Accurate Assessment of Bundled-Data Asynchronous NoCs Enabled by a Predictable and Efficient Hierarchical Synthesis Flow

Gabriele Miorandi†, Marco Balboni†, Steven M. Nowick§, Davide Bertozzi†
§ Department of Computer Science, Columbia University (USA)
† Engineering Department, University of Ferrara (Italy)

***Abstract*—Asynchronous interconnect technology leveraging transition signaling bundled-data is gaining momentum as a promising solution for the chip-level connectivity of GALS (Globally Asynchronous Locally Synchronous) integrated systems. However, the scope of most previous bundled-data network-on-chip (NoC) validations is limited to NoC switches in isolation. Studies with a broader scope admittedly end up in unstable results because of the incompleteness or low-maturity of the synthesis flow for asynchronous NoCs. By investing in the development of a predictable and hierarchical composition tool flow of NoC switches, this paper aims at major depth and insight in the comparative assessment of a complete bundled-data NoC with a competitive synchronous counterpart, when targeting an ultra-low power technology library.**

I. Introduction

Despite its appealing properties, asynchronous communication is far from pervasive in real-life Globally Asynchronous Locally Synchronous (GALS) systems. One reason is the lack of mature electronic design automation support for asynchronous circuits. This is largely due to a disconnect in the timing models employed for clocked and asynchronous design.

Another reason is associated with the common choice for the asynchronous communication protocol. Quasi-delay insensitive (QDI) design is typically used, since it is more robust to process and environmental variations, but also results in more complex circuits and less energy-efficient data encodings over networks-on-chip (NoC) links. This has raised the interest in bundled-data (single-rail) encodings, which use N lines to represent N-bit of information and two additional handshake lines indicating data validity and acceptance. This encoding typically results in simpler circuits, but has a relative timing constraint between control (validity) and data lines. This requirement further stresses the role of the computer-aided design (CAD) tool flow for convergence.

Recently, the superior energy savings of asynchronous NoC switches leveraging transition signaling (i.e., 2-phase) bundled-data over their synchronous counterparts have been validated based on hardness of experimental evidence [8], [17]. In particular, for the first time an asynchronous bundled-data router has been compared to an industrial synchronous competitor already fabricated in several commercial products, using a realistic advanced 14nm FinFET library [3]. The comparison shows significant area, latency and power savings for the asynchronous router even though several synthesis steps are still performed manually.

This trend is thus encouraging researchers to more systematically address the CAD challenge, especially by integrating and adopting mainstream industrial CAD tools where appropriate. This work is driven by the following guidelines: (i) generality of the approach for different asynchronous design styles [15]; (ii) overcoming a hard-macro approach to asynchronous design to keep up with modern flexibility requirements [1], [5], [7], [9], [17], [19]; (iii) development of timing validation methodologies for timing-driven logic and physical synthesis [12], [14], [16], [20], [24]. However, CAD tool flow improvements are most of the time still validated on simple designs (e.g., linear pipelines) that cannot capture the intricacy of real-life ones. In the best case, more complex designs are addressed (e.g., a fast fourier transform engine), but they are handled as monolithic flat entities due to the lack of consistent hierarchial design flows.

When restricting the scope to asynchronous NoCs, the main implication of the poor maturity of current synthesis tool flows is that the assessment of transition signaling bundled-data NoCs is typically performed only at the level of individual switching elements. However, switch-level and network-level validations of an interconnect technology do not necessarily bring the same conclusions about its quality metrics. First, switches need to be synthesized from the ground up for their compositional use in a hierarchical design in order to obtain predictable performance. This is typically accomplished by oversizing the driving strength of their input and output gates, which may offset area and power benefits characterized on the switch in isolation. Second, switch cycle time may be degraded when cascading it with a link, especially for asynchronous designs, since switch performance is often characterized in isolation through assumptions on the handshaking delay of connected modules. In the presence of the actual parasitic effects, more aggressive retiming may be needed to preserve cycle time, which degrades latency. While this may not be an issue in high-performance technology libraries, when low-power libraries for embedded system design are considered the latency implications may be significant.

In very few cases, the scope of literature is extended to the entire network, but the poor control on hierarchical composition steps of the network leads to admittedly unstable results in some cases. For instance, although [19] significantly improves upon state-of-the-art for the wide scope and cross-layer analysis when comparing synchronous vs. asynchronous NoCs, it reports unclear performance drops as large as 30% between static timing analyis and functional simulation for a design point under test, and post-layout operating speeds well below 500 MHz (not entirely explained by the 130nm old technology node) in all cases. Without carefully addressing and tackling the details of both circuit design and use of CAD tools, it is not possible to provide the necessary knowledge and insight that is needed to assess asynchronous interconnects of industrial scale and quality.

The goal of this paper is to bridge this gap in terms of evaluation and insight. It aims at fundamentally increasing the level of accuracy of network-level assessment frameworks for bundled-data NoCs by synergistically optimizing the switch architecture, the vertical and the horizontal synthesis methodologies for predictable and high-performance hierarchical NoC design. In more detail, the paper pursues a twofold objective:

(a) It aims at extending a cross-layer synthesis tool flow for bundled-data NoC switches, based on mainstream industrial tools, for a complete bottom-up hierarchical NoC design. The methodology is fine-tuned to avoid switch overdesign, and to preserve performance and energy efficiency when composing

NoC switches together into the topology as a whole.
(b) Leveraging this hierarchical synthesis methodology, the goal is to accurately contrast a 2-phase bundled-data NoC with its synchronous counterpart with similar architecture and synthesis flow, whenever possible, and with layout awareness. The synchronous NoC is itself assembled in a hierarchical way, thus capturing the different criticalities between hierarchical synchronous and asynchronous NoC design. As a result, for the first time this paper assesses bundled-data NoC performance and power on a 40nm low-power technology in relative terms with respect to a carefully optimized synchronous counterpart. To our knowledge, this paper sets a unique new advance in the apple-to-apple comparison between synchronous and bundled-data asynchronous NoCs.

The paper confirms that bundled-data NoC technology is more area-efficient than the synchronous one (-10% for a 4x4 2D mesh topology), unlike QDI solutions for which the opposite holds (up to 3x in [11]). For reasonable packet sizes (e.g., 20 32-bit flits), our analysis demonstrates a much better zero-load latency (-36%) and network saturation throughput (+33%). In contrast, latency becomes comparable with overly short packets (e.g., 3 flits) because of the asynchronous control overhead, which also leads to a steeper latency degradation after network saturation. Finally, asynchronous interconnects turn out to be a power efficient technology: 15% less power with short packets. However, the aggressive pipelining of the asynchronous switch architecture to make it robust for hierarchical composition into the whole topology (an effect that could not be seen in switch-level analyses) causes a power break-even point for long packets: the asynchronous NoC consumes up to 40x less power at low injection rates, and roughly 7% more power at the saturation point of the synchronous variant.

This paper is structured as follows. After reviewing previous work (Section II), switch architectures under test are described in Section III. Section IV presents the synthesis tool flow, while sections V and VI report link and network synthesis results, respectively. Finally, conclusions are drawn in Section VII.

II. Related Work

Several NoC designs have been proposed in the literature, using delay-insensitive communication [2], [6], [11]. Their validation shows that their timing robustness comes at a significant price with respect to synchronous NoCs, especially in terms of area and energy-per-bit.

There is a growing consensus in the field moving towards bundled-data encoding schemes. The architecture in [7] uses a 4-phase protocol in the switch, potentially resulting in simple circuits, and a 2-phase protocol on the link, which requires half the time-of-flight wire delays as 4-phase protocols. More recently, researchers have tried to extend the 2-phase protocol to the switch as well [8], [17], while revolving around a fast Mousetrap controller for the latter [18]. The design trade-offs spanned by bundled-data are not just design style-specific, but depend to a large extent on the specific choices for the design at hand. There is in any case agreement on the superior energy efficiency of bundled-data NoCs. This conclusion is definitely supported by the validation of bundled-data technology in an industrial environment on an advanced 14nm FinFET library [3]. Other works move away from use of the Mousetrap controllers while retaining 2-phase bundled-data handshaking, like the time-division multiplexed switch in [23], which uses Click elements.

Bundled-data NoC design requires the explicit management of absolute as well as relative timing constraints, which is hard to achieve with conventional EDA tools that come with a built-in synchronous mind-set. This section reviews previous work on the synthesis of bundled-data circuits.

Many proposed synthesis flows target the validation of relative timing constraints during both logic and/or physical synthesis [12], [14], [16], [20], [24]. However, they are often quite narrow in scope and validated on relatively simple designs, hence failing to capture the complexity of a real NoC.

More comprehensive methodologies have been proposed though. The flow in [15] consists of fully characterizing the asynchronous handshake clocking circuits as design templates that replace the clock tree in a traditional clock design. However, less emphasis is given to the physical design challenges, and validation is performed again on simple designs. A synthesis flow for domain-specific NoCs is proposed in [7], which optimizes the network topology and routers' physical placement on the floorplan based on the communication requirements. The flow gains visibility up to the floorplanning layer, and is validated by means of power and area physical models, not by actual place & route on an industrial technology library. The flow in [17] starts from a low-level asynchronous RTL description and delivers asynchronous NoC router layouts. The limitation is that it does not cover hierarchical design, where switches are assembled into a top-level NoC topology. This extension is one of the main goals of this paper.

The challenges of asynchronous design automation typically limit accurate comparisons between bundled-data asynchronous NoCs and their synchronous counterparts to individual switching elements. When a network-wide scope is targeted, the missing gaps in current CAD flows make comparative results not entirely trustworthy. The work in [19] sets a promising path for thorough asynchronous NoC assessment, due to the wide range of explored implementation options, and to the network-level analysis. Nonetheless, the lack of tight control over physical design issues leads to the poor post-layout performance of the synchronous switches, and to the tight coupling between the cycle time of the link and that of the input buffer in the asynchronous switch. As a result, the paper is not able to provide a convincing take-away message.

The industrial work in [22] is an exception. It presents both advanced architecture (e.g., communication channel pre-allocation) and advanced implementation (22 nm tri-gate CMOS technology). However, many circuits are designed with a full-custom methodology, and there are very few details on good synthesis practice and on the tool flow in general.

This paper moves from the awareness that effective electronic design automation support is a pre-requisite for the concrete evaluation of new technologies. Therefore, it provides the first comprehensive, flexible and realistic full NoC study with 2-phase bundled-data, by describing a complete hierarchical synthesis tool flow on an advanced low-power technology.

III. Switch Architectures Under Test

All the switches compared in this paper use wormhole switching, dimension-order routing and have 32-bit flit width. All the output buffers have six slots, while minimum storage requirements at input buffers are different, as explained next.

A. Synchronous Switch

The synchronous switch design implements the consolidated xpipes-Lite architecture from [21]. Its flow control protocol (stall/go) poses a requirement of minimum 2 slots on retiming stages: one default slot and one backup slot in case of stall assertion by the downstream node in order not to drop the flit in flight from the upstream node. xpipes takes a single cycle to traverse the switch (input-to-output port), and an additional cycle to transmit the data through the link.

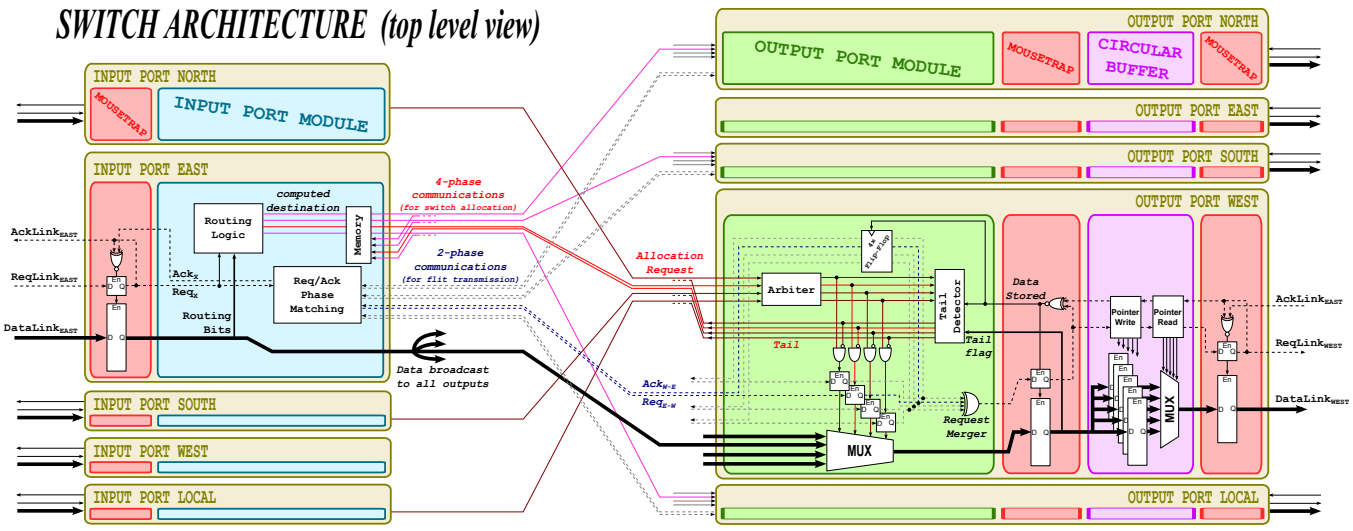

Fig. 1. Proposed asynchronous switch design.

B. Asynchronous Switch

Figure 1 shows the conceptual scheme of the proposed 5-ported asynchronous switch, inspired by [17]. This is the basic building block to design a 2D-mesh topology. It relies on Mousetrap stages, which build a low-overhead asynchronous pipeline providing high-throughput operation [18]. Starting from the input port interface on the left, the information is first stored into a Mousetrap stage. Then the INPUT PORT MODULE, which includes the routing logic, calculates the target output port and broadcasts data to output multiplexers. Interestingly, both 4-phase and 2-phase signaling are combined to connect input and output ports. A 4-phase 1-hot encoded signal generated by the routing logic is used to select the output port, while request and acknowledgement for flit-level transmission are propagated using a 2-phase protocol. Matching units are in charge of aligning the request phase at the input port with the corresponding phase at the connected output port. In the OUTPUT PORT MODULE, all the requesting input ports are arbitrated to exclusively allocate the port and drive the crossbar multiplexers. Requests are propagated from the input ports in a 2-phase fashion and blocked by a barrier of latches until the input port completes the allocation of that specific output port. The selected data is then stored into another Mousetrap stage. The data pass through a low-latency asynchronous CIRCULAR BUFFER stage before reaching an additional Mousetrap stage placed before the link interface.

With respect to the original architecture presented in [17], this paper aims at a novel high-performance buffering scheme to make switch performance more predictable when undergoing the top-level hierarchical design process. In fact, we intentionally inserted two retiming stages, one before and one after the circular FIFO. The former aims at sustaining throughput by acknowledging the input port as soon as data is safely stored in the associated Mousetrap; the latter aims at decoupling cycle time of the output buffer from that of the link, thus targeting minimum performance drops when switches and links are composed together. In fact, since link delay is involved twice in the link cycle time, it is not obvious at all to preserve throughput of isolated switches even for short links. Also, due to the internal circular FIFO control logic, the time required to generate an acknowledgement at the crossbar interface due to transition on the input request (or the time required to generate a request at the link interface due to a transition on the acknowledgement) increases linearly with the buffer size. For a 6-slot circular FIFO, it is about 300 ps in the target 40nm technology. Therefore, interposed Mousetrap stages are necessary to enable the use of moderately large buffer sizes.

IV. Design Flow

We synthesized both a synchronous and an asynchronous 4x4 2D-mesh NoC by means of mainstream CAD tools in the context of a fully automated design methodology. Logic and physical synthesis are performed with Synopsys Design Compiler and IC Compiler, respectively. All designs have been placed and routed targeting a low-power 40nm industrial technology library. To simplify the top-level network assembly, we followed a hierarchical bottom-up design approach for both synchronous and asynchronous NoCs. In a typical bottom-up hierarchical design flow, all the macro blockages are completely synthesized in separate processes. Without lack of generality, in our case we synthesized only a single 5-port switch macro with four directional ports (East, North, West and South) plus a local port on the North-West corner to interface it with the local core. During top-level NoC design, the same macro is replicated several times, and unused ports at the boundaries of the mesh are left unconnected. In this paper, the inter-switch link length was set to 1mm. This is compatible with the tile size of ultra-low power parallel accelerators for embedded computing, when scaled to the same technology node [13]. Other link lengths are left for future work.

A. Entry Level and NoC Compilation

A complete NoC design framework has been developed and used to generate a 4x4 2D-mesh NoC *starting from a graphical description* (standard *svg* format) of the NoC architecture. An in-house made NoC compiler automatically generates the Verilog entry level netlist for the NoC (top-level), while switch macros are synthesized separately and then imported during the top-level layout synthesis to exploit the bottom-up approach. Fig. 2-Left shows the entry level *svg* file for a 4x4 NoC.

Link entities are explicitly parameterizable to infer the desired number of link pipeline stages. As an option, the compiler can automatically insert the correct number of stages given the specification in the graphical user interface of the link length and of the characteristic link delay in ps/mm in the target technology.

Core entities are associated with floorplanning obstructions on top of which routing is forbidden, which will host the real layout of computation tiles. Core entities are also explicitly parameterizable, in order to specify the network coordinates for

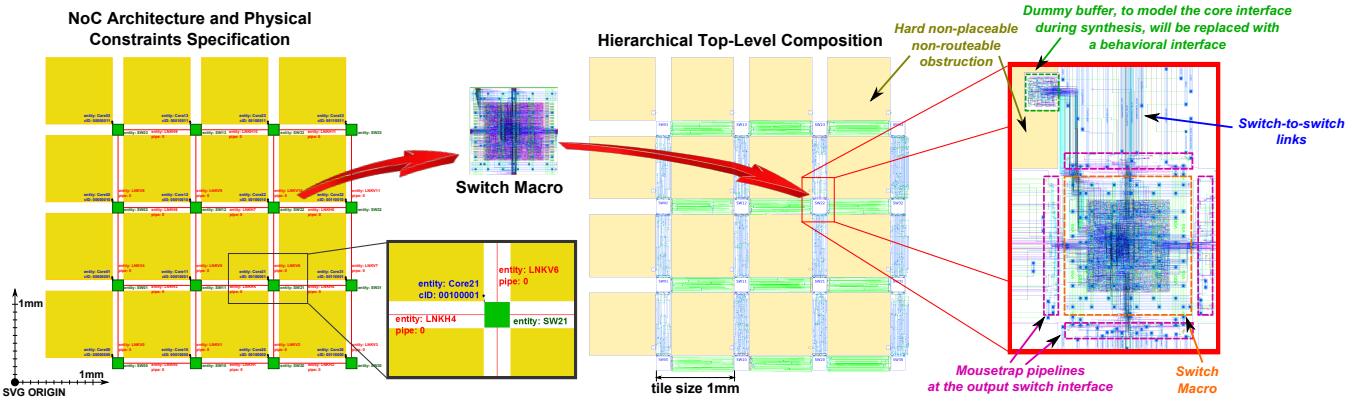

Fig. 2. Design framework, from graphical specification to Post-Place & Route.

algorithmic dimension-order routing, or to populate the routing tables at network interfaces in case source-based routing is used. Spatial coordinates related with the switch locations and the core blockages are automatically imported from the *svg* file and used to configure the Synopsys IC tool to automatically drive the floorplanning process.

The proposed framework is flexible, and can specify both regular topologies as well as irregular ones, in addition to selecting different routing mechanisms. In this paper, we will exercise only the 2D-mesh specification and compilation capabilities of the framework, combined with an algorithmic XY routing. We leave the unexplored configuration space for future work.

B. Synchronous Design Flow

In a bottom-up hierarchical synthesis flow for synchronous designs, switch macros are typically synthesized with overly tight constraints at their boundaries (i.e., slow signal transition times at input pins and high load capacitances projected for output pins). This is a conservative procedure which results in the oversizing of switch ports, so to potentially reduce the effort for timing and power closure during top-level hierarchical system design. In fact, in case of poor matching between projected pin loads and real ones, the maximum performance derived from characterizing NoC switches in isolation is largely wasted during top-level NoC design. At the same time, the poor control on signal slope can lead few cells at switch-link interfaces to dissipate a huge amount of static power due to the short-circuit current between power and ground rails. From another viewpoint, synthesizing the switches with overly tight constraints gives up the achievement of high-performance from the ground up (i.e., large cells or long buffer chains can better drive loads or restore signal integrity, but give rise to relevant propagation delays themselves).

Overall, final NoC layout quality metrics are closely related to the details of this hierarchical design step. Previous technical reports for NoC synthesis on similar low-power technologies [25] impose such conservative design constraints (e.g., output loads set to 100x the input capacitance of the biggest inverter in library, 0.4ns input transition time). However, those designs were targeting relatively slow operating speeds (below 500 MHz). This paper instead aims at maximizing performance on top of a low-power technology, hence the methodology was fine-tuned to reclaim every single MHz.

Thus, we performed an early design space exploration targeting different output loads. We forced the switch to drive a load capacitance equal to the input load of small-size (driving strength x5), average-size (x60), large-size (x200) buffers. We also tried to drive capacitances equal to 5/10/50/100x the input load of the biggest inverter in the library for the sake of comparison with common practice. When the constraint is overstated, the synthesis tool adds long buffer chains at switch output ports to virtually drive high output loads.

For each case, we hierarchically composed synthesized switches in a small setup including two switches and one connecting link. Link length for layout design was set to 1mm. We finally collected timing reports, and chose the solution that provides the maximum final performance, which was the average-size (x60) buffer.

In addition, we noticed more predictable performance (i.e., less deviation between switch in isolation and hierarchical 2-switch design) when fencing a layer of average-size buffers close to the switch output interface during top-level floorplanning.

On the other hand, signal integrity at the receiving switch input port is enforced by giving high priority to the *max_transition_time* constraint. This way the synthesis tool provides additional buffers along the link only if needed, and the switches preserve their own optimal performance.

Before running the top-level clock tree synthesis (CTS) we make the synthesis tool aware of the switch macro clock timing model by using the *set_clock_tree_exception* command. In particular, macro clock pins are modelled as *float_pins*, meaning that during the CTS the tool will consider the delay (specified by the user) from those clock pins to the leaf cells within the macros. Minimum and maximum rising time is first extracted using the *get_timing_path* command and then used as a parameter of the *set_clock_tree_exception* command. Once the clock tree is synthesized, it is marked with the *dont_touch* attribute, then placement and routing is performed on the rest of the design.

C. Asynchronous Design Flow

The proposed asynchronous design flow is detailed in Fig. 3, which extends the baseline flow in [17] for hierarchical design. It can be subdivided into a first-stage design flow for switch macros (from step #1 to #10), and a second-stage top-level design flow for the network as a whole (from step #11 to #15).

Our target technology library does not include asynchronous special cells (i.e. MullerC elements and mutexes), hence we used the standard-cell equivalent implementations for them. Besides this, other asynchronous circuits require a very low-level specification. In order to enable this, while still allowing a technology-independent specification, we use the generic GTECH Synopsys library. By using GTECH specification, the asynchronous designer has full control over the gate-level logic function. Such hybrid Verilog-GTECH RTL specification is fully synthesizable, and can be mapped on any technology library.

After reading the entry-level netlist (#1), logic manipulations are prevented by setting appropriate compile directives

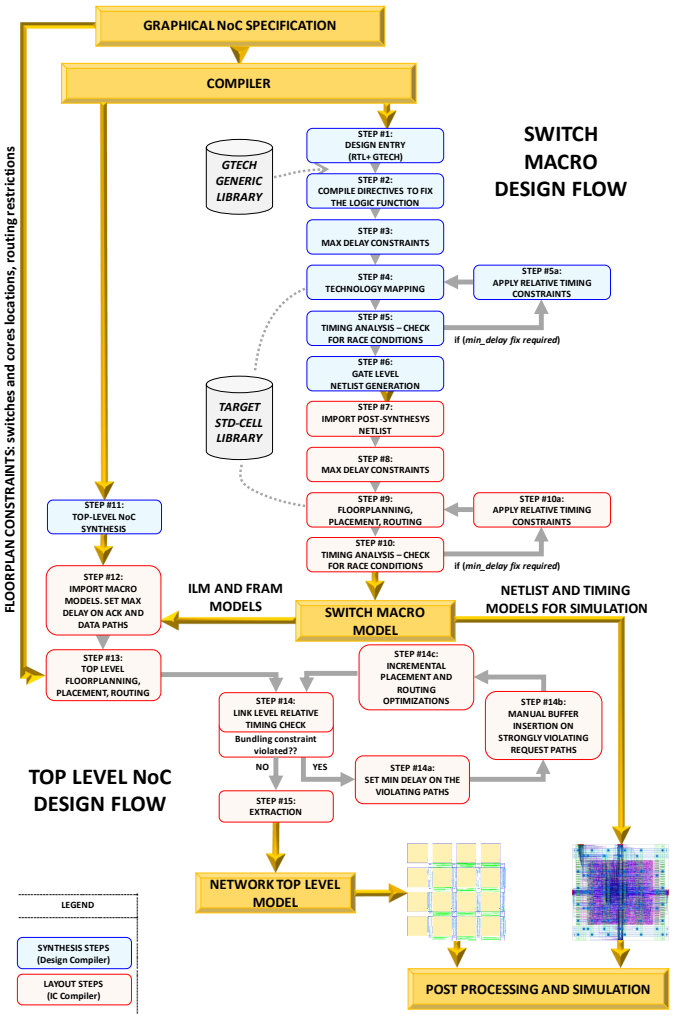

Fig. 3. Complete asynchronous design framework with detailed design flow procedures.

(#2): only gate sizing and buffer insertion options are enabled. The design is then constrained for max. performance (#3) by enforcing the *set_max_delay* command to all the timing paths in the design. After a first technology mapping run (#4), relative timing constraints are then checked (#5). The delay of the paths that have to be matched (data) can be extracted from the mapped netlist using the *get_timing_path* command. The delay is then assigned to the path to be delayed (request) using *set_min_delay* (#5a). To upper bound the request delay, a *max_delay* that is slightly larger than the *min_delay* is enforced on this path too. This procedure is iterated until all the bundling constraints are met with a safety margin of +10% with respect to the data-path delay[1] (#4 $\leftrightarrow$ #5). A similar procedure holds for the layout flow with IC compiler: timing constraints are iteratively checked until the margin between request and data lines is met (#9 $\leftrightarrow$ #10).

As for the synchronous design, we performed an early design space exploration on the asynchronous switch macro in order to make it robust for top-level composability. Interestingly, we achieved high performance predictability without instantiating a layer of buffers between switches and links, but by simply moving the last Mousetrap pipeline stage (the

[1]Validation of the bundled-data timing margin through on-chip variation analysis is left for future work.

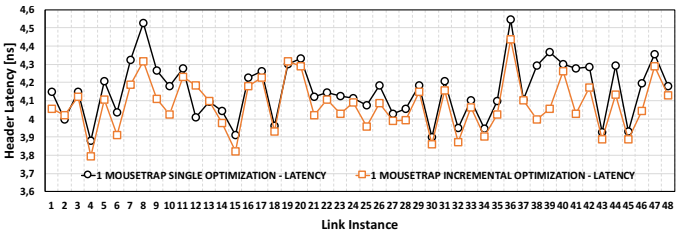

Fig. 4. Contrasting incremental optimization and one-shot optimization during top-level relative timing fix.

one after the output buffer in Fig.1) out of the switch macro, and exposing it as a top-level macroblock. At the same time, we set an output load on switch output pins equal to input capacitance of a latch input pin. During the top-level floorplanning procedure, this Mousetrap stage will be fenced close to the associated output port, as depicted in Fig.2-Right.

The final NoC layout flow starts with a top-level synthesis (#11), where only top-level connections between switches are specified, while switch macros are treated as black boxes. The synthesized top-level netlist and the committed switch macro models are imported and linked in the Synopsys IC Compiler environment (#12). *Max_delay* constraints are applied on data channels and acknowledgements only, in order to get the best performance. Once floorplanning, placement and routing are terminated (#13), we start the procedure to fix the relative timing constraints (#14). In most cases setting *min_delay* constraints (#14a) is not enough to achieve design convergence, even if we give high priority to the *min_delay* optimizations. When this is the case, we help timing closure by means of an engineering change order (ECO): we insert a buffer cell on the request net and we place it with the *place_eco_cell* command (#14b). This procedure is iterated through incremental placement and routing optimizations (#14c) until all the bundling constraints are met with a safety margin of +10%. A script automatically masters the ECO iterations and leads to timing closure.

In this methodology, we initially forced the place & route tool to resolve all relative timing constraints with a margin of 10% with respect to the datapath delay (one-shot optimization). We later found out that this procedure often ends up in exceedingly high relative timing margins, despite the upper bound we enforce on it, which penalize NoC performance. Therefore, we fine-tuned the methodology to gradually meet the target: at first we set a 0% margin, then 5%, and finally 10%. Every intermediate target is met by means of the iterative ECO procedure. Fig. 4 shows 1-hop header latencies measured on all the 24 bidirectional links of a 4x4 2D-mesh with tile size of 1mm. Clearly, incremental optimization improves header latency up to 6% with respect to one-shot optimization. Therefore, we used incremental optimization for the experimental results.

D. Post-Processing and Simulation

To connect the switch local port with the associated *Core* interface we placed a "dummy macro" in the position where the *Core* interface is expected to be. For this purpose we use a macro model of a simple circular buffer. As depicted in detail in Fig. 2-Right, this trick ensures a correct and consistent routing of the connections with the local *Core*, while in the rest of the *Core* area, placement and routing operations are prohibited. During the netlist post-processing, the entities of the dummy buffers are replaced with behavioural traffic generators.

E. Assumptions on Injection/Ejection Interfaces

Assuming local synchronous cores, the traffic generators should also take into account the clock domain crossing

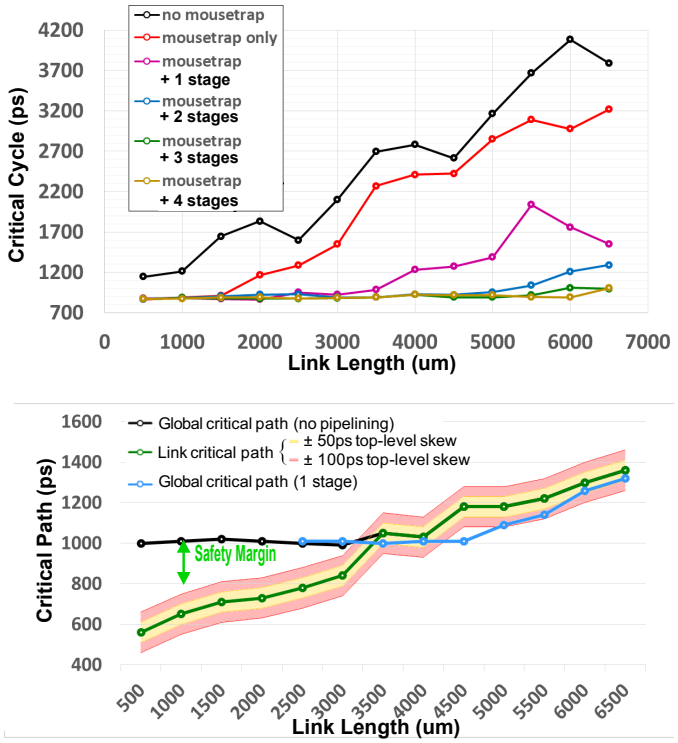

Fig. 5. Asynchronous (top) and synchronous (bottom) link performance for different link lengths and pipelining depths.

issue. In this work, we leverage ideal synchronizers at injection/ejection interfaces. As a result, this paper accounts for the lower bound on the latency difference that can be experienced by the synchronous vs. the asynchronous NoCs, so that we keep the focus on inherent network performance and not on the distinctive features of the clock domain crossing interfaces at hand.

At the injection interface, an infinite FIFO buffer is assumed, with 1GHz injector cores. When the network is faster than the core (which is the typical case), the ideal synchronization latency depends on the relative misalignment between the rising clock edges. Instead, when cores inject into an asynchronous NoC, then ideally the FIFO crossing latency is equal to just the req/ack cycle delay to transmit the flit to the downstream switch. Intuitively, synch-to-asynch converters do not have to perform the complementary asynch-to-synch conversion that a dual-clock FIFO would perform.

At the ejection interface, we simply assume that as soon as a flit is written onto the receiver FIFO (with a synchronous write or with an asynchronous handshake), the flit is ejected, and its timestamp taken for latency calculations. We do not model synchronous ejectors, since their ejection rate would artificially limit the performance of the NoCs under test. Hence, we are interested in extracting peak NoC performance.

V. Preparing for NoC Synthesis

A. Fine-Tuning Link Design

Before synthesizing the NoC as a whole, we had to decide the degree of link pipelining (if any) over the target 1-mm links. For this purpose, we instantiated two switches, and laid out the connecting link for different lengths.

1) Synchronous Design: Fig. 5-Bottom contrasts the global critical path of the mini-experimental setup against the link critical path. Starting from 3mm and beyond, the critical path lies on the link. At 1mm (target of this paper), the link timing path has a margin of about 300ps. However, such timing margin might be nullified or artificially extended by a negative or positive skew in the top-level CTS, respectively. Therefore, in the same figure, the effect of $\pm$ 50 ps and $\pm$ 100 ps (roughly 10% of the clock cycle) top-level clock skew are pointed out as well.

Instead of performing post-Place & Route optimizations to exploit this margin for area/power relaxations, we decided to maintain it and use it to relax the skew constraints on the top-level clock tree. This is an interesting degree of freedom to help convergence of the most critical synthesis step for a synchronous NoC, that is, its top-level clock tree. In practice, timing margins can be retrieved on NoC links, while at the same time relaxing global skew constraints. As a result, as the link length increases, the designer must trade between timing robustness and latency overhead caused by the insertion of an additional pipeline stage on the link. Since the skew constraint becomes more and more critical as the die size increases, we conclude that a pipeline stage is required for link lengths greater than 2.5mm with the target technology in order to restore a good margin and simplify the CTS process. As depicted in Fig. 5-Bottom, the additional pipeline stage is capable to link two switches up to 4.5mm without incurring in throughput losses. As far as this paper is concerned, no pipeline stage is needed for 1-mm links.

2) Asynchronous Design: We run several link layouts from 0.5mm to 6.5mm with a different number of Mousetrap stages. Results are illustrated in Fig. 5-Top. It turns out that a pipeline stage is required each 1.5mm to avoid throughput losses. To further motivate the new pipeline organization proposed in this paper, we included an experiment called “no_Mousetrap” in the same figure. This design refers to a pair of switches with no “output” Mousetrap interposed between the switch circular FIFO and the link. As expected, the performance penalty is quite severe also for overly short links. This setup likely reflects the link design style in [19], where no link retiming is inferred, and explains the poor performance of the asynchronous NoC in that paper. As far as this paper is concerned, the simple decoupling Mousetrap between output buffer and link is enough for 1-mm links.

VI. Experimental Results

Three 4x4 2D-mesh NoCs were synthesized with 1-mm links: the synchronous one, the baseline asynchronous one without any Mousetrap stage on the link, and the proposed NoC with one decoupling Mousetrap between switch and link.

A. Design Closure

The synchronous NoC achieves timing closure at 1 GHz, which matches the performance of the switches in isolation: the critical path in fact stays inside the switch. Top-level CTS convergence was achieved with a skew constraint of 14%, which is compatible with an average link timing margin of 30%. For the asynchronous NoC, a layout quality metric is given by the homogeneity of link performance, since each link has its own ”virtual” clock speed. Cycle times on a link basis are illustrated in Fig.4, using incremental optimization of relative timing constraints. Considering all links, the deviation from the mean value is lower than 8%, thus proving that the presented flow does a good performance equalization job despite the implicit timing representation as a clock frequency is not available, and all timing constraints have to be explicitely managed. Finally, the asynchronous NoC saves roughly 10% area: 210855 μm^2 vs. 234141 μm^2, with a breakdown dominated by switches and different contributions of link area for the different design styles (see Table I).

Area Breakdown				
Technology	**Total Switches [%]**	**Top-Level [%]**	**CTS [%]**	**Total Area [μm^2]**
Async	92.8	7.2	–	210855
Sync	97.9	1.9	0.2	234142

TABLE I. NOC AREA ANALYSIS.

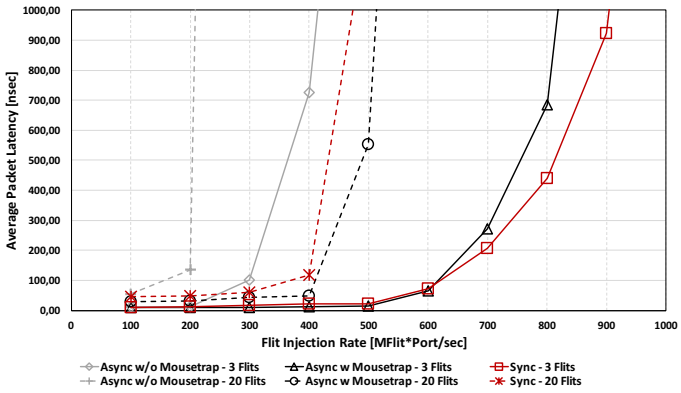

Fig. 6. Load curves for NoCs under test.

B. Performance Analysis

We analyzed each one of the designs under test under uniform traffic with long and short packets (without lack of generality, 20 flits and 3 flits, respectively). In Fig. 6, load curves are compared. The asynchronous design without any Mousetrap stage always reports a worst performance than the proposed asynchronous design.

Contrasting synchronous and proposed asynchronous NoCs, a minimum latency difference is observed for *short packets and low injection rates*. This is due to the fact that asynchronous performance is dominated by the header latency. The asynchronous header latency to traverse one switch is in general higher than the synchronous latency, given that asynchronous control logic is usually more complex to ensure glitch-free operation, and is further penalized by the safety margin that we impose on all the relative timing constraints (10% of the matching data path delay). Ultimately, the asynch. NoC requires about 1500ps to traverse the switch from input to output, and 1000ps to traverse the link. In contrast, the synchronous design takes two cycles at 1000ps each to cover the same distance. As explained in Sec. V, for 1mm designs the synchronous critical path lies on the switch, while the slack margin on the link is useful to simplify the constraints on the top-level clock tree. Another contribution to the asynchronous latency comes from the non-negligible propagation delay experienced by latches in ultra-low power technology libraries, where retiming is not as inexpensive as is generally believed.

Short packets also imply that NoC switches arbitrate quite frequently. Given that header processing is critical for the asynchronous NoC performance, the synchronous NoC outperforms the asynchronous counterpart when *short packets and high injection rates* are considered.

Body flit latency is dominated by the cycle time (in the sense that these flits are interleaved by an amount of time equal to the cycle time). However, in their cycle time there is no arbitration, which has been already performed by the head flit. Therefore, once an asynchronous header flit allocates the switch internal path through the arbiter, many tail flits can take advantage of that, and rapidly flow through the network at high data rates.

Given that asynchronous cycle time for body flits is much lower than the synchronous one (roughly 870ps), the longer the packet size the larger the asynchronous performance improvement. This is a typical asynchronous benefit which can not be exploited by synchronous designs, where the latency/cycle-time for header and payload flits are the same (aligned to the worst-case). This explains the better zero-load latency (-36%) and saturation bandwidth (+33%) of the asynchronous NoC for *long packets.*

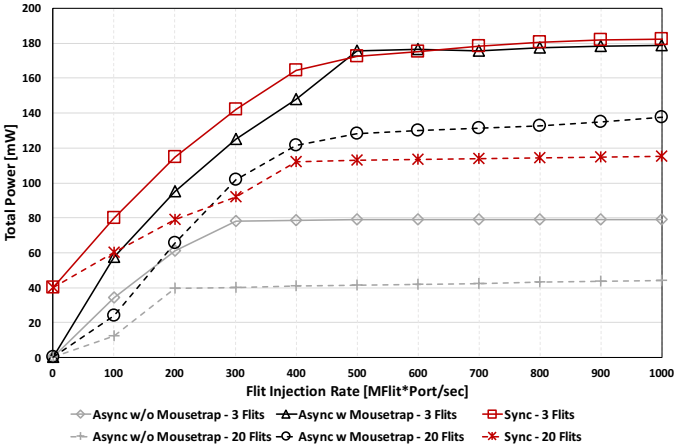

Fig. 7. NoCs power consumption.

C. Power Analysis

Fig. 7 shows results for power consumption analysis for different injection rates and different packet lengths. Power at low-injection rates is almost null for asynchronous designs, while the synchronous NoC consumes about 40mW despite clock-gating is applied. From the plot, the asynchronous NoC without Mousetrap stages is clearly a trade-off between lower performance and ultra-low power.

When considering the performance-optimized designs only, with *short packets* the synch. and the asynch. NoCs have the same power variation as a function of injection bandwidth, but the former consumes roughly 17% more power. The asynchronous power overhead mainly depends on the pipeline organization: the asynchronous NoC has 4 sequential elements per hop on the datapath (three Mousetraps and the circular buffer, see Fig.1), while the synchronous NoC has only two (input and output buffers). Nonetheless, power curves of the synch. and asynch. NoCs have the same slope since the synch. one is able to make only a partial use of clock gating.

When we consider *long packets and low injection rates*, we see that the asynchronous curves have the same slope as before. This means that for a given injection rate, with the exception of the arbitration overhead, sending few long packets or many short packets consumes the same power. On the other hand, the slope of the synchronous NoC is less steep than before, since many short packets trigger several switches in the network, while few long packets leave most of the network in idle state. In this latter case, the synchronous switch can better capitalize on the power benefits of clock gating. As a result, as the *injection rate increases*, there is a power break-even point at 65% of the saturation bandwidth of the synchronous NoC. The power overhead of the asynchronous solution is however limited to no more than 7% at the synchronous saturation point. The power gap at *higher injection rates* is then simply motivated by the fact that the asynchronous NoC saturates much later.

Finally, when comparing short and long packet transmissions, the latter have always lower power. This fact is mainly related with the lower arbitration frequency.

CPU USAGE IN HOURS	Incremental	Synch. Cumulative	Incremental	Asynch. Cumulative	Asynch. Slowdown
Switch Synthesis Time	0,03	0,03	0,09	0,09	2,49x
Switch Layout Time	0,35	0,38	0,69	0,78	2,03x
Top-level Floorplan	0,02	0,41	0,03	0,81	1,99x
Top-level CTS (Synch. only)	0,27	0,68	0,00	0,81	1,20x
Top-level Placement	0,21	0,89	0,39	1,20	1,35x
Top-level Routing	0,53	1,42	0,77	1,97	1,39x
Relative Timing Constraint Fix (Asynch. only)	0,00	1,42	12,39	14,36	10,10x
Final Extraction	0,09	1,51	0,08	14,44	9,57x

Fig. 8. CPU time required by synchronous and asynchronous synthesis flows.

D. Total Synthesis Time

Fig. 8 shows the CPU time required to synthesize the synchronous 4x4 NoC and the best performing asynchronous counterpart. All logic and physical synthesis runs were performed on an 8-core Intel Xeon E5520 (2.27GHz, 12GB RAM) server machine. The first two lines in Fig. 8 refer to the switch macro generation flow. This takes 0.38 CPU hours for the synchronous NoC, and 0.78 hours for the asynchronous one. Synchronous and asynchronous top-level layouts take 1.12 hours and 13.66 hours, respectively. For the asynchronous layout, the relative timing fix procedure is the most time-consuming operation. It takes more than 12 hours because a lot of timing extractions and optimizations are iteratively (although automatically) performed on all the req/data channels until the bundling constraints are satisfied.

VII. Conclusions

This paper provides a comparison between a 16-node 2-phase bundled-data NoC and its synchronous counterpart on top of a low-power 40nm technology library with major depth and insight, by building on a complete, predictable and efficient hierarchical synthesis flow for bundled-data NoCs.

From a performance viewpoint, we find that reasonably long packets are required to capitalize on the better body flit latency and cycle time of the asynchronous NoC. In fact, performance of short 3-flit packets depends mainly on the performance of their head flits, which pay the price of a slightly longer asynchronous control logic delay than in synchronous designs. Nonetheless, when packet sizes are custom-tailored to the realistic cache-line sizes of modern high-end embedded processors, packets will be at least larger than 8 flits even for 64-bit processors. In these conditions, an asynchronous NoC already delivers better overall performance.

From a power viewpoint, for low injection rates the asynchronous NoC is more than 40x more power-efficient than the synchronous counterpart, thus confirming previous switch-level evaluations. However, the power gap with the synchronous NoC tends to be erased as the injection rate increases. In fact, the pipelined asynchronous NoC is penalized by the larger number of sequential elements on the flit datapath. This overhead comes in part by the need to make switches more robust for predictable hierarchical composition. Nonetheless, at the saturation point of the synchronous NoC, the power overhead of the asynchronous NoC does not exceed 7%. Finally, the asynchronous NoC yields also 10% less total area, but takes roughly 10x more time to synthesize for a 4x4 2D-mesh topology.

Directions for future work include a faster convergence of relative timing constraints and more power-efficient solutions for performance predictability throughout the synthesis flow.

Acknowledgements

This work was partially supported by NSF Grant CCF-1527796.

References

[1] Y. Thonnart, E. Beigne and P. Vivet, *"A pseudo-synchronous implementation flow for WCHB QDI asynchronous circuits"*, IEEE Int. Symposium on Asynchronous Circuits and Systems, 2012, pp. 73-80.

[2] J. Bainbridge and S. B. Furber, *"CHAIN: a delay-insensitive chip area interconnect"*, IEEE Micro, v. 22, n. 5, 2002, pp. 16-23.

[3] W. Jiang, D. Bertozzi, G. Miorandi, S. M. Nowick, W. Burleson and G. Sadowski, *"An asynchronous NoC router in a 14nm FinFET library: comparison to an industrial synchronous counterpart"*, DATE 2017, pp.732-733.

[4] S. B. Furber et al., *"Overview of the SpiNNaker system architecture,"* IEEE Transactions on Computers, v. 62, n. 12, 2013, pp. 2454-2467.

[5] P. A. Beerel, G. D. Dimou and A. M. Lines, *"Proteus: an ASIC flow for GHz asynchronous designs,"* IEEE Design and Test of Computers, v. 28, n. 5, 2011, pp. 36-51.

[6] A. Lines, *"Asynchronous interconnect for synchronous SoC design,"* IEEE Micro, v. 24, n. 1, 2004, pp. 32-41.

[7] D. Gebhardt, J. You and K. S. Stevens, *"Design of an energy-efficient asynchronous NoC and its optimization tools for heterogeneous SoCs,"* IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, v. 30, n. 9, 2011, pp. 1387-1399.

[8] M. Gibiluka, M. T. Moreira, F. G. Moraes and N. L. V. Calazans, *"BAT-Hermes: a transition-signaling bundled-data NoC router"*, IEEE Latin American Symposium on Circuits and Systems, 2015, pp.1-4.

[9] L. A. Plana et al., *"SpiNNaker: design and implementation of a GALS multicore system-on-chip"*, ACM Journal on Emerging Technologies in Computing Systems, v. 7, n. 17, 2011, pp. 1-18.

[10] S. M. Nowick and M. Singh, *"Asynchronous design - Part 1: overview and recent advances"*, IEEE Design and Test, v. 32, n. 3, 2015, pp. 5-18.

[11] Y. Thonnart, P. Vivet and F. Clermidy, *"A fully-asynchronous low-power framework for GALS NoC integration"*, DATE 2010, pp. 33-38.

[12] W. Lee, T. Sharma and K. S. Stevens, *"Path based timing validation for timed asynchronous design"*, VLSI Design 2016, pp. 511-516.

[13] F. Conti, D. Rossi, A. Pullini, I. Loi and L. Benini, *"PULP: a ultra-low power parallel accelerator for energy-efficient and flexible embedded vision"*, Signal Processing Systems, v. 84, n. 3, 2016, pp. 339-354.

[14] T. Sharma, W. Lee and K. S. Stevens, *"Relative placement in timed asynchronous design"*, ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems 2016, pp.120-123.

[15] K. S. Stevens, Y. Xu and V. S. Vij, *"Qualifying relative timing constraints for asynchronous circuits"*, IEEE International Symposium on Asynchronous Circuits and Systems, 2016, pp. 91-98.

[16] J. V. Manoranjan and K. S. Stevens, *"Characterization of asynchronous templates for integration into clocked CAD flows"*, IEEE International Symposium on Asynchronous Circuits and Systems, 2009, pp. 151-161.

[17] A. Ghiribaldi, D. Bertozzi and S. M. Nowick, *"A transition-signaling bundled data NoC switch architecture for cost-effective GALS multicore systems"*, DATE 2013, pp. 332-337.

[18] M. Singh and S. M. Nowick, *"MOUSETRAP: high-speed transition-signaling asynchronous pipelines"*, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, v. 15, n.6, 2007, pp. 684-698.

[19] M. Imai, T. V. Chu, K. Kise and T. Yoneda, *"The synchronous vs. asynchronous NoC routers: an apple-to-apple comparison between synchronous and transition signaling asynchronous designs"*, IEEE/ACM International Symposium on Networks-on-Chip, 2016, pp. 1-8.

[20] M. Gibiluka, M. T. Moreira and N. L. V. Calazans, *"A bundled-data asynchronous circuit synthesis flow using a commercial EDA framework"*, Euromicro Conference on DSD, 2015, pp. 79-86.

[21] D. Bertozzi and L. Benini, *"Xpipes: a network-on-chip architecture for gigascale systems-on-chip"*, IEEE Circuits and Systems Magazine, v. 4, n. 2, 2004, pp. 18-31.

[22] G. Chen, et al., *"A 340 mV-to-0.9 V 20.2 Tb/s source-synchronous hybrid packet/circuit-switched 16x16 network-on-chip in 22 nm tri-gate CMOS"*, IEEE JSSC, v. 50, n. 1, 2015, pp. 59-67.

[23] I. Kotleas, D. Humphreys, R. B. Sorensen, E. Kasapaki, F. Brandner and J. Sparso, *"A loosely synchronizing asynchronous router for TDM-scheduled NoCS"*, IEEE/ACM International Symposium on Networks-on-Chip, 2014, pp. 151-158.

[24] C. T. Muller, E. Kasapaki, R. B. Sorensen and J. Sparso, *"Synthesis and layout of an asynchronous network-on-chip using standard EDA tools"*, Proceedings of the 32nd NORCHIP Conference, 2014, pp. 1-6.

[25] NaNoC Project, Deliverable 5.2, http://cordis.europa.eu/docs/projects/cnect/2/248972/080/deliverables/001-D52REVISED.pdf.