroelectronics CMOS 65 nm LP technology in a chip dedicated to flexible baseband processing for 3G/4G wireless telecommunication applications [18].

The resulting architecture is made of 15 asynchronous routers arranged in a 5x3 2D-mesh, with a synchronous external interface, and interconnecting 22 independent synchronous islands. Depending on their different sizes, NoC links of various lengths are needed on the chip. Various depths of pipelining were added to these links to sustain a global throughput on the whole NoC, and avoid bandwidth limitations due to a longer handshake loop. With an average size of 1 mm^2 for the processing units as in the current implementation, the whole NoC structure represents about 15% of the chip area, which is an acceptable cost considered the dataflow structure of telecommunication applications.

B. Equivalent Synchronous NoC Implementations

In order to precisely benchmark the asynchronous framework presented in this paper against synchronous solutions, a fully synchronous NoC has also been developed following the same requirements. The initial constraint was that the target implementation of the NoC should be transparent to both the synchronous islands and the external interface, so that the logical view of the NoC remains the same. Besides, the comparison has been led up to physical layout of the chip in all the implementations, and the global chip floor-planning was kept for the synchronous NoC, aside from an additional logical pin for the NoC clock.

Hence, the synchronous implementation is based on the same NoC protocol as presented Figure 2, with a 32-bits data-path and 2 Virtual Channels. Moreover, bi-synchronous FIFOs were inserted at each IP interface with the NoC to offer GALS-like decoupling and DVFS at IP level. Finally, the synchronous NoC has been implemented with standard fully scannable flip-flops and clock gating; this was done in order to have a fair comparison as regards testability and power consumption.

It was chosen for this benchmarking to implement a fully synchronous NoC with a unique NoC clock domain at top level. Indeed, if implementable, this appears to be the solution with the lowest overhead compared to mesochronous or multisynchronous versions, which require re-synchronization FIFOs at each timing domain boundary. Those FIFOs would lead to higher area, latency and power consumption compared to a fully synchronous version with a single clock domain. The actual limitation of this fully synchronous version is the clock-tree distribution that may be challenging for back-end tools, but yet achievable on a regular enough design.

As regards the Power-Performance-Area tradeoff for the synchronous version of the NoC, two versions of the synchronous NoC router have been implemented, in order to get the most fair comparison points: a low-area one with a single clock cycle to cross the NoC router and the link, and a high-performance one with two clock cycles per router and possibly another cycle on the link. The single cycle version of the router registers the flits at the input ports, and then combinationally routes and arbitrates to the output ports and the links, while the pipelined version adds a register stage after arbitration within each VC, just before multiplexing the VCs on the output link.

C. Experimental Setup

Hence, two comparison chips with both versions of the synchronous NoC were realized up to layout in the same STMicroelectronics CMOS 65 *nm* LP technology, with the same floorplan as the asynchronous version, and all required information were collected to run post-layout back-annotated simulations of these chips.

On these three circuits was mapped part of the real telecom application running on the prototype presented in [17][18], i.e. the CFO (Carrier Frequency Offset) estimation and correction of the 3GPP-LTE receiver chain. This leads to a total of about 150000 flits transiting in the NoC in about 250 μs. For this application, a 200 *Mflit/s* NoC throughput is enough to sustain real-time constraints, and the synchronous versions were hence clocked at only 200 *MHz* to minimize power consumption. Post-layout back-annotated simulation was performed on the nominal PVT corner, and power estimation was done using Synopsys PrimePower on this simulation. Performance and power figures from the asynchronous simulation were successfully matched against the real silicon measurements with an error of less than 10% on all observable results (throughput, IPs power, …).

V. PERFORMANCE RESULTS

In Figure 10 is presented the results of the comparison between the asynchronous NoC implementation and its synchronous counterparts.

As regards area, the asynchronous version is clearly larger than the synchronous ones, as it could be expected from QDI logic, which suffers from multiple-rail encoding: timing flexibility and robustness come at the cost of about twice the same logic as what is needed with binary encoding in the synchronous versions.

As regards performance, thanks to QDI asynchronous logic intrinsic auto-regulation by the handshake loops, it is possible to achieve nominal case speed results on the asynchronous implementation while synchronous logic is characterized in worst case conditions and should not be used above the worst case frequency. The usual gain is between 30% and 50%, depending on the targeted technology. Here, the asynchronous version achieves a 550 *Mflit/s* throughput on nominal conditions (as verified on silicon measurements), which corresponds to the target timing optimization frequency as defined in the methodology presented in Section III.

Even higher throughputs were attained on small ANoC testcases up to 750 *Mflit/s*. As for the high-performance synchronous version, a maximum clock frequency of 480 *MHz* (worst case) was attained at the cost of a deep clock-tree (5 *ns*

	Asynchronous NoC					Synchronous NoC		
	Bare Router Version	Router with Auto Power Down		Router with Test Wrapper	Async/Sync GALS IF	Low-Area Router single stage	High-Perf Router pipelined 2 stages	Sync/Sync GALS IF
		Power On	*Power Down*					
Area	0.17 mm²	0.20 mm²		0.23 mm²	0.014 mm²	0.053 mm²	0.094 mm²	0.014 mm²
Flit Cycle Time	1.8 ns	1.8 ns	7.2 ns	1.8 ns	2.0 ns	3.6 ns	2.1 ns	2.0 ns
Flit Throughput	550 Mflit/s	550 Mflit/s	140 Mflit/s	550 Mflit/s	500 MHz	280 MHz	480 MHz	500 MHz
Flit Latency	2.3 ns	2.5 ns	5.8 ns	2.6 ns	4.8 ns (round trip)	3.6 ns (1 cycle)	4.2 ns (2 cycles)	8 ns (round trip)
Supply Voltage	1.2 V	1.2 V	0.6 - 0.8 V	1.2 V	1.2 V	1.2 V	1.2 V	1.2 V
Static Power	240 µW	250 µW	100µW	316 µW	10 µW	43 µW	105 µW	8 µW
Idle Power	240 µW	250 µW	100 µW	316 µW	1100 µW	1980 µW	5170 µW	1100 µW
Energy per flit	30 pJ/flit	30 pJ/flit	14 pJ/flit	37 pJ/flit	3 pJ/flit	26 pJ/flit	24 pJ/flit	2 pJ/flit
Appl. Tot. Power	11.9 mW	11.9 mW	8.8 mW (*)	13.7 mW	—	33.6 mW	82.6 mW	—

(*) Auto-power-down mode is applied when routers are idle; routers in use are in power-on mode to meet realtime constraints on this application

Figure 10. Asynchronous NoC performance results and benchmarking against synchronous NoC implementations

insertion time) to minimize the skew at the FF leaves (500 *ps*). This performance, though, is quite comparable to the asynchronous one given the worst case corner. The low-area synchronous router only achieves 280 *MHz*, due to the long combinational paths required to arbitrate all the flows. Finally, the asynchronous version offers a reduced latency (almost a ratio of 2): for a NoC path of 5 routers, ANoC latency is 17.3 *ns* compared to 29 *ns* for the synchronous version.

As regards leakage, the asynchronous router is quite behind the synchronous versions, although in about the same ratio as for area for the high-performance synchronous router. This is yet partly due to the availability of only the low-Vt option for the asynchronous cells. However, static current is the only power dissipation when the asynchronous router is idle: there is no dynamic activity at all. On the contrary, synchronous versions greatly suffer from the clock switching even though clock gating is performed, with up to 5 *mW* idle consumption for the high-performance synchronous router. Overall, on the telecom application implemented, the energy budget on the whole NoC (15 routers) is drastically reduced from 82.6 *mW* on the high-performance synchronous version down to 11.9 *mW* on the asynchronous version.

VI. Conclusion

The proposed asynchronous Network-on-Chip is a technology ready for assembling complex manycores chips. It provides a GALS NoC infrastructure with 550 *MFlit/s* throughput and five-times power reduction compared to a synchronous equivalent. The hard macro methodology provides a guaranteed performance during the physical implementation, reducing Time-To-Market. Finally, a complete CMOS 65 *nm* demonstrator has been used to show the efficiency of the proposed scheme.

Acknowledgment

The authors would like to thank their colleagues from the TIMA laboratory for the development of the TAL65 asynchronous standard cell library.

References

[1] P. Guerrier, A. Greiner, “A Generic Architecture for On-chip Packet-switched Interconnections”, in *Proceedings of the Design And Test in Europe Conference (DATE’2000)*, pp 250-256, march 2000.

[2] M. Dall'Osso, G. Biccari, L. Giovannini, D. Bertozzi, L. Benini, "Xpipes: a Latency Insensitive Parameterized Network-on-chip Architecture for Multi-Processor SoCs", in *Proceedings of the IEEE International Conference on Computer Design (ICCD'03)*, pp.536, 2003.

[3] M. Krstic, E. Grass, F. K. Gürkaynak, P. Vivet, “Globally Asynchronous, Locally Synchronous Circuits:Overview and Outlook”, *IEEE Design and Test of Computers*, vol 24, n°5, pp 430-441, sept. 2007.

[4] J. Bainbridge, CHAINWorks, Silistix, http://www.silistix.com

[5] R. Dobkin, V. Vishnyakov, E. Friedman, R. Ginosar, “An Asynchronous Router for Multiple Service Levels Networks on Chip”, in *Proceedings of the 11th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC’05)*, pp. 44-53, march 2005.

[6] T. Bjerregaard, J. Sparso, “A Router Architecture for Connection-Oriented Service Guarantees in the MANGO Clockless Network-on-Chip”, in *Proceedings of the IEEE Design, Automation and Test in Europe Conference (DATE’05)*, pp. 1226-1231, march 2005.

[7] A. J. Martin, M. Nyström, “Asynchronous techniques for system-on-chip design”, *Proc. of the IEEE,* vol. 94, n°6, pp 1089-1120, june 2006.

[8] R. Mullins, S. Moore, “Demystifying Data-Driven and Pausible Clocking Schemes”, in *Proceedings of the IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC’07)*, pp 175-184, march 2007.

[9] R. Dobkin, R. Ginosar, C. Sotiriou, “High Rate Data Synchronization in GALS SoCs”, *IEEE Transactions on VLSI Systems*, vol 14, n°10, pp 1063-1074, october 2006.

[10] T. Chelcea, S. M. Nowick, “Robust Interfaces for Mixed-Timing Systems”, *IEEE Transactions on VLSI Systems*, vol. 12, n° 8, pp 857-873, august 2004.

[11] Y. Thonnart, E. Beigné, P. Vivet, “Design and Implementation of a GALS Adapter for ANoC based Architectures”, in *Proceedings of the 15th International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC’09)*, pp 13-22, april 2009.

[12] IEEE 1500 working group. IEEE 1500 Standard for Embedded Core Test. *http://grouper.ieee.org/groups/1500*.

[13] A.M. Amory, K. Goossens, E.J. Marinissen, M. Lubaszewski, and F. Moraes, “Wrapper Design for the Reuse of Networks-on-Chip as Test Access Mechanism”, in *Proceedings of the IEEE European Test Symposium (ETS’06)*, pp. 213–218, may 2006.

[14] X.-T. Tran, J. Durupt, Y. Thonnart, V. Beroulle, C. Robach, “Design-for-Test Approach of an Asynchronous Network-on-Chip Architecture and its Associated Test Pattern Generation and Application”, *IET Journal on Computers and Digital Techniques,* vol. 3, n°5, pp. 487-500, sept. 2009.

[15] Y. Thonnart, E. Beigné, A. Valentian, P. Vivet, “Power Reduction of Asynchronous Logic Circuits using Activity Detection”, *IEEE Transactions on VLSI Systems*, vol. 17, n° 7, pp. 893-906, july 2009.

[16] P. Maurine, J.B. Rigaud, F. Bouesse, G. Sicard, M. Renaudin, “Static Implementation of QDI Asynchronous Primitives”, in *Proceedings of the 13th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS’03)*, pp. 181-191, sept. 2003.

[17] F. Clermidy, R. Lemaire, X. Popon, D. Kténas, Y. Thonnart, “An Open and Reconfigurable Platform for 4G Telecommunication: Concepts and Application”, In *Proceedings of the 12th Euromicro Conference on Digital System Design (DSD’09)*, august 2009.

[18] F. Clermidy, C. Bernard, R. Lemaire, J. Martin, I. Miro-Panades, Y. Thonnart, P. Vivet, N. Wehn, “A 477mW NoC-Based Digital Baseband for MIMO 4G SDR”, in *Proceedings of IEEE International Solid-State Circuits Conference (ISSCC’2010)*, february 2010.