Hermes-A – An Asynchronous NoC Router with Distributed Routing

Julian Pontes, Matheus Moreira, Fernando Moraes, Ney Calazans

Faculty of Informatics, PUCRS, Porto Alegre, Brazil
{julian.pontes, matheus.moreira, fernando.moraes, ney.calazans}@pucrs.br

Abstract. This work presents the architecture and ASIC implementation of Hermes-A, an asynchronous network on chip router. Hermes-A is coupled to a network interface that enables communication between router and synchronous processing elements. The ASIC implementation of the router employed standard CAD tools and a specific library of components. Area and timing characteristics for 180nm technology attest the quality of the design, which displays a maximum throughput of 3.6 Gbits/s.

Keywords: asynchronous circuits, network on chip.

1. Introduction

Interest in asynchronous circuits has increased due the growing limitations faced during the design of synchronous System on a Chip (SoC) circuits, which often result in over constrained design and operation [1]. However, asynchronous computer aided design (CAD) tools still have to undergo a long evolutionary path before being accepted by most designers. The lack of such tools renders difficult the access of traditional circuit designers to the full capabilities of asynchronous circuits.

Globally Asynchronous Locally Synchronous (GALS) design techniques may help overcoming limitations of synchronous design while maintaining a mostly synchronous design flow [2]. GALS techniques simplify the task of reaching the overall timing closure for SoCs, but typically require the addition of synchronization interfaces between each pair of communicating modules.

Synchronization interfaces bring a new set of design concerns, including metastability-free operation and keeping latency and throughput figures at acceptable levels when traversing several synchronization points. A good approach is to reduce as much as possible the number of synchronization points, to achieve better data transfer rates and improve overall robustness. One way to reduce this number in a complex GALS SoC is to employ fully asynchronous communication mechanisms.

Communication in current and future SoCs relies on the use of Networks on Chip (NoCs) [3]. Using a fully asynchronous NoC as communication architecture for a SoC composed by synchronous processing elements (PEs), the number of synchronizations involved in a single point to point data transfer is reduced to two: one at the sender-NoC interface and another at the NoC-receiver interface. This paper describes the

design and implementation of an asynchronous NoC router that can give support to implement fully asynchronous NoCs.
The rest of this paper is divided into five Sections. Section II describes related work and positions the new proposition with regard to it. Section III describes the architecture of the Hermes-A router, while Section IV explores the characteristics of the router to PE interface. Section V discusses the ASIC implementation of Hermes-A and Section VI presents conclusions and directions for further work.

2. Related Work

During this decade there has been a small, yet steady movement towards research and implementation of fully asynchronous routers and corresponding NoCs. An encompassing review of the state of the art revealed ten relevant propositions of fully asynchronous interconnect architectures. Table 1 summarizes the main features for each of these, with the last row of the Table presenting the features for the proposed Hermes-A router and NoC.
Table 1 is organized by the date of the first proposition for each interconnect architecture, in a temporal line, although in some cases it cites later papers, where updated data about the NoC is present.
Chain and RasP belong to a first generation of asynchronous interconnect frameworks, based on the careful design of point-to-point links using repeaters, pipelining and wire length control. To support implementation, both offer a set of asynchronous components (the so-called routers, arbiters and multiplexers) that permit sharing the point-to-point links from multiple sources to one destination. Nexus is a very efficient industrial implementation of an asynchronous (16x16) crossbar. Strictly speaking, none of these three architectures really agree with the most accepted definition of NoCs as a network of multi-port routers and wires organized in a topology that forwards packets of information among processing elements. Accordingly, all three should display scalability problems as the number of PEs grow without bounds, what is expected for future technologies.
Another group of works include the propositions of Quartana et al. and the asynchronous version of the Proteo NoC. These are experiments in prototyping asynchronous NoCs in FPGAs, with the corresponding lack of performance and prohibitive cost in area. Implementations of asynchronous devices in FPGAs more efficient than those cited in these works exist, as described in [14]. These rely on use of FPGA layout and timing control tools to create asynchronous devices as FPGA hard macros that are compact and respect tight timing constraints. However, so far these have not been used for NoCs.
The remaining five NoCs/routers in Table 1 (QoS, MANGO, asynchronous QNoC, ANoC, ASPIN) and Hermes-A propose ASIC implementations of routers and links for 2D mesh topologies, although in some cases there is mention to adequacy to support other topologies as well. This is not the case for ASPIN, because of the chosen router organization. In this NoC, the router ports are distributed around the periphery of the PE, making inter router links small compared to intra router links. This facilitates connection of PEs by abutment, but prevents easy use of topologies

other than 2D mesh. Even a similar 2D torus would be problematic to build in this case.

Table 1 – A comparison of fully asynchronous interconnection networks and/or routers for GALS SoCs. Legend: **A2S, S2A** – Async. to Sync./Sync. to Async., **As.** -Asynchronous, **BE** – Best Effort service, **DI** - delay insensitive, **GS** – guaranteed service, **Irreg/Reg**- Irregular/Regular, **N.A.** - Information Not Available, **OCP** – Open Core Protocol, **VC** – virtual channel.

Characteristics → NoC	**Topology**	**Routing / Flow Control**	**Network Interface**	**Asynchronous Style**	**Links and encoding**	**Implementation**
Chain [4]	Framework / point-to-point (Irreg/Reg)	Source / EOP	Ad hoc	QDI / pipelined	Point-to-point 1-of-4 DI / 8-bit flits	180nm, 1Gbits/s per link, ASIC
QoS [5]	2D Mesh 4 3GS/1BE VCs	XY / wormhole / credit-based	N.A.	QDI	1-of-4 DI / 8-bit flits	Simulation only
Nexus [6]	Single 16x16 Crossbar	Source / BOP-EOP	A2S, S2A	QDI / 1-clock converters	1-of-4 DI / 36-bit phits	130nm, 780Gbits/s, ASIC
MANGO [7]	2D Mesh (Irreg/Reg) 4GS/1BE VCs	Source	A2S, S2A, OCP	4-phase bundled-data	Dual-rail, 2-ph. DI / 33-bit flits	130nm, 650Mflits/s, ASIC
As. QNoC [8]	2D Mesh (Irreg/Reg) 8VCs	Source / wormhole / credit-based with preemption	N.A.	4-phase bundled-data	10-bit flits	180 nm, 200Mflits/s, ASIC
Quartana et al. [9]	Crossbar or Octagon	N.A.	Self-timed FIFOs	QDI	N.A.	FPGA, 56 Mflits/s
ANoC [10]	2D Mesh (Irreg/Reg) / 2 VCs	Source / odd-even / wormhole	A2S, S2A FIFOs	QDI	34-bit flits	65nm, 550Mflits/s, ASIC
As. Proteo [11]	Bidirectional Ring	Oblivious	OCP	QDI / 4-phase dual-rail	32-bit flits	FPGA, 202 Kbits/s
RasP [12]	Framework / point-to-point (Irreg/Reg)	Source / bit serial	Ad hoc	Dual-rail	Point-to-point pipelined serial links	180nm, 700Mbits/s Simulation
ASPIN [13]	2D Mesh (Reg)	Distributed XY / wormhole / EOP	A2S, S2A FIFOs	Bundled-data	Dual-rail, 4-ph., 34-bit flits	90nm, 714Mflits/s
Hermes-A	2D Mesh (Reg)	Distributed XY / wormhole / BOP-EOP	Dual-Rail SCAFFI [14]	Dual-rail / bundled data	Dual-Rail	180nm, 727Mbits/s, ASIC

Four of the NoCs (QoS, MANGO, asynchronous QNoC, ANoC) claim support to quality of service through the use of virtual channels and/or special circuits (GS routers). ANoC is the most developed of the proposals and presents the best overall performance. It has been successfully used to build at least two complete integrated circuits [15]. However, most of the characterization for ANoC (and for other asynchronous NoCs) derives from a detailed knowledge of the application in sight. If the application has unpredictable dynamic behavior, it is fundamental to employ a more flexible approach to topology choice, routing and incorporating the capacity to take decisions based on dynamic information of the network. These are some reasons behind the proposal of Hermes-A, described in the next Sections.

3. The Hermes-A Router Architecture

Unlike most other asynchronous routers, Hermes-A employs a distributed routing scheme, where the router itself decides which path incoming packets will follow. This enables the use of adaptive routing algorithms and, more importantly, the router may employ these algorithms to solve network congestion problems in real time. Another characteristic of Hermes-A is that it uses an independent arbitration at each router port. The reason for this design choice is to allow that dynamic voltage level schemes be used to assign distinct voltage levels to distinct paths along a NoC. Such a fine grained voltage level resolution can be quite useful to fulfill important power-performance constraints so frequent in SoCs. Distributed routing and scheduling are characteristics shared by Hermes-A and ASPIN. Differences between these NoCs are on the lumped router design for Hermes-A, which facilitates the use of the router in topologies other than 2D meshes and the concerns for designing the router to support multiple voltage levels and adaptive routing algorithms.

A traditional 2D mesh topology NoC with wormhole packet switching is the test environment used to validate the Hermes-A router. Each router in the experimented setup comprises up to five ports: East, West, North, South and Local. As usual in direct NoCs, the Local port is responsible for the communication between the NoC and its local PE. All experiments described herein assume the use of 8-bit flits. The packet format is extremely simple: the first flit contains the XY address of the destination router and the subsequent flits contain the packet payload. Two sideband signals control the transfer of packets and support arbitrary-size packets: begin of packet (BOP), activated with the first flit of a packet, and end of packet (EOP), activated with the last flit. All intermediate flits display BOP=EOP=0.

Most of the router architecture employs a delay insensitive, 4-phase, dual-rail encoding. Note that each input port interface consists of 21 wires: 16 wires carry the 8-bit dual-rail flit value (DR-Data), four wires contain the dual-rail BOP and EOP information and the last is the single rail acknowledge signal. The router detects data availability when every pair of wires that define each bit value in the DR-Data signal is distinct from “00”. Thus, the all 0’s value in DR-Data is the spacer for the DI code.

A. Input Port

Figure 1 depicts the Hermes-A input port structure as a simplified asynchronous data-flow diagram [16]. There are three alternative paths in this module, one used for the first flit (1), one for intermediate flits (2) and one for the last flit (3). In Figure 1 two wires represent each bit. Thus, a 10-bit path is in fact a 20-wire bus.

When BOP is signaled at the input port, the first demux selects the path that feeds the module responsible for computing the path to use. This module receives ten information bits that are forwarded (8 data bits plus EOP and BOP), plus four destination bits using dual-rail one-hot encoding. Note that just the bit associated to the selected path is enabled in this 4-bit code. Since the routing decision must be kept for all flits in a packet, a loop was added to register the decision. The loop appears in Figure 1 as a chain of three asynchronous registers (4) in order to enable the data flow inside the 4-phase dual-rail loop. Each two successive asynchronous stages communicate using an individual handshake operation [16]. Thus, in this kind of circuit it is not possible that three successive stages exchange two data simultaneously. Exactly three stages are the minimum necessary to propagate information circularly. Less than three stages incur in deadlock situation. This can be better understood remembering that between every two valid data there is always a spacer, and that before propagating a spacer the first data must be copied to the next stage.

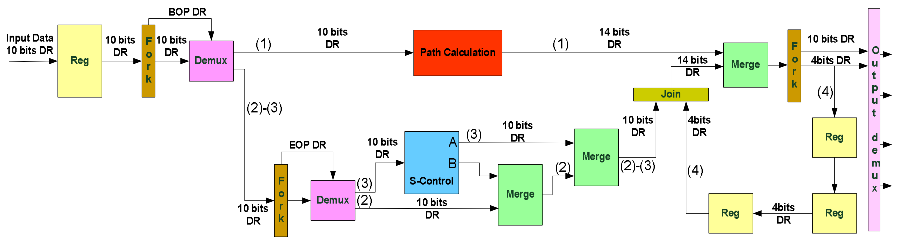

Figure 1: Hermes-A router input port architecture. All paths employ dual-rail encoding.

After computing the output port where to send the incoming flit, the rightmost module in Figure 1 (Output demux) sends the flit, based on the 4-bit routing information.

Subsequent flits in a packet go through the lower output of the leftmost demux and are input to a second demux after the fork element. This demux looks for the EOP bit before choosing the right direction for each flit. If there is no EOP indication the flit follows path (2) to the first merge component. Otherwise, the S-Control module is used. The next Sections cover the behavior of the Path Calculation and S-Control modules.

a) Path Calculation

The basic route computation architecture is depicted in Figure 2. In direct 2D topologies like 2D mesh or 2D torus, each router is defined with two values, its X and Y coordinates. The first flit of a packet carries the destination X address in the four less significant bits and the destination Y address in the four most significant bits. When a flit is accompanied by an active BOP signal it feeds the Path Calculation module. This flit arrives at the input of a completion detector (CD). Detection of a valid dual rail data token causes the propagation of the destination X and Y coordinates to two subtraction circuits. The outputs of these circuits will determine the path the packet must follow.

If both subtractions result in 0, then the packet reached the target router and it proceeds to the Local port. For the XY routing algorithm, if the X axis subtraction is different from zero, the packet will follow either to East or West, depending only on the sign of the result (positive and negative, respectively). If the X subtraction result is 0 but the Y subtraction is not, the packet may follow to North or South, depending again only on the signal of the result (positive and negative, respectively). The Routing Logic module is just a purely combinational logic that produces the resulting one-hot dual-rail 4-bit packet destination code. It points the output port to use.

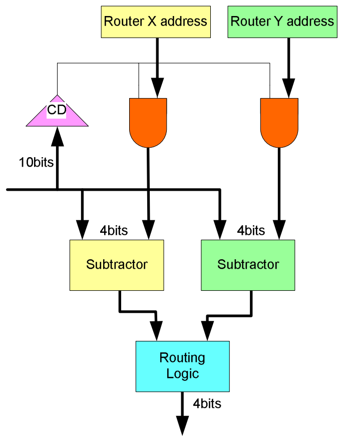

Figure 2: Hermes-A Path Calculation circuit.

b) S-Control

When the last flit of a packet is received (EOP=1), it is directed to the S-Control module (see Figure 1). The S-Control protocol description appears in Figure 3.

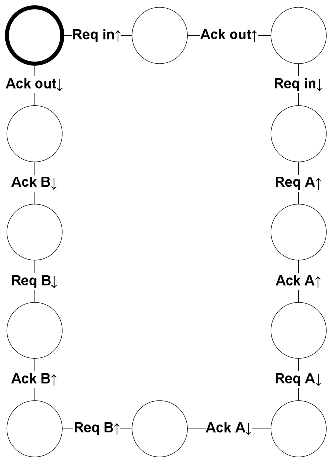

Figure 3: State machine for the S-control module.

The function of this module is to send the last flit through the output marked A in Figure 1, and then send a kill token in the output marked B to indicate the end of a packet transmission. This has as effect to de-allocate the output currently reserved for this packet. To avoid defining a new dual-rail signal, the unused code BOP=EOP=1 is employed internally to the router to signal this situation. The circuits that interpret this code are two: the allocated output port and the one that controls the chain (4) of asynchronous registers (not explicit in Figure 1). The later, upon receiving the code, empty the chain using spacers. Remembering that asynchronous circuits rely on explicit local handshake between every pair of communicating modules, the S-Control only generates an acknowledge signal to the previous demux after receiving the acknowledge signals for both, A and B outputs. Completeness detectors produce all request signals. The Petrify tool was used to synthesize the equations that implement a speed-independent controller operating as the state machine in Figure 3.

B. Output Port

In the Hermes-A router each output port receives four data flows. For instance, Figure 4 shows the Local output port structure that receives data from input ports North, South, East and West.

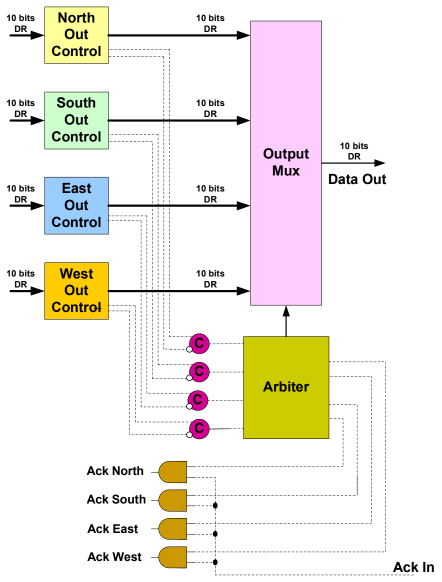

Figure 4: Local output port structure. Dashed lines represent actual wires. Solid lines represent dual-rail encoded lines.

An arbiter circuit controls the behavior of each output port. This arbiter achieves fairness with a structure of six 2-input, 2-output arbiters connected in a shuffle-exchange topology. Each atomic arbiter decides which request to serve from between two input requests, using a first-come-first-served strategy. This allows the processing of up to four simultaneous input port requests. The bit used to produce the request to the output port is produced by the logic that computes routing on the input port. Since this bit is a dual-rail representation, conversion to single-rail is necessary, since arbiters are the only single-rail module in the output port. A 2-input C-element with one negated input executes the conversion. Figure 5 details the structure of each output control circuit of an output port.

This module receives data directly from some input port. Its role is to generate requests for the output port arbiter or to undo the internal connection between input and output ports after transmitting the last packet flit and receiving the kill token.

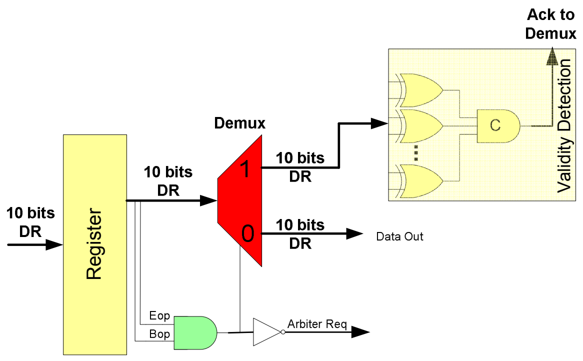

Figure 5: Output control structure. All paths employ dual-rail encoding.

4. Network Interface

The synchronization mechanism is one of the crucial components of a GALS system. Traditional synchronizers like the series-connected flip-flops do not guarantee elimination of metastability, and since synchronization latency is usually large in such synchronizers, these components often impose low throughput to the communication architecture. To overcome these limitations, this work chooses to employ clock stretching techniques, which do eliminate the risk of metastability. Also, this kind of synchronization can support higher throughput than traditional synchronizers.
The synchronization mechanism adopted here is based on the SCAFFI [14] asynchronous interface. SCAFFI is an asynchronous interface based on clock stretching that supports dual rail communication schemes. The network interface between Local Ports and PEs appears in Figure 6. More details on this interface are available in reference [14].

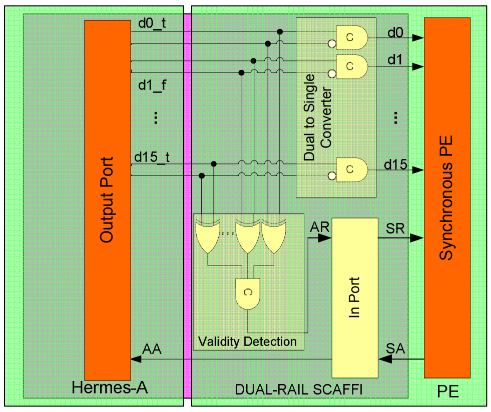

Figure 6: SCAFFI network interface between a Hermes-A router and a synchronous PE. The interface employs clock stretching techniques to avoid metastability. The stretcher circuits are not represented in the picture.

5. ASIC Implementation

Since traditional design kits do not usually contain asynchronous components, the Hermes-A ASIC implementation started with the implementation of an asynchronous digital cell library. The library includes several versions of C-elements, metastability filters and control circuits, like sequencers. The first version of the asynchronous library uses the XFab 180nm design rules and includes liberty timing files (.lib), abstract views (.lef) and Verilog models using UDP primitives to enable timing annotated simulations.
The asynchronous library is the base to develop a set of data flow elements (fork, join, merge, mux, demux, half-buffer registers, validity detectors, etc.).
During the asynchronous router synthesis it is important to guarantee that the (synchronous) synthesis tool do not change asynchronous components. For instance, in the Cadence RTL Compiler synthesis tool it is possible to ensure that this will not happen by using the PRESERVE property, which can be assigned to each module instance. This property instructs the tool not to touch the cell instance characteristics.
The results presented in the Table 2 refer to the XFab 180nm ASIC implementation of the Hermes-A router. The operating conditions are 25°C, 1.8 Volts. Also, the library build employs typical transistor models. Power results were obtained when all router input and output ports are operating at their highest rate of 727 Mbits/s on each router link. The throughput presented in Table 2 is for a single link operation. The router can sustain, in the best possible case, operation at this performance level on all of its five ports, totalizing approximately 3.6 Gbits/s of maximum throughput for the whole router.

Table 2 – ASIC Implementation results for a 180nm XFab technology.

Throughput (Mbits/s)	Area (mm^2) Cell – Total Area	Total Power (mW)
727	0.21 – 0.33	11.14

6. Conclusions and Future Work

The Hermes-A router demonstrates that asynchronous circuits are useful as a communication architecture for a high performance complex GALS SoCs. Ongoing work proceeds in several directions, including: (1) providing support to adaptive routing algorithms in Hermes-A; (2) enabling Hermes-A to work with multiple supply voltages and power shutoff features, in order to reduce the power utilization mainly in idle ports; (3) implementing complete NoC topologies and applications for testing router operation, such as 2D meshes and 2D tori. It is important to note that in the case of a 2D torus, the routing module has to be modified, since a pure XY routing algorithm is not deadlock-free for this network topology.

Acknowledgements

The Authors would like to acknowledge the support of the CNPq through research grants 551473/2010-0, 309255/2008-2, and 301599/2009-2. Also, they would like to acknowledge the National Science and Technology Institute on Embedded Critical Systems (INCT-SEC) for the support to this reseach.

References

[1] R. Ho, K. Mai, and M. Horowitz, “The future of wires,” *Proceedings of the IEEE*, 89(4), pp. 490-504, Apr 2001.
[2] D. Chapiro, “Globally-Asynchronous Locally Synchronous Systems,” PhD th., Stanford University, Oct 1984, 134 p.
[3] R. Marculescu, U. Ogras, L.-S. Peh, N. Jerger, and Y. Hoskote, "Outstanding Research Problems in NoC Design: System, Microarchitecture, and Circuit Perspectives," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 28(1), pp. 3-21, Jan 2009.
[4] J. Bainbridge and S. Furber, “Chain: A Delay-Insensitive Chip Area Interconnect,” *IEEE Micro*, 22(5), pp. 16-23, Sep-Oct, 2002.
[5] T. Felicijan and S. Furber, “An Asynchronous On-Chip Router with Quality-of-Service (QoS) Support,” In: *17th IEEE Int. SoC Conf. (SOCC’04)*, pp. 274-277, 2004.
[6] A. Lines, “Asynchronous Interconnect for Synchronous SoC Design,” IEEE Micro, 24(1), pp. 32-41, Jan-Feb 2004.
[7] T. Bjerregaard, M. Stensgaard, and J. Sparsø, “A Scalable, Timing-Safe, Network-on-Chip Architecture with an Integrated Clock Distribution Method,” In: *Design, Automation, and Test Europe (DATE’07)*, pp. 1-6, Apr 2007.
[8] R. Dobkin, R. Ginosar, and A. Kolodny, "QNoC asynchronous router," *Integration the VLSI Journal*, 42(2), pp. 103-115, Feb 2009.
[9] J. Quartana, S. Renane, A. Baixas, L. Fesquet, and M. Renaudin, “GALS systems prototyping using multiclock FPGAs and asynchronous network-on-chips”. In: *Int. Conf. on Field Programmable Logic and Applications (FPL’05)*, pp. 299-304, 2005.
[10] E. Beigné, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, “An Asynchronous NoC Architecture Providing Low Latency Service and its Multi-level Design Framework,” In: *IEEE Int. Symp. on Asynchronous Circuits and Systems (ASYNC’05)*, pp. 54-63, 2005.
[11] X. Wang, T. Ahonen, and J. Nurmi, “Prototyping a Globally Asynchronous Locally Synchronous Network-On-Chip on a Conventional FPGA Device Using Synchronous Design Tools,” In: *Int. Conf. on Field Programmable Logic and Applications (FPL'06)*, pp. 657-662, 2006.
[12] S. Hollis, S. Moore, “RasP: An Area-efficient, On-chip Network,” In: *Int. Conf. on Computer Design (ICCD'06)*, pp. 63-69, 2006.
[13] A. Sheibanyrad, A. Greiner, and I. Miro-Panades, “Multisynchronous and Fully Asynchronous NoCs for GALS Architectures,” *IEEE Design and Test of Computers*, 25(6), pp. 572-580, Nov-Dec 2008.
[14] J. Pontes, R. Soares, E. Carvalho, F. Moraes, and N. Calazans. "SCAFFI: An intrachip FPGA asynchronous interface based on hard macros," In: *Int. Conf. on Computer Design (ICCD'07)*, pp. 541-546, 2007.
[15] Y. Thonnart, P. Vivet, and F. Clermidy, “A Fully Asynchronous Low-Power Framework for GALS NoC Integration,” In: *Design, Automation, and Test Europe (DATE’10)*, pp. 33-38, 2010.
[16] J. Sparsø and S. Furber, “Principles of Asynchronous Circuit Design – A Systems Perspective”. Kluwer Academic Publishers, Boston, 2001. 354p.