PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

FACULDADE DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CONTRIBUTIONS TO THE DESIGN AND PROTOTYPING OF GALS AND ASYNCHRONOUS SYSTEMS

MATHEUS TREVISAN MOREIRA

Submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science at the Pontifícia Universidade Católica do Rio Grande do Sul

ADVISOR: PROF. NEY LAERT VILAR CALAZANS

PORTO ALEGRE

2012

FICHA CATALOGRÁFICA

M838c Moreira, Matheus Trevisan

Contributions to the design and prototyping of GALS and asynchronous systems / Matheus Trevisan Moreira. – Porto Alegre, 2012.

79 f.

Diss. (Mestrado) – Fac. de Informática, PUCRS.

Orientador: Prof. Dr. Ney Laert Vilar Calazans.

1. Informática. 2. Circuitos Assíncronos. 3. Arquitetura de Redes. 4. Roteamento – Redes de Computadores. I. Calazans, Ney Laert Vilar. II. Título.

CDD 004.6

Ficha Catalográfica elaborada pelo

Setor de Tratamento da Informação da BC-PUCRS

Pontifícia Universidade Católica do Rio Grande do Sul
FACULDADE DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

TERMO DE APRESENTAÇÃO DE DISSERTAÇÃO DE MESTRADO

Dissertação intitulada *"Contributions to the Design and Prototyping of GALS and Asynchronous Systems"*, apresentada por Matheus Trevisan Moreira como parte dos requisitos para obtenção do grau de Mestre em Ciência da Computação, Sistemas Embarcados e Sistemas Digitais, aprovada em 13/04/2012 pela Comissão Examinadora:

[illegible]

Prof. Dr. Ney Laert Vilar Calazans – Orientador	PPGCC/PUCRS

[illegible]

Prof. Dr. Fernando Gehm Moraes –	PPGCC/PUCRS

[illegible]

Prof. Dr. Edson Ifarraguirre Moreno -	FACIN/PUCRS

Homologada em 08/06/2012, conforme Ata No. 12 pela Comissão Coordenadora.

[illegible]

Prof. Dr. Paulo Henrique Lemelle Fernandes
Coordenador.

PUCRS **Campus Central**
Av. Ipiranga, 6681 – P32– sala 507 – CEP: 90619-900
Fone: (51) 3320-3611 – Fax (51) 3320-3621
E-mail: ppgcc@pucrs.br
www.pucrs.br/facin/pos

ACKNOWLEDGMENTS

To my advisor, who became a friend over these years, Prof. Dr. Ney Calazans, thanks for all his support and guidance in the development of this work. Thanks for the confidence deposited in me and for all the useful discussions we had. Thanks for the opportunities given to me and for always being available when I needed.

Thanks to the members of GAPH and GSE, to my friends and the people who contributed somehow in the development of this work. Thanks to all my friends and colleagues, especially to Felipe de Magalhães, Jerônimo Corrêa and Bruno Oliveira for their friendship and the support in the development of this work.

Thanks also go to Professors Fernando Moraes, Edson Moreno, Alexandre Amory and César Marcon for all the useful and insightful discussions we had. Special thanks to my colleague and friend Julian Pontes for all the discussions about microelectronics and asynchronous circuits and for solving my CAD tools problems.

Finally thanks to my family and to my girlfriend for all the support. Thanks for your love and patience. Without you this work would not have been possible

ABSTRACT

As CMOS technology nodes scale down, new problems arise concerning the design of synchronous circuits and systems. This is due to tight constraints resulting from the use of a single signal to control a whole complex integrated circuit. Moreover, modern chips integrate whole systems that require a large amount of intellectual property cores, each with specific requirements and design constraints. In this context, asynchronous design techniques present appealing solutions to help designers achieving efficient systems, as each core can be independently implemented and then employ asynchronous communication at the system level. Different works available in literature demonstrate that asynchronous circuits are well suited for low power, high speed and robust applications. However, these circuits are very difficult to be implemented, due to the lack of design automation tools and basic components. In this way, experiments with asynchronous circuits are practically limited to full custom approaches. In order to help overcoming such limitations, the Author has been involved with asynchronous circuits design for five years. This work presents details of part of this research work, including the implementation of five non-synchronous network-on-chip routers, a standard cell library with over five hundred components for asynchronous circuits and a design flow proposed for such components.

RESUMO

Com o avanço de tecnologias CMOS, novos desafios surgem para o projeto de circuitos e sistemas síncronos. Isso se deve ao fato de que o uso de um único sinal para controlar um circuito integrado complexo resulta em restrições de projeto difíceis de serem atendidas. Além disso, chips atuais integram sistemas inteiros, que necessitam de uma grande quantidade de núcleos de propriedades intelectual, cada um com necessidades e restrições específicas. Neste cenário, técnicas assíncronas de projeto representam soluções interessantes para ajudar projetistas a obter sistemas eficientes, uma vez que cada núcleo de propriedade intelectual pode ser projetado de forma independente e então comunicar-se assincronamente, a nível de sistema. Diversos trabalhos disponíveis na literatura demonstram que circuitos assíncronos são adequados para implementações que necessitem baixo consumo de potência, alto desempenho ou alta robustez. Entretanto, atualmente, é muito difícil de projetar tais circuitos, dada a falta de ferramentas de automação e de bibliotecas de componentes básicos. Dessa forma, o uso de circuitos assíncronos é praticamente limitado a abordagens full-custom. A fim de contribuir para a superação dessa barreira, o Autor está envolvido na pesquisa de circuitos assíncronos há cinco anos. Este trabalho apresenta detalhes de parte dessa pesquisa, incluindo a implementação de cinco roteadores de redes intra-chip não síncronos, uma biblioteca de células com mais de quinhentos componentes assíncronos e um fluxo de projeto proposto para o projeto de tais componentes.

LIST OF FIGURES

LIST OF TABLES

CONTENTS

1 INTRODUCTION

Synchronous digital circuits assume not only the use of discrete voltage levels that translate into Boolean values, but also a discrete notion of time. The latter is generally implemented with all components sharing a common control signal distributed throughout the circuit, the so-called *clock* signal. In this way, the complexity of designing circuits is considerably reduced, as the notion of event sequencing is commanded by pulses of the clock signal everywhere. However, as semiconductor technology nodes evolve, new challenges emerge concerning the design of synchronous circuits, specifically due to this signal's global nature.

Transistors are getting smaller and electrical characteristics such as static power consumption are no longer irrelevant in the design of a synchronous digital circuit. In this way, tighter design constraints are imposed on clocks and other global signals, increasing design complexity [EKE10]. Moreover, guaranteeing that the clock signal presents exactly the same phase and skew in all registers of a modern complex integrated circuit (IC) can be very difficult. Even when a synchronous design is fully verified, variations in the fabrication process of a single transistor may jeopardize the whole circuit behavior, due to strict timing constraints imposed by the synchronous paradigm. Timing closure of complex designs and design robustness have become major concerns in the last years [ZHA07].

Albeit techniques to cope with the difficulties of designing a synchronous circuit exist, they can end up being very expensive. For instance, the clock tree, employed to guarantee same phase and skew in the clock signal of all registers of a circuit is typically responsible for 30% and up to 50% of the total power consumed by a synchronous circuit. For this and other reasons, using non-synchronous design techniques for the design of future ICs is considered inevitable [ITR08].

Asynchronous circuits [SPA01] assume the use of discrete voltage levels that translate into Boolean values, just as synchronous circuits do, but the latter consider time as a continuous quantity. Synchronization, communication and sequencing of events among components are done through local handshaking. The result in "synchronous terms" is that registers are only activated when and where it is necessary. Such characteristic provides advantages when employing the asynchronous paradigm and according to [BER99] [SPA01] [MYE01] [BOU07] and [CRI10] for example, asynchronous circuits may present:

- Lower dynamic power consumption: when not computing, the circuit is quiescent, i.e. consumes only leakage current. When requested, the circuit starts to compute and consumes dynamic power. However, after the computation of a task, the circuit returns to a quiescent point until the next request arrives.
- Higher operating speed: the operating speed is given by local latencies rather than by a global signal. Asynchronous circuits may thus operate with average case delays, while their synchronous counterparts are designed based on worst case delays.
- Better composability and modularity: since synchronization is given by local handshakes rather than a global signal.

- Lower electromagnetic emissions: because registers do not switch all or most at the same time. Instead, they switch at random moments, based on local requests.
- Higher robustness: asynchronous circuits may continue to operate correctly over a large range of variations in power supply and over several technologies. Moreover, since no assumptions are made about the inter-cell communication delays, such circuits are more tolerant to process, voltage and temperature variations.

Complex systems-on-chip (SoCs) are composed today by a large amount of intellectual property (IP) cores, each of each using particular standards and/or protocols and presenting varying design constraints. Often, the requirements for each IP core determine the use of specific communication protocols and/or operating frequencies. These requirements make the design of SoCs easier if implemented with multiple frequency domains that communicate and synchronize using asynchronous techniques. SoCs designed through this approach are classified as globally asynchronous locally synchronous (GALS) [CHA84] and may help overcoming the limitations of synchronous design, while at the same time maintaining a mostly synchronous design flow. Conversely, the lack of adequate electronic design automation (EDA) tools imposes a great barrier in the design of asynchronous circuits/interfaces. This is because the majority of commercial applications target the synchronous paradigm. According to the ITRS 2009 Edition [ITR09] further progress in the asynchronous paradigm depends on commercial tool support. The high complexity of modern ICs makes the task of ensuring that such designs will operate correctly on silicon nearly impossible without the use of adequate EDA software and consolidated design methodologies [RAB03]. Additionally, most standard cell libraries provided by vendors still contain only basic devices for synchronous design, such as logic gates and flip-flops, and efficient asynchronous designs require other specific devices as well. In this way, asynchronous design is still used only in restricted applications domains.

The present work describes a set of contributions brought by the Author in previous years (since 2008) to advance the solution of some of the problems that restrain an ampler utilization of GALS and asynchronous systems and circuits in practice. In this period of time, several works were conducted by the Author to help in the development of design automation and facilitation of the design for asynchronous circuits and GALS systems.

1.1 Contributions for Asynchronous EDA

The Asynchronous Standard Cells Enabling n Designs (ASCEnD) Library [MOR10] and [ASC12], is a standard cell library designed to support the implementation of asynchronous circuits. It was initially designed for the XFab 180 nm CMOS technology as a simple set of basic asynchronous components drawn at the standard cell layout level. Its most recent version contains over five hundred standard cells and addresses the STMicroelectronics 65nm CMOS technology. Moreover, current work works on an extension of the library to other technologies. ASCEnD was built as part of the Author's End of Term work and has been since employed in the design of three NoC routers and other circuits. By design, ASCEnD is compatible with the basic standard cell libraries of the underlying technology and has been extensively validated through simulation at several abstraction levels, including post-layout extraction at the physical level.

All components of ASCEnD result from a design flow proposed by the author in [MOR11a]. The flow counts with automated tools and the only manual process is the layout generation. An extension of this work modifies the flow to provide the designer with a choice between low power and high performance designs [MOR11b]. Currently, this flow is parameterizable and compatible with most CMOS technologies.

In [MOR11a], the Author also proposes the integration of ASCEnD with the Manchester Teak asynchronous synthesis tool [BAR09], based on the Balsa language [BAR98] [EDW06] and [SPA01]. Both, Balsa and Teak are developed and maintained by the Advanced Processor Technology Group [APT12] of the University of Manchester. Currently, ASCEnD is also compatible with the Balsa synthesis environment itself. Results show that lower area, higher performance and lower power consumption can be obtained with low design effort, since Balsa is a language specifically designed for implementing asynchronous circuits.

Another previous research work compares asynchronous components and their effect in the circuit after finishing the back end design. The article [MOR12a] describes part of this research. The paper demonstrates some weaknesses of the Teak synthesis tool, and points to possible directions for improvement of the tool. Current work comprises optimizing Teak and Balsa synthesis flows. To do so, a study is under way to scrutinize how asynchronous synthesis tools map the circuit to locate specific weaknesses and search for better solutions.

1.2 Non-synchronous NoC Routers

The Author took part in the design of five non-synchronous NoCs. Hermes-G and [PON08] is a NoC and associated router, planned to allow the implementation of GALS systems, by serving as a GALS system communication infrastructure. Its low power version is Hermes-GLP, presented in the same work. The router was developed to reduce power consumption and is based on a simple dynamic frequency scaling (DFS) mechanism that allows significant power dissipation reduction with small latency penalty. Instances of the Hermes-GLP NoC were prototyped and validated in field programmable gate arrays (FPGAs) as described in [HEC12]. There is currently an effort by others to design the Hermes-GLP modules in a 65nm CMOS technology and validate it through ASIC prototyping.

Hermes-A [PON10a] is a NoC router that employs a distributed routing scheme, where the router itself decides which path incoming packets will follow. The router was initially designed in the XFab 180nm CMOS technology and mapped to the foundry-provided standard cell library, and the initial version of ASCEnD. Later, it was implemented in the STMicroelectronics 65nm CMOS technology [PON10b], when the newest version of ASCEnD was released. Hermes-A was designed through a semi-custom approach, where handshake components are generated as schematics of standard-cells, described in VHDL and interconnected in the same way at the top level circuit. The router was synthesized using typical commercial EDA tools, designed for synchronous circuits, where adaptations were necessary in the design flow, to generate a functional circuit. The netlist was placed and routed and validated through simulation after circuit extraction at physical layout level.

Hermes-AA [PON10b] is an extension of Hermes-A, where packets are routed through an adaptive routing algorithm. The router was implemented in STMicroelectronics 65nm CMOS technology with the same approach used in the design of Hermes-A.

However, the design of NoCs Hermes-A and Hermes-AA demonstrated to be very laborious handcraft, due to the required steps inferred by the use of synchronous EDA tools when designing asynchronous circuits. Therefore, once ASCEnD was integrated with the Balsa synthesis processes, an asynchronous NoC router was described in Balsa language, reducing design complexity. The router was called Balsa-Based Router (or BaBaRouter) [MOR12b]. The router is based on the original Hermes NoC router [MOR04] a very simple router that uses handshake communication. BaBaRouter has exactly the same functionalities and structure as the Hermes handshake router. It was implemented in the STMicroelectronics 65nm CMOS technology and mapped to the cells of ASCEnD and of the foundry-provided standard cell library. The BaBaRouter was validated through simulation after extraction at physical layout level. The obtained results are quite promising. The router proves to be faster and is less power- and area-consuming than Hermes-A and Hermes-AA. Moreover, the design effort is drastically reduced when using Balsa.

1.3 Objectives

The objective of this work is to present a set of contributions aimed mostly at the development of automation methods for the design of asynchronous circuits and GALS systems. These contributions encompass the development of a parameterizable and automated design flow for typical asynchronous standard cells, as well as a standard cell library containing over five hundred components dedicated to asynchronous design. This library was integrated with automated tools for implementing asynchronous circuits and validated at the physical layout abstraction level. Furthermore, three asynchronous and two GALS network on chip (NoC) routers were designed and validated. Some of these targeted FPGA implementations and other ASIC versions up to the layout level. The conducted works indicate that the development of EDA tools for supporting the asynchronous paradigm can help overcoming difficulties imposed by current and future silicon technologies. This last observation points to a future PhD theme the Author intends to follow starting in 2012. The results obtained through the development of these activities were already published in seven international conference articles.

2 CONCEPTS

This Chapter presents basic concepts of asynchronous circuits and addresses the Balsa and Teak systems, which comprise a language and a framework for describing and implementing such circuits in ASIC and FPGA technologies, as well as two asynchronous circuit synthesis methods.

2.1 Asynchronous Circuits

A digital circuit is (fully) asynchronous when no clock signal is used to control any sequencing of events. Instead, modules use local handshaking among them to synchronize, communicate and operate [SPA01] [MYE01]. In "synchronous terms", the resulting behavior is registers being enabled to store new values only when and where this is needed [BAR98] [SPA01]. Such characteristic presents advantages over the use of a global clock signal in modern technologies.

Asynchronous circuits can be classified according to several criteria. One important criterion is based on the delays of wires and gates. The most robust and restrictive delay model is the delay-insensitive (DI) model, which operates correctly regardless of gate and wire delay values. Unfortunately, this class is too restrictive. The addition of an assumption on wire delays in some carefully selected forks enables to define the quasi-delay-insensitive (QDI) circuit class. Here, signal transitions must occur at the same time only at each end point of the mentioned forks. QDI circuits are quite common, although other models, such as bundled-data [SPA01] are still used in specific contexts.

There are different ways to encode data to adequately support delay models. The use of regular binary encoding of data implies the use of separate request-acknowledge control signals. While this makes design straightforward for those used to synchronous techniques, the timing relationship between control and data signals needs to be guaranteed at every handshake point, making design of large asynchronous modules difficult and hardly scalable. As an alternative, DI encodings are robust to wire delay variations, because request signals are embedded within data signals. An example is the dual-rail encoding, that uses two wires to represent each bit, and can represent bit values as well as the absence of data. The request signal is computed from the data and therefore demands extra hardware. More efficient DI encodings exist and are discussed in detail in several publications e.g. in [AGY10].

2.1.1 The Muller C-Element

Most of the asynchronous design techniques proposed to date require devices other than the ordinary logic gates and flip-flops available in current standard cell sets. These include e.g. metastability filters, event fork, join and merge devices. Although most of these may be built from logic gates, in general this is inefficient. A fundamental part that enables to build such devices more effectively is the Muller C-element, also called just C-element [SPA01]. Its importance comes from the fact that C-elements operate as an event synchronizers. Figure 1 depicts the truth table and state diagram for a C-element with symmetric behavior. Its output may switch only when all its inputs assume a same logical value. When inputs A and B are equal, the output Q assumes this same value. However, when these inputs are different, the output keeps the previous logic value. It is possible to build and use several alternative similar behaviors, by individually negating the inputs or the output, increasing the number of inputs and associating differentiated logic behavior to distinct inputs. This last characteristic produces the asymmetric C-elements, which are discussed for example in [TOM06].

A	B	Q_i
0	0	0
0	1	Q_{i-1}
1	0	Q_{i-1}
1	1	1

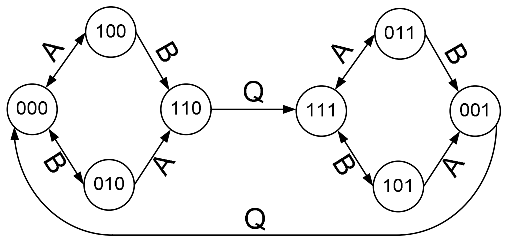


Figure 1 – A basic 2-input C-Element truth table and its state diagram, assuming a symmetric behavior with regard to all inputs.

2.1.2 Metastability

Metastability Filters are crucial components to implement asynchronous circuits. This is due to the fact that they guarantee a robust implementation of control elements, like arbiters. These elements are essential to compute data and take decisions for the correct operation of the circuit [SPA01]. The role of this component is to decide which of two concurrent events is to be served first. In ASCEnD, metastability filters have two request inputs (RA and RB), that receive requests, which can be concurrent, and two acknowledge outputs (AA and AB), that signal which event is to be served at the moment. The logical function of this component is showed in Equation (1). In the case of two events taking place in the exact same instant of time, e.g. simultaneous low-to-high transitions in RA and RB, the filter will eventually decide for one, eliminating the possibility of metastable states. This scenario, which is not covered by the logical function, is decided through an electric race, where the request with best potential is served first. Because AA and AB are mutually exclusive, the request with lower potential will only be served after the first finishes its communication.

$$AA = \overline{RB} \wedge RA \quad \text{Equation (1)}$$

$$AB = \overline{RA} \wedge RB \quad \text{Equation (2)}$$

2.2 Asynchronous EDA

With the growing interest for asynchronous circuits, different tools have been proposed, in order to automate asynchronous design. Among these, Balsa [SPA01] [BAR98] stands as a comprehensive open source environment. The tool was designed and is maintained by the Advanced Processor Technologies Group of the University of Manchester.

Balsa is both a language to describe asynchronous circuits and a framework to simulate and synthesize them. The compilation of a Balsa description is transparent, since language constructs are directly mapped into handshake components [SPA01]. In this way, it is relatively easy for the designer to visualize the circuit-level architecture of a Balsa description, contrary to what happens in ordinary hardware description languages (HDLs) like VHDL and Verilog. Moreover, Balsa description changes reflect in predictable changes in the resulting circuit, which means that the designer has a clear control of the generated hardware.

Balsa requires the designer to give, through language operators, the order of handshake events, where these events mean communication between handshake components (data exchange or pure control) and can be sequential or concurrent. Moreover, handshake components are transparent and are given through higher abstraction language constructs. Therefore, one of the main advantages of using Balsa to describe and implement asynchronous circuits is that communication control between handshake components is abstracted. Due to the fact that all the complexity of describing control signals is transparent, the designer just needs to describe the data flow and implement its logic. The tool automatically generates the handshake components and the required control.

Different technologies can be used to synthesize Balsa circuit descriptions, from FPGA to ASICs. This work uses the previously mentioned ASCEnD, an in-house standard-cell library designed to support asynchronous circuits, fully compatible with the Balsa framework. In this way, netlists generated by Balsa can be imported into back-end commercial tools for physical implementation. Finally, Balsa allows the designer to choose the design style/encoding type used by the target circuit. The available choices are bundled-data/binary, QDI/dual-rail and QDI/1-of-4.

Another option to synthesize Balsa descriptions into a netlist of asynchronous devices is the open source system designed at the University of Manchester: Teak [BAR09]. It consists in a new target parameterizable component set and a synthesis scheme that aims at the improvement of circuits described in the Balsa language. The tool optimizes Balsa descriptions synthesis by replacing data-less activation channels with separate control channels. Albeit the Balsa System allows different data encodings over two- or four-phase protocols, Teak implementations are typically restricted to QDI, four-phase, dual rail implementations. Hence, circuits synthesized through this tool are limited to that choice of design style, protocol and data encoding.

3 ASYNCHRONOUS EDA CONTRIBUTIONS

The interest in non-synchronous circuits is increasing. The International Technology Roadmap for Semiconductors (ITRS) in its 2008 edition [ITR08] describes a clear need for asynchronous communication protocols in integrated circuits (ICs) control and synchronization along the next decades. For example, the ITRS estimates that global clock-based ICs, which comprised 93% of the chips sold worldwide in 2007, will have only 55% of the market by 2022. The other 45% of the chips will be local handshaking circuits, including clockless or multi-clock ICs. However, the lack of adequate electronic design automation (EDA) tools imposes a barrier to the design of asynchronous circuits. Most commercial EDA tools focus currently on purely synchronous designs.

In order to help overcoming the difficulty of implementing efficient asynchronous circuits, a design flow for C-Elements was proposed by the Author and a library of over 500 cells was designed for a 65nm technology.

3.1 C-Element Design Flow

A standard cell is an elementary device, such as a logic gate, defined at the chip layout level, with some predefined characteristics that usually comprise cell height and current driving strength. Chip vendors and associated enterprises provide designers with libraries of standard cells and support to define design methods based on them. Cells are characterized in detail, and their standardization facilitates automation of IC design. Most current Application Specific Integrated Circuit (ASIC) SoC designs use standard cells extensively, e. g. to meet time to market constraints. Indeed, standard cells are the key to speed up the design of high performance ICs and are often referred as the main success factor for the rapid growth of integrated systems technologies [ERI03].

The efficient implementation of asynchronous circuits often requires devices other than those available at current commercial standard cell sets. Moreover, asynchronous design is in fact not a single alternative to the synchronous paradigm, but a collection of methods, several assuming the existence of a specific set of basic devices. Thus, no single set of new cells may be useful for every asynchronous design technique. In [MOR11a] a set of cells is proposed, along with an automated novel design and characterization flow. This enables the construction of asynchronous standard cell based ASICs. The cell set is integrated with the Teak System [BAR09], a synthesis tool for high level asynchronous circuit descriptions. Teak netlists are quasi delay insensitive (QDI) and employ 4-phase dual rail protocols. They can be placed and routed with commercial tools.

Some works propose methods to design asynchronous circuits with conventional IC design tools and standard cell libraries. For example, Chong et al. [CHO07] suggest a method that enables creating latch-based pipelines and an asynchronous latch controller for stage synchronization. As example design the work proposes an asynchronous FIR filter in a 0.35µm CMOS process.

Cortadella et al. present the desynchronization paradigm [COR06] for automating the design of asynchronous circuits from synchronous specifications using synchronous tools. The design flow consists in exchanging each pipeline edge-triggered register by two level-sensitive latches with local handshake units for local synchronization, eliminating the need for a global clock. Combinational logic delays need to be matched in control lines. Validation occurs through desynchronized versions of a DES cryptography core and a DLX processor. None of these works exploit the asynchronous paradigm capabilities, limiting themselves to bundled-data protocols, which prevent the use of more robust and performing DI approaches [SPA01]. Moreover, the need to adjust combinational logic delays by hand results in circuits that are hard to reuse in other contexts. Additionally, both works assume the use of conventional standard cell libraries, which may lead to designs that spend more area and/or power.

The asynchronous literature proposes a few standard cell libraries. Ferretti and Beerel suggest the single-track full-buffer (STFB) template library, designed to support high-speed area-efficient asynchronous nonlinear pipeline designs [FER06]. It employs the TSMC 0.25µm technology and contains cells to support dual-rail and 1-of-3 data encoding circuits, being available through the MOSIS prototyping service. Ozdag and Beerel propose the QDI pre-charged half-buffer (PCHB) template library for asynchronous low-power high performance circuits [OZD06]. This library also uses TSMC 0.25µm and is equally available through MOSIS. It counts 40 dynamic cells to implement function blocks and 10 cells to implement control logic. Gulati and Brunvand describe a specific macro cell library to support the design of asynchronous microengines [GUL05]. The library employs the AMI 0.5µm technology, contains 66 cells and was validated through a test chip that implements a differential equation solver microengine. Most cells in the three discussed libraries display a short range of output load capacitance, due to the fact that they are deemed to validate a template-based design technique or support specific applications. Consequently, no fine grain optimization is proposed in the design of the libraries themselves. Besides, they limit the design of asynchronous circuits to specific templates. Other asynchronous design styles may require different sets of devices.

To overcome specificity issues, the French laboratories TIMA and LETI developed a library with cells to support more generic QDI circuit design. Maurine et al. present an STMicroelectronics 130nm technology version of this library [MAU03]. A private communication to the Author informs that this library is currently available in the STMicroelectronics 65nm CMOS process and several chip designs use it at the LETI laboratory.

The work proposed in [MOR11a] is fully integrated with back-end commercial EDA tools and some front-end asynchronous tools. The cell subset this work addresses enables automatic generation of circuits from high level descriptions and allows designers to experiment with different asynchronous templates and styles like bundled-data and dual-rail DI. The cell design flow is fully automated, except for the physical cell layout generation and technology specific requirements.

To enable implementing netlists generated by Teak in real processes, a very small specific set of standard cells is required. TABLE I shows this cell set. All cells are variations of the basic C-element. The proposed flow for each cell appears in Figure 2 and comprises three main steps: specification, design and validation. To synthesize Balsa descriptions, Teak also needs conventional standard cells like ANDs, ORs and AND-OR-INVERTERs. These components are available in most conventional standard cell libraries.

TABLE I – Proposed cell set logic functions [MOR11a]. Q is the cell single output. RST is a reset signal. Note the last cell displays an asymmetric behavior.

Cell	Logic Function
C2	$Q_{i+1} = (A \wedge Q_i) \vee (A \wedge B) \vee (Q_i \wedge B)$
C3	$Q_{i+1} = (A \wedge B \wedge C) \vee (A \wedge Q_i) \vee (B \wedge Q_i) \vee (C \wedge Q_i)$
C2R1	$Q_{i+1} = RST \wedge ((A \wedge Q_i) \vee (A \wedge B) \vee (Q_i \wedge B))$
C1U1	$Q_{i+1} = B \wedge ((A \wedge B) \vee (Q_i \wedge B))$

Functionality is given as an expression defining the output(s) as a function of the circuit input(s), as in TABLE I. The electrical requirements consist in the cell speed for charging or discharging its output(s). The electrical specification of a standard cell is a tradeoff between area, power and speed. High performance gates require larger transistors and consequently consume more power and demand more silicon area. Usually, different cells with a same logical function support different output driving strengths. In the proposed flow, the designer defines the required cell speed to meet constraints or for compatibility with another gate set.

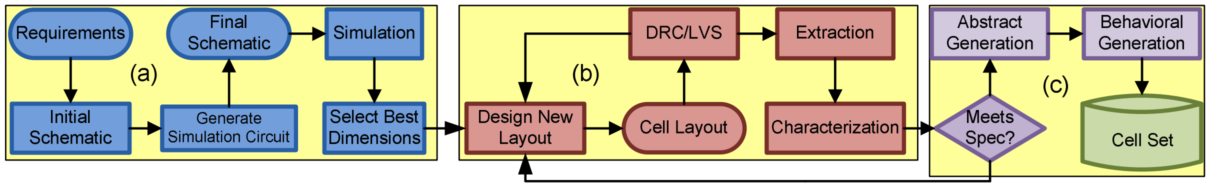


Figure 2 – Standard cell proposed design flow. The three main steps are: (a) Specification, (b) Design and (c) Validation. Actions are represented by boxes, decisions by diamonds, descriptions as rounded corner boxes and the repository as a cylinder.

The next step is to design an initial schematic of the cell. Figure 3 presents this schematic implementation for the logic function of the C2 cell from TABLE I, a 2-input C-element.

The schematic is then exported to a SPICE description. The basic structure of the example C-element (in fact, of any similar cell) can be represented by blocks (1), (2) and (3) of Figure 3. In [SHA98] a similar approach is used to implement C-elements in CMOS technologies. Here, (3) represents the state keeper, (2) is the output driving inverter and (1) is the inverted logic function. The first (state keeper) is usually set to minimum transistor sizes, since it does not influence the switching performance of the output and is used only to statically keep the cell state. Block (2) is responsible for driving (charging/discharging) the output load of the cell. Its transistors size is obtained by simulating an inverter with varying transistor sizing until the required output driving strength is obtained (as de-fined by the electrical specification).

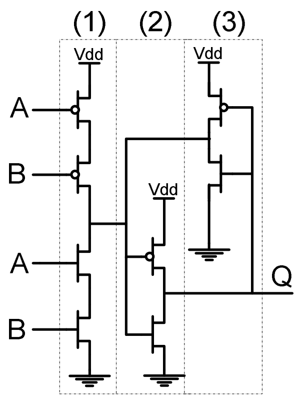


Figure 3 – Examples of schematic for the C2 cell, a 2-input C-Element.

Once the dimensions of the driving inverter are fixed, a specifically tool designed in the context of this work, called Ring Oscillator Generator (ROGen) automatically produces a circuit that can be simulated to define the size of the transistors that compose block (1). This tool is implemented in C language. Its input is a configuration file and the schematic of a NAND cell of the target technology and the cell to be designed, which transistors are not yet dimensioned. The output is an oscillator ring, as presented in Figure 4, and its simulation environment, described in SPICE language.

The circuit generated by ROGen is the ring oscillator represented in Figure 4, composed by a NAND and 10 of the cells to be sized. The NAND controls the oscillation.

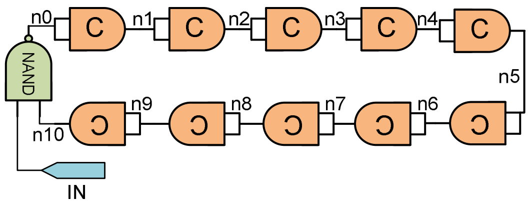


Figure 4 – Example of the gate level view of a simulation circuit generated by ROGen.

Next, extensive simulation defines the number of cell instances in the ring (10) as an amount sufficient to normalize the effect of the NAND in the circuit and allow correct evaluation of the C-element. The circuit is described in SPICE and variations on the size of the transis-tors are achieved through the use of the SPICE .alter function from minimum size to three-finger transistors with maximum size. A finger is the maximum transistor size that can be drawn without using layout folding.

Simulating the SPICE circuit description, information about electrical behavior of each transistor size is obtained through the SPICE .measure function. This information comprises: dynamic and leakage power of the whole ring, rise and fall propagation times and transition delay of each cell, and operating frequency of the whole ring. Leakage power is the average power consumed from the power source while the circuit is in a static state (IN=0) and the dynamic power is the average power consumption from the source when the ring is oscillating (IN=1). A second tool designed in the context of this work is the Cell Specifier (CeS). It is used to analyze the results obtained through simulation and choose the PMOS transistors size that best drives the cell for each NMOS transistor size variation. Its input is the results generated by the simulation of the file generated by RoGen. The result is the set of the best PMOS-NMOS combination for each NMOS size variation. The output is a set of charts, showing all simulations and pointing the fastest combination.

Once the size of the transistors is defined, step (b) of Figure 2 flow may occur. From the circuit schematic, physical characteristics of the standard cell can be designed with the use of a layout editor. The layout must respect layout design rules, verified through a Design Rule Check (DRC) tool and Design for Manufacturability (DFM) rules along with a layout versus schematic (LVS) check. Once the layout is designed and fully verified, the parasitic effects must be extracted and electrically characterized for the specific process. The electrical characterization is a crucial step in the generation of standard cells, since it defines the behavior of the circuit for different operating conditions and generates information required by EDA tools. After characterization, a cell must be properly verified.

The last flow step consists in the validation of the cell by checking if the information generated during characterization is equivalent to that defined in the specification. Moreover, timing simulation must be carried out for a design composed by a single cell to check if the delay obtained during characterization is correctly annotated and the behavioral description is followed. If the cell passes this verification step, it can be used in a design.

Besides the layout/schematic views, two other views are needed: an abstract one, used by place and route tools and to assemble the circuit on a chip; and a behavioral view, for use in high level simulations. The abstract view is usually generated automatically by a layout tool. Its format is a designer choice. The behavioral view can be captured by some HDL code, like Verilog or VHDL. In Verilog for instance, cell behavior can be implemented through user defined primitives (UDPs) defined as truth tables, and a Verilog module associates the UDP to a cell behavioral view and defines the cell pins. Once the cell is fully designed, all the views are generated and specification is met, the cell may be aggregated to the cell set repository. With this repository, the designer is able to implement QDI dual rail asynchronous circuits.

The cell set specified in [MOR11a] was designed using the proposed flow for the STMicroelectronics 65nm, general purpose, standard Vt technology. A 32-bit public key cryptographic circuit running the RSA algorithm was described in Balsa and implemented using the cells. The circuit was synthesized with Teak, obtaining a netlist. This netlist was automatically exported by Teak to a Verilog netlist composed of a set of components in the target library, in this case, the combination of the conventional STMicroelectronics standard cell library and our asynchronous cell set ASCEnD.

The Verilog netlist was simulated with a VHDL testbench, running multiple encryption and decryption operations. After verifying functional circuit correctness through simulation, the netlist was placed and routed using the abstract views provided by STMicroelectronics and those designed within ASCEnD. The design uses a total of 132,274 standard cells, for a total area of 0.41 mm^2. The circuit employs five metal layers to route, giving a total wire length of 843,668.855 µm.

To compare results, a synchronous version of this circuit was also implemented. Its area was roughly fifteen times smaller than the asynchronous circuit obtained through Teak synthesis, as detailed in TABLE II.

TABLE II – A comparison of asynchronous and synchronous implementations of an RSA cryptographic core.

	Asynchronous RSA	**Synchronous RSA**
Standard Cells	132,274	7,712
Total cell area	0.41mm^2	0.027mm^2
Total wire length	843,668.855µm	57,691.770µm

This overhead is not unexpected, not only because asynchronous logic takes more area, but mostly because asynchronous synthesis techniques are still largely undeveloped. Also, the total power consumption is 3.5 times higher in the asynchronous circuit, as detailed in TABLE III. This is due to the elevated leakage and switching power consumption, consequences of the large area overhead. Internal power increase however, was not so significant, just 1.5 times higher.

After place and route, the delay of each circuit path was annotated, using the electrical characteristics provided by STMicroelectronics and those of the proposed cell set. Next, the post placed and routed netlist was generated and timing simulations for different encryption and decryption operations were conducted, to verify the functionality of the mapped circuit. The correct operation proved the successful integration of Teak, the ST conventional library and the proposed cell set.

TABLE III – A comparison of power consumption for asynchronous and synchronous implementations of the RSA cryptographic core.

	Asynchronous RSA	**Synchronous RSA**
Internal Power	2.625mW	1.816mW
Switching Power	4.834mW	0.515mW
Leakage Power	2.201mW	0.411mW
Total Power	9.66mW	2.742mW

The proposed cell set and the validation design showed a successful vertical integration of front-end (Balsa-Teak) back-end (Cadence) and standard cell libraries for achieving asynchronous designs. Current work includes prototyping the test chip to validate the cell set on silicon.

The elevated area consumption revealed in the case study is a consequence of a clear lack of development in asynchronous EDA tool technology. These results are certainly worse if techniques like employing conventional logic gates to implement asynchronous logic were employed. For example, a 3-input C-element cell in a 65nm technology has an area of 7.80µm^2, while the required conventional cells to implement the same logic would take 16.64 µm^2, counting cell area alone. In the asynchronous RSA circuit case study, roughly 20% of the standard cells were 3-input C-elements. Implementing this circuit with conventional cells would result in an additional area overhead of 0.240mm2, without taking into account other C-element variations used in the design. Such increase in the area would dramatically affect static power consumption and average latency, easily leading to depletion of the advantages of using asynchronous circuits. Moreover, improving EDA tools and specific cells to design asynchronous circuits can present a solution to deal with some of the emerging constraints of deep submicron (DSM) technologies.

The proposed design flow for C-Elements provides tools to automate transistor size generation, simulation and transistor sizing choice processes. As decribed previously, the original flow uses a tool called CeS, to select the fastest combination of transistor sizes as the best transistor sizing, from the large set of resulting simulations, disregarding power consumption. This version is named Best for Speed (BS). Another development published in [MOR11b], suggests an enhancement for this tool, by modifying CeS to output another implementation choice. The tool now selects the best speed and power consumption tradeoffs and presents them in a chart. In this way designers can choose either the BS or the implementation more adequate for low power consumption, called Best for Power (BP).

Figure 5 presents an example output of the new CeS for an instance of a 2-input C-element. In this case, a total of 1219 simulation scenarios were automatically generated and simulated, each obtained by varying dimensions of block (1) transistors for this C-element (see Figure 3). The "Max. Freq." curve shows the maximum frequency achieved by ring oscillators composed of 10 C-elements and a control NAND, for each scenario. The "Pwr. Eff." curve depicts the power efficiency of the simulated circuits, in Watts per Hertz at the maximum operating frequency.

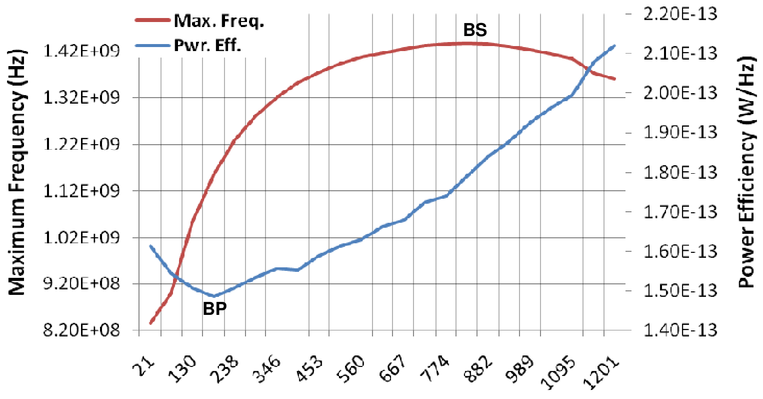


Figure 5 – Example of CeS output for a C-Element.

As Figure 5 shows, the BS version corresponds to the transistor sizes (in fact widths) that provide the highest operational speed. The BP version, in turn, is the scenario corresponding to less Watts per Hertz. TABLE IV presents the specific transistor widths for each implementation and their respective operating frequency and power consumption figures. Results are part of the output generated by CeS and based on the ring oscillators' simulation. In this case, the BP implementation operates at only 80% the frequency of the BS version. However it consumes only 66% of the power required by the BS version.

TABLE IV – Transistors gate widths and performance figures for the circuits simulated with BS and BP C-Elements.

	BS	BP
NMOS Width (µm)	0.900	0.250
PMOS Width (µm)	1.700	1.300
Operating Frequency (GHz)	1.440	1.160
Total Power Consumption (mW)	0.257	0.171

The design phase, second step of the proposed flow, consists in taking the selected transistor sizing and producing a physical view, a layout that fulfills this specification. This includes hand-drawing the cell in a layout editor for the chosen process, extracting parasitic and electrically characterizing the resulting circuit for the selected process. Figure 6(a) and Figure 6(b) show, respectively, the physical layout of the BS and BP implementations of the example 2-input C-element, drawn to fulfill the parameters obtained by CeS. Note that the BS version requires larger transistors, to achieve higher operating frequencies.

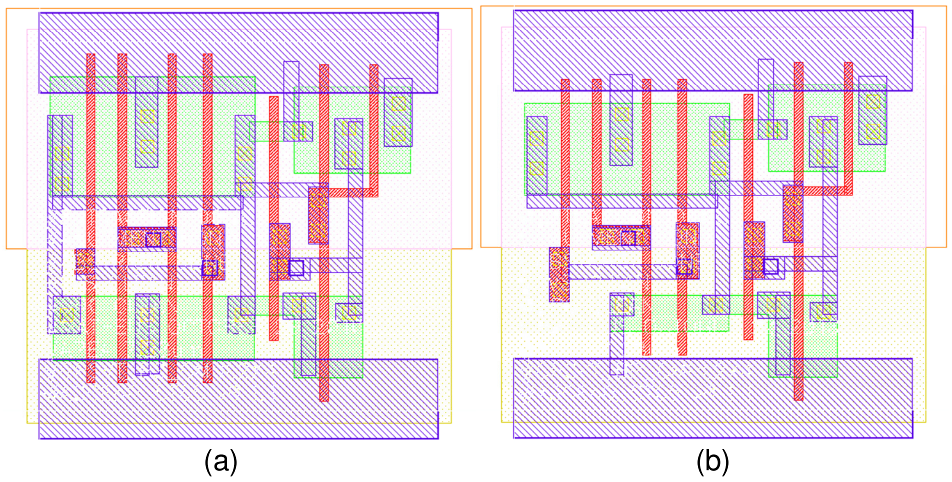


Figure 6 – Physical layouts for (a) one BS and (b) one BP implementation of the 2-input C-Element.

Again, a 32-bit public key RSA crypto core was described in the Balsa language and implemented using the BS and the BP cells through Teak synthesis. Due to the fact that in Teak synthesis operations are implemented through delay insensitive minterm synthesis (DIMS) logic, which is basically composed by C-elements [SPA01], the impact of these cells is more significant and can thus be efficiently evaluated. BS results are different from the ones showed in the first version of the design flow, because the driving strength used in this case study is different.

As explained before, the circuit was synthesized with the design tool Teak [BAR09] for two C-element versions, BP and BS. Teak generates QDI, dual-rail circuits. Netlists were automatically exported from Teak to Verilog. A VHDL testbench simulated the Verilog netlists, running encryption and decryption operations. After verifying the circuit's functional correctness through simulation, netlists were placed and routed using the abstract views provided by STMicroelectronics and those of ASCEnD. TABLE V shows area and wire length results for the two implemented designs. These results were obtained after physical design of the circuits. The BS version takes a total of 132,274 standard cells, for a total area of 0.43mm^2. The BP version requires 130,893 standard cells, for a total area similar to its BS counterpart, 0.41mm^2. Total standard cells difference is explained by the fact that BS cells require larger transistors. Thus, they require more silicon, and employ more physical cells in its design.

TABLE V – Standard cell area and wire length for the asynchronous implementations of the RSA in STMicroelectronics 65nm technology employing BS and BP versions of C-Elements.

	BS RSA	BP RSA
Number of standard cells	132274	130893
Number of C-elements	17041	17041
Total cell area (mm^2)	0.43	0.41
C-elements cell area (mm^2)	0.20	0.19
Total wire length (mm)	843.67	845.41
Average wire length (mm)	0.15	0.15

The number of standard cells in both designs is 54,776 if physical cells are omitted. Physical cells are the cells used to connect the power lines to the substrate, tap cells, and the filler cells employed in the core. From these, 17,041 are C-elements (~31%), attesting the importance of such devices in asynchronous circuits. In both designs, C-elements occupy 50% of the total cell area. The other 37,375 cells are ordinary logic gates from the foundry standard cell library. Note that higher level asynchronous components (fork, join, etc.) are built-up from C-elements and logic gates exclusively. No metastability filters were required in these designs. Clearly, C-element characteristics may have significant impact on the final design.

The delay of each circuit path after physical design was annotated using the electrical characteristics provided by STMicroelectronics and those of the proposed cell set. Next, parasitic effects extraction for the post-placed and routed netlist takes place, generating a Verilog netlist. With this netlist, timing simulations for different encryption and decryption operations were conducted to verify functionality. The simulation scenario was based on cryptographic operations of random messages and random keys for 100ms. The same scenario was employed in the simulation of both designs, meaning that random messages and keys were only generated once. The average times to perform a cryptographic operation were 34.084µs and 36.374µs for the BS and the BP versions, respectively.

According to the results obtained in the simulation of an oscillator ring, the BS version was expected to operate at a much higher frequency than the BP implementation. However, due to the fact that all the cells employed in this case study had the same drive (X4), the performance of both versions was equivalent in terms of speed. This is in fact a limitation of the Teak synthesis tool, which is not capable to choose among the distinct driving strengths available at the cell libraries. The choice during the RSA design was to ensure correct operation, picking the largest strength only. A more precise evaluation of the power-speed-area trade-off is a future work, to employ weaker driving cells where possible.

TABLE VI displays the power results of the RSA crypto core. The switching activity of the circuits nets, extracted from the timing simulations, are the source to conduct power analysis. The employed operating conditions are 1V and 25^o C. For the BS implementation, leakage power is roughly 42% of the total power consumption. In contrast, in the BP version leakage power is only 11%. This is relevant, since it is well known that leakage power plays an increasingly important role on DSM technologies [EKE10]. Moreover, comparing the implementations, the BP version consumes only 64% of the total power consumed by the BS version. Also, when comparing performance and power consumption tradeoffs for both designs, large power savings are obtained in the BP implementation for only a minor loss in throughput.

TABLE VI – Power consumption for the asynchronous implementations of the RSA in STMicroelectronics 65nm technology employing BS and BP versions of the C-Element.

	BS RSA	**BP RSA**	**Gain(%)**
Internal Power	2.335mW	2.012mW	13.83
Switching Power	4.300mW	3.007mW	30.07
Leakage Power	2.201mW	0.664mW	69.83
Total Power	8.836mW	5.684mW	35.67

BS and BP versions of the C-Element demonstrate how to achieve a good trade-off between speed and power. As it was expected, the BP implementation proved to operate at lower speeds than the BS version. However the improvement on power consumption is significant. Moreover, fine-grain tuning could be attained if design tools were able to mix combinations of BP and BS cells according to application requirements.

3.2 ASCEnD

The Asynchronous Standard Cells Enabling “n” Designs (ASCEnD) library was designed and is maintained by the GAPH research group of the Pontifical Catholic University of Rio Grande do Sul. Currently, the library is composed by two basic primitive component types, C-elements and metastability filters. Currently, ASCEnD is available for the 65nm STMicroelectronics CMOS technology (ASCEnD-ST65). A total of 504 C-elements compose the library as Figure 7 shows. It counts with C-elements with varying driving strengths (speed to charge/discharge a load), functionalities and topologies. Moreover, these components are available in high speed and low power versions, each with its own advantages. ASCEnD-ST65 C-elements were all designed according to the flow proposed in [MOR11a] and [MOR11b].

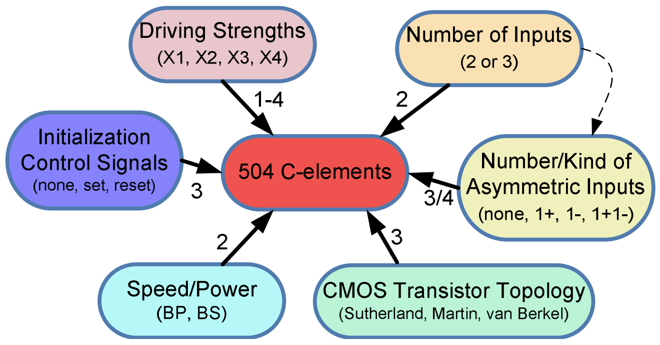


Figure 7 – ASCEnD-ST65 C-Elements composition. Rounded corner rectangles represent one parameter choice. Edges are labeled with the number of choices the parameter implies. The dotted edge represents dependencies between associated parameter choices.

ASCEnD-ST65 also counts with 4 versions of metastability filters. Each implementation employs different transistor sizes and, consequently, presents distinct driving strengths. Together with a typical standard cell library that contains combinational logic gates, the library can be used to implement asynchronous circuits of different classes through a semi-custom approach which is much less restrictive than template based approaches.

In ASCEnD-ST65 all the components are designed through the flow depicted in Figure 2. As showed before, this flow comprises three main steps: specification, design and validation. All components are designed at the physical layout level, employing design for manufacturability (DFM) techniques. Parasitics are extracted from the components and electrically characterized. The library was extensively validated through simulation after place and route of some complex applications, such as an RSA cryptographic core, as previously discussed, and three different network-on-chip routers, the subject Chapter 4. Finally, the library can be easily ported to other CMOS technologies, given that the adopted design flow counts with automated and parameterizable tools. The library is described in more detail in [MOR10] and [MOR12c].

Another work conducted by the Author, published in [MOR12a], compared different implementations of the C-Element, all available in ASCEnD-ST65. Three different static CMOS topologies of the C-element were implemented as showed in Figure 8. Figure 8 (a) shows the Sutherland pull-up pull-down implementation, proposed by Sutherland in [SUT89] and employed in micropipeline-based design methods. Another C-element type, shown in Figure 8 (b), was proposed by van Berkel in [BER92]. Finally, Figure 8 (c) shows the weak feedback C-element, proposed by Martin in [MAR89] and used extensively in the asynchronous microprocessors designed at Caltech.

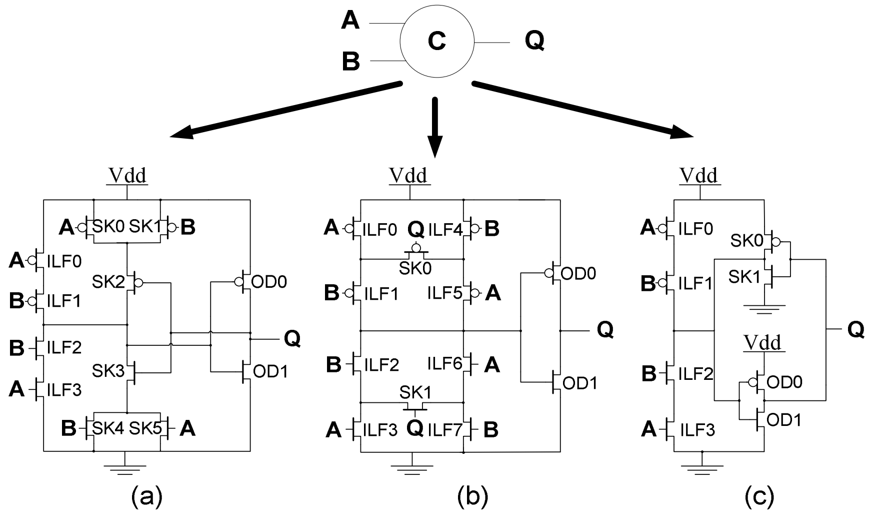


Figure 8 – Alternative CMOS topology implementations of the C-Element: (a) Sutherland's pull-up pull-down, (b) van Berkel's and (c) Martin's weak feedback.

Each of the three C-element topologies was designed through the ASCEnD C-Element design flow. They employ general purpose, standard threshold transistors for the 65nm STMicroelectronics CMOS technology. The obtained cells are able to drive the same load with the same speed, a maximum of 270fF in 1ns. In other words, the driving strength of each implementation is normalized, in order to precisely compare performance and area efficiency in a fair manner. The standard-cells are fully integrated within the synthesis flow of the Teak tool [BAR09], which builds asynchronous circuits from a high level description language, Balsa. Therefore the complexity of designing asynchronous ICs can be significantly reduced. Figure 9 shows an example layout for each of the designed standard cells.

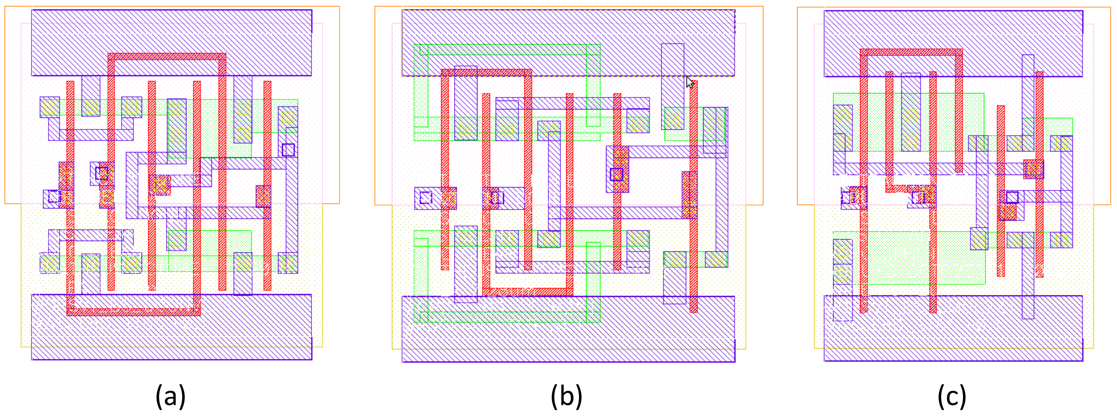


Figure 9 – Example physical layout at the cell level of the designed C-Elements: (a) Sutherland's pull-up pull-down, (b) van Berkel's, (c) Martin's weak feedback.

The silicon areas required by each C-element standard cell appear in TABLE VII. The cell that requires less area is the weak feedback C-element. TABLE VII also shows the resultant internal parasitic after RC extraction. As expected, the van Berkel implementation requires more silicon area and presents the highest parasitic capacitance, over two times the parasitic of the weak feedback C-element.

TABLE VII – Area and parasitic capacitance required by CMOS implementations of the C-Elements after RC extraction.

C-element topology	***Cell Area (μm^2)***	***Parasitic Cap. (fF)***
Sutherland	6.24	6.066
van Berkel	7.28	9.383
Martin's Weak feedback	5.72	4.469

After electrical extraction, each cell was characterized for a typical fabrication process corner, with typical delay for the NMOS and PMOS transistors, for an operational condition of 25^{o}C and 1 V power supply. TABLE VIII shows electrical results obtained through the electrical characterization. The weak feedback implementation presents the highest capacitance on its inputs. In fact, in comparison with the van Berkel implementation, which presents the lowest input capacitance, it shows an overhead of 85% of capacitance on its inputs. That is due to the fact that, albeit the van Berkel implementation is the most area consuming (due to the elevated number of required transistors), the weak feedback C-element is the one that employs larger transistors, as Figure 9 shows.

TABLE VIII – Capacitance of the input pins and power consumption of the designed standard cells, after electrical extraction.

C-element topology	***Input Capacitance (fF)***		***Average Internal Power[1] (fW)***		***Cell Leakage Power (nW)***
	A	***B***	***Rise***	***Fall***	
Sutherland	4.565	4.410	1.316	12.454	20.447
van Berkel	3.113	3.081	1.190	12.871	17.628
Martin's Weak feedback	5.633	5.849	4.889	21.930	30.591

As for the internal power required to switch the output of the standard cell, the Sutherland and van Berkel implementations are equivalent for rise and fall transitions. The former consumes slightly less internal power, roughly 2% in average. However, the weak feedback cell requires roughly 4 times the power required by the other implementations in rise transitions and almost twice the power required for fall transitions. As expected, the weak feedback topology has the highest leakage power consumption, when compared to the others. This is also a consequence of transistor size.

The average propagation delay of each cell, measured as the average delay of rise and fall transitions, was also obtained through electrical characterization. Figure 10 shows the obtained results for two scenarios. In Figure 10, the input slope was fixed in 1.2ps and the output load varied from 0.001pF to 0.15pF. The time required for the Sutherland and van Berkel topologies to switch their respective outputs is equivalent, as illustrated by the overlapping values. Moreover, the weak feedback presents equivalent propagation delay for small output loads (from 0.001pF to 0.015pF). However, the higher the load gets, the worse its propagation delay is, in comparison to the other topologies. This behavior shows that the weak feedback is also the implementation most sensitive to output load variations.

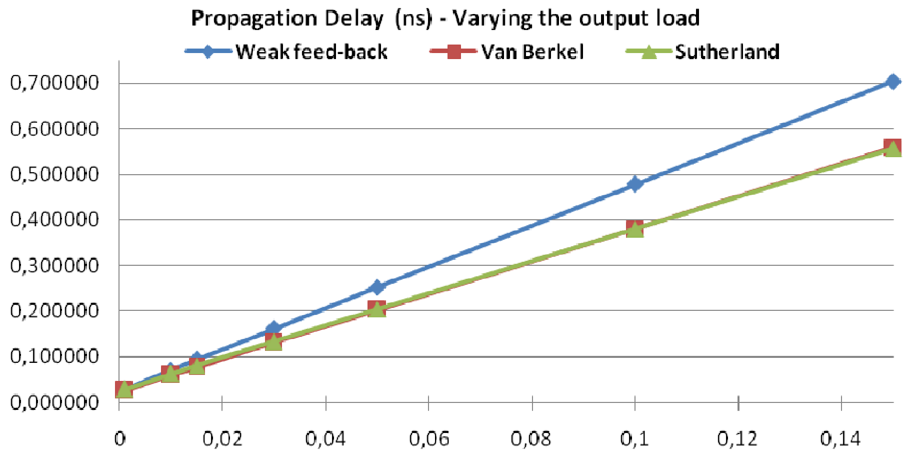


Figure 10 – Propagation delay of the designed standard cells after electrical extraction as a function of the output load capacitance. Results were obtained by fixing the input slope in 1.2 ps and varying the output load from 0.001pF to 0.15pF.

Figure 11 shows the electrical behavior of each implementation when the output load is fixed in 1fF and the input slope varies from 0.0012ns to 0.180ns. In this scenario, the fastest

[1] Average internal power was measured as the average power consumption of the input pins, for a scenario where the cell switches its logical value with an input slope of 1.2ps and an output load of 0.015pF.

topology is van Berkel’s, followed by Sutherland’s and next by weak feedback. The obtained results show that for small input slopes (0.0012ns to 0.0132ns), the speed of Sutherland’s and weak feedback C-elements is equivalent. However, as the input slope gets more significant, the propagation delay of the weak feedback topology gets worse. In this way, the weak feedback implementation is also the most sensitive to input slope variations. As Figure 11 shows, the delay of this implementation grows at a much higher rate as the input slope grows, while the other two implementations see their delay grow more linearly. The van Berkel implementation displays the smallest propagation delay, regardless of input slope variations. Therefore, we can consider the van Berkel C-element the most robust implementation for both input slope and output load variations. Results show an agreement with the work presented in [BER92], which conducted an analysis of C-elements threshold.

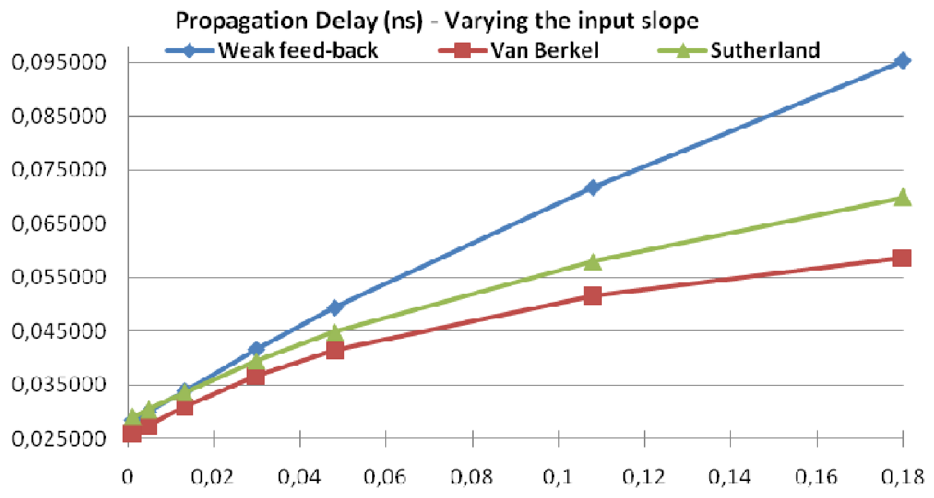


Figure 11– Propagation delay of the designed standard cells after electrical extraction as a function of the input slope. Results were obtained by establishing the output load in 1fF and varying the input slope load from 0.0012ns to 0.180ns.

As an initial comparison of the impact of each C-element implementation on asynchronous circuits, a low complexity circuit was employed, the oscillator ring showed in Figure 4. The circuit is composed by a NAND and 10 C-elements. Extensive simulation defined the number of C-elements in the ring (10) as an amount sufficient to normalize the effect of the NAND and allow correct evaluation of the C-elements. The NAND is required to keep the circuit static, when the “IN” pin is set to ‘0’, and to make the circuit oscillate, when the “IN” pin is switched to ‘1’. In this way, static and dynamic power consumption and operational frequency can be precisely measured. Three oscillator rings were generated, one for each implementation of the C-element after RC extraction.

After simulating each oscillator ring, power consumption and operational frequency were obtained through the SPICE “.measure” function, as TABLE IX shows. Leakage power was measured as the average power consumed from the power source while the circuit was quiescent, and the dynamic power was measured as the average power consumption from the source when the ring is oscillating. The frequency is measured as the inverse of the period between two similar edges in any node of the ring (in this case “n5”).

TABLE IX – Performance figures of the oscillator rings.

C-element Implementation	***Operational Frequency (GHz)***	***Leakage Power (μW)***	***Dynamic Power (μW)***
Sutherland	0.865	0.17	74.41
van Berkel	1.148	0.13	74.33
Martin's Weak feedback	0.808	0.28	119.20

As TABLE IX shows, the conducted experiment confirms the results obtained through the electrical characterization of the C-elements. The van Berkel implementation presents the lowest leakage power consumption, while its dynamic power consumption is equivalent to that of the Sutherland C-element. The weak feedback topology presents higher dynamic and leakage power, as expected. Power figures enforce the statement that the weak feedback is the most power consuming and Sutherland's and van Berkel's present similar power consumption figures.

It would be expected that at least two of the rings, the ones composed by the Sutherland and van Berkel topologies, would present equivalent operating frequency. However, the ring composed of van Berkel C-elements operates roughly 32% and 42% faster than the ring composedby Sutherland and weak feedback C-elements, respectively. As TABLE IX shows, the sum of the pin capacitances of the Sutherland C-element is 8.975fF, while for van Berkel topology it is only 6.194fF. In other words, each cell of the ring in Sutherland implementations, except the one that drives the NAND, must drive a load roughly 44% bigger than that of the van Berkel C-element ring. Both implementations showed to be equivalently sensitive to output load variations. However, these variations interfere in the transition time of their output, which feeds the next cell in the ring. Thus, the slope in the input of the next cell increases. In this case, the resulting input slope generated in the inputs of each Sutherland C-element, after the circuit stabilizes its oscillating frequency, is roughly 0.06ns, as Figure 12 shows. Considering that the Sutherland C-element is more sensitive to input slope variations, it is clear why the rings operate at different frequencies ranges.

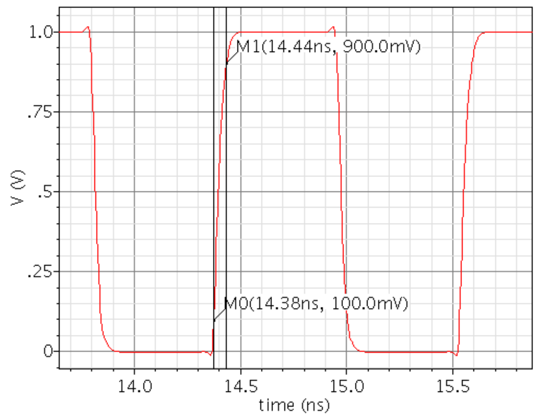


Figure 12 – Input slope of a single cell of an oscillator ring composed by Sutherland C-Elements.

For the weak feedback C-element, the operational frequency range is even worse. This is due to the fact that the topology presents not only higher input capacitance, which contributes for an elevated slope in the input of each cell of the ring, but is also much more sensitive to input slope variations than the Sutherland C-element. Its performance would be yet worse if, for instance, each C-element was required to drive a load bigger than 0.015pF. See Figure 10, where the delay of the weak feedback implementation starts to get worse than the other two. In this case, the sum of its input pin load that each cell (except the one that drives the NAND) must drive, is 11.482fF.

A more realistic comparison of the impact of the C-elements implementation was conducted through the design of a 32-bit RSA cryptographic core. The circuit was described in the Balsa language and synthesized through the Teak System. Teak automatically maps the Balsa description into a specific set of cells, a group of C-elements with different functionalities, generating asynchronous QDI circuits. Three versions of the asynchronous RSA cryptographic core were generated using Teak, each one employing exclusively one distinct C-element type implementation on its schematic. The choice for the RSA function was due to the fact that its algorithm employs arithmetic operations as well as control functions. Arithmetic operations are implemented through delay insensitive minterm synthesis (DIMS) logic, which is basically constructed with C-elements. In this way, the impact of the choice of C-element on the circuit can be efficiently evaluated.

The total number of standard-cells employed in all designs, without taking into account physical cells, was 57,168. The circuits had the same number of logic cells due to the fact that the only difference between them is the choice of C-element implementation and Teak does not optimize cells dimensions and employs a well defined set of cells for each handshake component. From the total cells, 22,063 were C-elements. For these examples, roughly 40% of the required logical standard cells were C-elements.

TABLE X shows the physical characteristics of the generated RSA cryptographic cores, obtained after place and route. The design implemented with weak feedback C-elements requires less silicon area, while the ones implemented with van Berkel and Sutherland C-elements were the largest. The area overhead imposed by both in comparison with the weak feedback is roughly 12% and 7%, respectively. Therefore, the weak feedback C-element is the most efficient implementation for high density designs and the van Berkel C-element is the most area and wire consuming implementation. These results are in agreement with the information obtained at layout level.

TABLE X – Area and wire results for the three asynchronous RSA cryptographic core implementations after place and route.

C-element Implementation	*Sutherland*	*van Berkel*	*Weak feedback*
Number of Standard Cells	92,922	94,015	91,276
Total cell area (mm²)	0.295	0.311	0.276
Cell area - physical cells (mm²)	0.244	0.258	0.228
C-elements cell area (mm²)	0.161	0.175	0.145
Total wire length (mm)	648.208	703.694	616.176
Average wire length (µm)	10.746	11.665	10.215

The RSA netlists' delay of the paths generated after place and route were annotated and served as input to a set of simulations. These comprised multiple cryptographic operations for each netlist, collecting performance results. Employed operational conditions were 25°C, 1V supply for a typical fabrication process corner. The average delay to perform a cryptographic operation for the weak feedback, the Sutherland and the van Berkel C-element based implementations were 104.311µs, 83.96µs and 73.241µs, respectively. These results are in agreement with the information obtained in the simulation of an oscillator ring. The van Berkel implementation presented higher operating speed, due to the fact that it is the less sensitive topology to input slope and output load variations and Teak synthesis is not able to optimize dimensioning of the selected standard cells. In other words, every standard cell employed in the circuit has the same output driving strength and input capacitance, some of these ending up overloaded. This is a limitation of the tool, which leads to slower designs mostly when employing Sutherland or weak feedback C-elements, since these present higher delays for high output loads and input slopes.

From the simulations, the switching activity in the nets of each circuit was annotated for a period of 3ms and served as input to evaluate power consumption. TABLE XI shows the information obtained for the three netlists. Employing Sutherland or van Berkel C-elements generated circuits with similar power consumption. Comparing these implementations, the latter consumed less leakage power, roughly 5%, while the former presented lower dynamic power consumption, roughly 4%. The circuit generated with Sutherland C-elements presents slightly less total power consumption. This is due to the fact that the power consumed while the circuit is quiescent represents a portion of the total power smaller than the dynamic power consumption. Notably, weak feedback was the less power efficient implementation.

TABLE XI – Power consumption of the three asynchronous RSA cryptographic core implementations.

C-element Implementation	***Sutherland***	***van Berkel***	***Weak feedback***
Internal Power (mW)	1.878	2.161	4.361
Switching Power (mW)	1.581	1.433	1.342
Leakage Power (mW)	1.729	1.639	2.162
Total Power (mW)	5.188	5.233	7.865

These results are in agreement with those obtained at layout level and in the simulation of an oscillator ring, except for the dynamic power consumption of the Sutherland C-element. In the first case study, this present dynamic power consumption worse than the van-Berkel C-element. However, in that case, each cell was driving a single 2-input cell, while in the circuit generated by Teak, cells were required to drive multiple nets and, consequently, higher loads. In this scenery, the van Berkel dynamic power efficiency was compromised.

Figure 13 shows details the power consumption for the C-elements from the total power consumed by the placed and routed netlists. The total power consumed by Sutherland's, weak feedback and van Berkel's C-elements, in their respective netlists, was 2.803mW (54%), 5.722mW (73%) and 2.815mW (54%), respectively. These results show that in realistic applications, the reason for the Sutherland C-element to consume less power than the van Berkel is because the internal power consumed by the latter is more significant. One aspect that worsens internal power consumption is the amount of transistors in short circuit when switching the van Berkel C-element output logical value. Moreover, the bigger the input slope is, the bigger is the period of time the transistors are in short circuit when switching the output of the C-element.

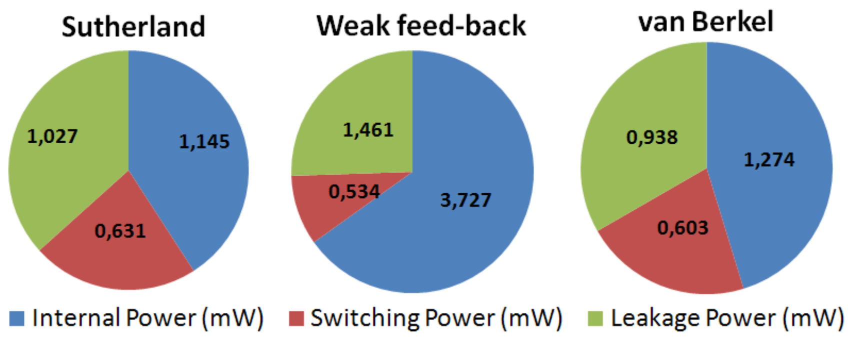


Figure 13 – C-Elements power consumption for each asynchronous RSA cryptographic core implementation.

The results obtained show that previous findings for the electrical behavior of C-elements must be reevaluated. The use of a realistic asynchronous synthesis tool like Teak allowed to evaluate C-elements' behavior in current state of the art situations. The inability of the tool to optimize standard cell selection based on current drive capacity indicates the need of enhanced synthesis tools for asynchronous circuits.

In summary, albeit the van Berkel C-element appears as the lowest static power consuming implementation, it has been shown to consume more dynamic power than the Sutherland C-element for bigger input slopes. Moreover, the dynamic power consumption represents a portion of the total power bigger than the static power consumption. In this way, the Sutherland C-element appears as the most indicated for low power designs, regardless of input slope variations. The weak feedback presented the worst propagation delay, regardless of output load or input slope variations. Moreover, the van Berkel and the Sutherland C-elements proved to be equally robust to output load variations. However, the Sutherland implementation presented higher propagation delay for high input slopes. Therefore, the van Berkel C-element appears as the most speed-efficient implementation. The results on required area for each C-element, showed that the van Berkel implementation is the most silicon area consuming, while the weak feedback is the most area-efficient. Hence, the latter is the most suitable for high density designs. Finally, the work described here shows that the choice of C-element type in a DSM asynchronous design is a triple (speed/area/power) tradeoff.

4 NON-SYNCHRONOUS NOC ROUTERS

Current VLSI technologies offer means to develop complex systems-on-chip (SoCs). However, to take advantage of such devices in the production of advanced products while obeying time to market and physical constraints, designers must revise traditional design methods [HEN03]. Advanced mobile applications, one of the mainstream product niches today, assume as critical two physical design parameters: propagation delay in long wires, and power dissipation. These deserve special attention, since they are expected to rule the advance of future design processes [ITR08]. Another important parameter, strongly related to power dissipation is energy consumption.

IP Core reuse is indicated as one of the main techniques to reduce SoC development time [SAL06]. However, efficient hardware reuse implies heterogeneous systems where each module works at its optimum operating frequency, achieves ideal latency and throughput values, etc. Ensuring such constraints for each and every processing module in a modern SoC is a daunting task. The main assumption of traditional synchronous design, i.e. the adoption of a unique clock signal for the whole system, is rapidly becoming a major limiting factor. This happens because of several factors: (i) constraining all modules to work at a single clock frequency leads to suboptimal designs; (ii) the clock distribution is responsible for a significant part of the whole power dissipation in a SoC; (iii) the problem of controlling clock skew in a chip to implement using DSM technologies that needs to work at a high operating frequency is becoming intractable.

The use of asynchronous techniques can be a solution to the above mentioned problems. Nevertheless, the design of fully asynchronous systems suffers from the lack of a widely adopted design technique like the RTL model for synchronous systems. Also, there is a lack of mature support CAD tools. A strategy that fits the gap between synchronous and asynchronous design is the adoption of globally asynchronous and locally synchronous (GALS) design techniques [CHA84]. The adoption of GALS keeps the use of synchronous methods and CAD tools inside synchronous modules design and leaves the synchronization problem restricted to the interface among modules. In this context, the use of NoCs with support to GALS communication comes out as a trend [BAI02] [BJE06] [KIM05] [ROS05]. NOCs provide more scalability than traditional shared busses and allow more communication parallelism as well.

GALS design partitions the clock tree, reducing clock buffering needs and clock tree size and spread. These actions have a potential to significantly reduce the chip overall power consumption [CHA84]. Also, adoption of multiple clock domains still enables techniques for power reduction, such as dynamic frequency and voltage scaling (DFS, DVS). Although these may affect system performance, they also reduce energy consumption. In this scenario, non-synchronous NoCs present an elegant solution to cope with current technologies challenges. In the context of this work, the Author took part in the design of five different non-synchronous NoC routers, three of which are fully asynchronous.

4.1 Hermes-G and Hermes-GLP

Non-synchronous communication assumes that (synchronous) communicating modules do not share a same clock. They either work with unrelated or simply non-identical clocks (due to frequency, duty-cycle or phase differences). Two modules communicating asynchronously need that a synchronizing interface be interposed between them, what is called here an *asynchronous interface*. Asynchronous interfaces are critical components for GALS systems performance [CHA84].

Several NoC proposals to support the GALS paradigm exist in current literature. These proposals can be classified according to the router implementation or according to the relative position of asynchronous interfaces w.r.t. routers and IP Cores. Router implementation can be synchronous or asynchronous. Communication between routers or between a router and an intellectual property (IP) Core can also be synchronous or asynchronous. TABLE XII shows the possible combinations of router type, asynchronous interface position and NoC and SoC resulting types. In NoCs employing synchronous routers, each router may have its own independent clock or the clock signal can be shared by all routers. In this case router and IP Core form either an asynchronous [KIM05] or mesochronous subsystem (in a mesochronous system or subsystem, a single clock signal is used, but the phase of this signal is not guaranteed to be the same or within defined bounds in every point of the system). NoCs implemented with asynchronous routers are another possibility to develop GALS systems [BAI02] [ROS05]. Since such routers have no clock signal, the dynamic power dissipation can be reduced. However, since asynchronous circuits may present important area overheads, its static power may increase the power dissipated in the NOC.

TABLE XII – NoC-SoC classification as a function of router type and asynchronous interface positions. R=Router, IP=IP Core.

R	R to R Inteface	R to IP Interface	NoC	SoC
Sync	Sync	Sync	Sync	Sync
Sync	Sync	Async	Sync	GALS
Sync	Async	Sync	GALS	GALS
Sync	Async	Async	GALS	GALS
Async	Async	Async	Async	GALS

Kim et al. [KIM05] present a synchronous NoC with asynchronous interfaces between the router and its local IP Core(s). The interface consists of a bisynchronous FIFO which allows a reading operation with a clock distinct from that used in writing [CUM02]. This FIFO allows writing and reading operations to occur at each respective clock cycle. The advantage of the approach is to decouple IP Core and router operating frequencies, but the clock tree can still spread along the whole chip.

The DSPIN NoC [PAN06] employs synchronous routers with bisynchronous FIFOs for the communication between each pair of routers and between a router and an IP Core. In the first case, the FIFO is designed for mesochronous communication, while in the second case it is designed to allow communication between unrelated clock domains. The Authors state that their design of a mesochronous FIFO presents latency between one and two clock cycles, while the other designed FIFO has latency between two and three clock cycles.

Bjerregaard et al. [BJE07] present the architecture of a mesochronous NoC. The asynchronous interface employs a 4-phase handshake protocol. The adopted NoC topology is a tree, and the clock signal distribution follows the same paths as the communication channels between routers. In this way, clock skew can be deterministically computed and delay elements inserted on data signals respecting the skew. Thus, setup and hold times are respected everywhere in the system. The approach presents two disadvantages: first, NoC reuse is difficult and second, the use of delay elements makes the design sensitive to fabrication Process-Voltage-Temperature (PVT) variations.

Although the above NoCs enable the construction of GALS systems, their proposal do not comprise any power control mechanisms on the NoC architecture. Hsu et al. [HSU05] on the other hand present a frequency scaling low power mechanism (FSLP), to control the power dissipation of a SoC that uses a NoC as communication architecture. However, the power control mechanism is applied to IP Cores connected to the NoC and not to routers. The scheme controls the frequency of the IP Core depending on measured and required communication rates. Simunic et al. [SIM04] also propose a control system to determine the operating frequency of IP Cores connected to a NoC. The system uses power and QoS requirements to determine the best frequency for IP Cores.

Worm et al. [WOR05] present a self calibrating mechanism to control the DVS in an NoC aiming a tolerable bit error rate. The proposed mechanism uses a module to detect the error and request retransmission. This mechanism can generate a large latency, due to retransmission, thus violating the communication QoS requirements. Ogras et al. [OGR07] propose a method for partitioning a NoC-based GALS SoC. The objective is to define voltage and frequency islands (VFIs). The flow creates clusters composed by routers and IPs operating at a same frequency and voltage. Within each island, power can be reduced and controlled through the use of DVS. The communication between islands uses interfaces to cross the frequency-voltage domain. The islands are defined and fixed at design time.

One work where the Author took part, presented partially in [PON08], stands off from the above mentioned due to the capacity to create frequency islands at run time based on message priorities. The power control mechanism is quite simple and implies low area overhead.

To obtain the Hermes-G router, the original Hermes router had its input buffer structure modified, to allow the routing to provide communication among modules operating at distinct frequencies. The bisynchronous FIFO proposed in [CUM02] substitutes the original Hermes input buffer FIFO which is synchronous. The ability to enable data writing and reading in one single clock cycle motivated the choice of a bisynchronous FIFO.

Figure 14 describes the structure of the bisynchronous FIFO. The FIFO implementation uses two pointers, one defining the next write position and another defining the next read position. The FIFO state is either full or empty when both pointers refer to the same address. Thus, it is necessary to compare pointers. Although this procedure is trivial in synchronous circuits, it implies some complexity in multi-clock devices such as a bisynchronous FIFO, because each pointer is generated by a different clock. The usual solution to solve this problem is to transfer and synchronize the write pointer value with the receiver clock, which generates the **`empty`** signal, and mutatis mutandis for the **`full`** signal.

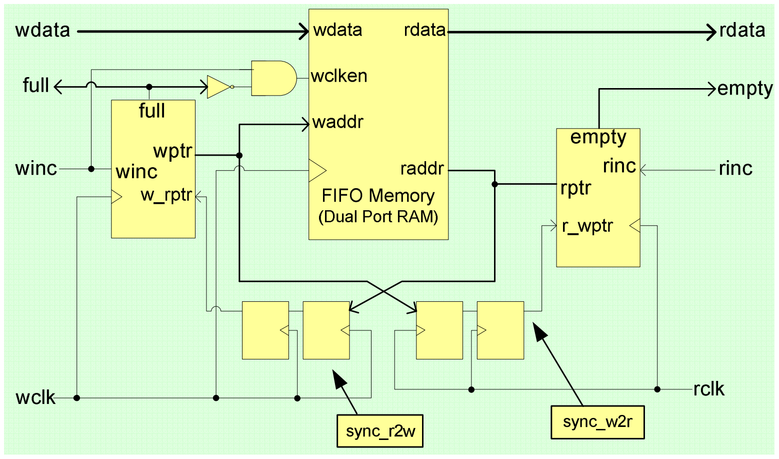


Figure 14 – Structure of a by-synchronous FIFO.

Pointer exchange can be accomplished using synchronizers or clock stretching [PON07a]. However, adding a handshake protocol to control pointer exchange implies additional latency in the FIFO. Cummings [CUM02] presents an elegant solution to pointer exchange with synchronizers. Addresses are translated to Gray code, which guarantees that consecutive addresses are at a Hamming distance of 1. In this way, the metastability problem is confined to a single bit and synchronizers can be employed without handshake. In Figure 14, **`wptr`** represents the write pointer and **`rptr`** the read pointer and they are used for pointer exchange. In this FIFO, if synchronization fails the only consequence is the premature generation of **`empty`** or **`full`** signals, but there is no possible data loss or corruption.

The Hermes-G NoC may have a bisynchronous FIFO in each router input port or just in selected ports. This enables the definition of arbitrary clock domains containing any number of connected routers and IPs. To let the IP core connected to a router work with a clock distinct from that of its router, a bisynchronous FIFO must also be inserted in the IP core Network Interface input port.

The Hermes-GLP router assumes every router interface to be asynchronous and also allows the use of clock gating and dynamic frequency scaling. The clock gating mechanism consists in disabling the clock of idle modules of the system [PED05]. This mechanism can be applied at different granularities. Hermes-GLP applies the mechanism at the router level. A router is considered idle when all of its ports are idle, meaning that no port has any data to transmit. To implement clock gating, a clock control module was added in the router design. This module receives a signal from each input port and updates the respective port state accordingly. When all ports report no data to transmit, the clock control disables the router clock input. The router remains in this state until a write operation is signaled in one or more of its input port, setting the port to the transmitting state. This system is possible due to the fact that the each port has its own write clock signal, but all ports share the same read clock signal, used internally by the router.

As for the DFS mechanism, it was implemented using a simple glitch-free dynamic clock switching between two or more clocks. More elaborated clock scaling techniques as presented e.g. by Pontikakis et al. in [PON07b]. The use of DFS allows the self calibration of the router operating frequency based on expressed communication requirements. Routers involved in a high priority communication operate at a higher frequency than routers involved in low priority communication.

To enable dynamic router configuration, each packet carries its own priority in a sideband signal. This signal is routed along with the packet. Each router receives a set of **n** priority signals, where **n** is the total number of ports in each router, and computes the largest indicated priority, selecting the best operation frequency based on the result. In this way, latency constraints for all packets is more easily achievable, while providing a simple power control mechanism that requires small area overhead in routers.

Figure 15 illustrates a Hermes-GLP clock control module structure. Signals **sel_clk_in** at input ports are used to control the DFS and are each routed to the output signal **sel_clk_out** of some output port along with the packet. Signals **Port_State** indicate the state of each port and are used to control clock gating. These signals are used either to decide which **sel_clock_in** is used to choose the current router clock, or to perform clock gating. Only active ports are considered to determine the router clock frequency.

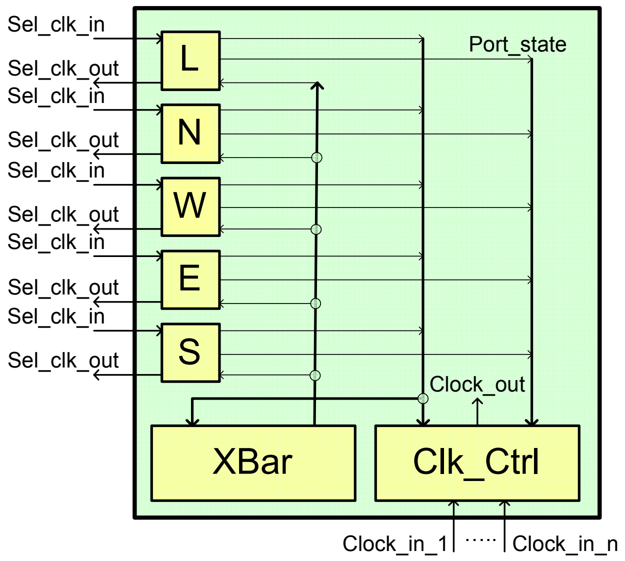


Figure 15 – Structure of the Hermes-GLP router, highlighting the clock control module structures.

Figure 16 shows an example of dynamic clock determination for two concurrent communication flows with different priorities. In the first step (a), the NoC starts with a low priority flow established. In (b), a higher priority flow starts, forcing the routers in its path to work at a higher frequency. Routers that belong to both communication paths are configured to the higher frequency, to meet the most stringent flow requirement. In (c), the maximum frequency flow ended first, which sets router 11 back to a lower frequency and stops routers 21 an 01. When the first flow ends in (d) all routers reach an idle state and all clock routers are stopped.

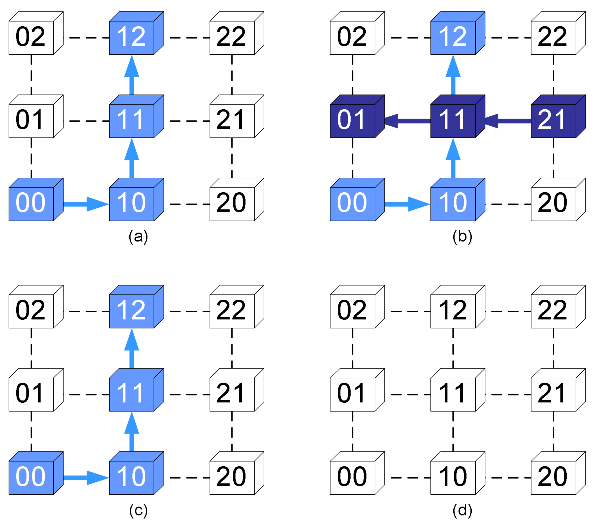


Figure 16 – Example of dynamic clock determination and clock gating. The darker the router, the higher is its operating frequency. White routers are under clock gating.

Test scenarios were employed to validate the functionality of the Hermes-GLP NoC. The test set allows evaluating the potential to reduce the router activation rate, a parameter that can be related to power dissipation. The employed simulation scenarios are presented in TABLE XIII, where a traffic pattern comprises always two pairs of IP Cores exchanging messages according to two flows, (a) and (b). Each distinct traffic pattern corresponds to a different mapping of IP Cores into the NoC. The objective of these scenarios is to illustrate the dependence of router activation rate on the IP core mapping in the NoC.

TABLE XIII – Simulated producer consumer pairs

Traffic	Sender Address	Receiver Address	Sender Frequency (MHz)	Receiver Frequency (MHz)	Communication Priority
T1	02(a) – 02(b)	20(a) – 22(b)	200	170	High
T2	22(a) – 01(b)	00(a) – 21(b)	120	150	High
T3	12(a) – 20(b)	21(a) - 00(b)	90	70	Low
T4	20(a) – 22(b)	12(a) – 20(b)	170	90	Low
T5	01(a) – 12(b)	11(a) – 10(b)	50	180	Low
T6	21(a) – 00(b)	02(a) – 02(b)	70	200	Low

Routers in this case study can switch between one of two clock sources, 200 MHz and 100 MHz. Message priorities were defined as a function of the operating frequency of the IP cores involved in communication. Thus, if the min (Fs, Fr) > 100 the priority is set to High, else it is set to Low. A High priority flow makes the router work at 200 MHz, while a Low priority flow makes the router work at 100 MHz, if there is no other High priority flow passing through the same router.

The activation rate of a router is described as a function of its state. If at a time `t` router `x` works at 200 MHz, then its activation rate is expressed by:

$$A_x(t) = F_{op}(t) / F_{\max} \qquad \textbf{Equation (3)}$$

The average activation rate of each router is computed during simulation by SystemC code that extracts operating state values for each router at each 1 ns period. These data are used to verify the average activation rate of each router, employing Equation 4:

$$A_{AV}(n) = \frac{\sum_{i=0}^{n} A_x(i)}{n} \quad \textbf{Equation (4)}$$

Here, **n** represents the number of times a SystemC monitor code extracted the router state. The average activation rate of the NoC can be described as the average of the average rates for each router, as shown in Equation 5.

$$A_{NoC} = \frac{\sum_{k=1}^{r} A_{av}(k)}{r} \quad \textbf{Equation (5)}$$

Figure 17 depicts example values for the average activation rate of each router for flow (a) of TABLE XIII. The results show that even when the communication rate presents a high value, Hermes-GLP displays a significant reduction in activation rate when compared to the fixed 100% rate Hermes-G router.

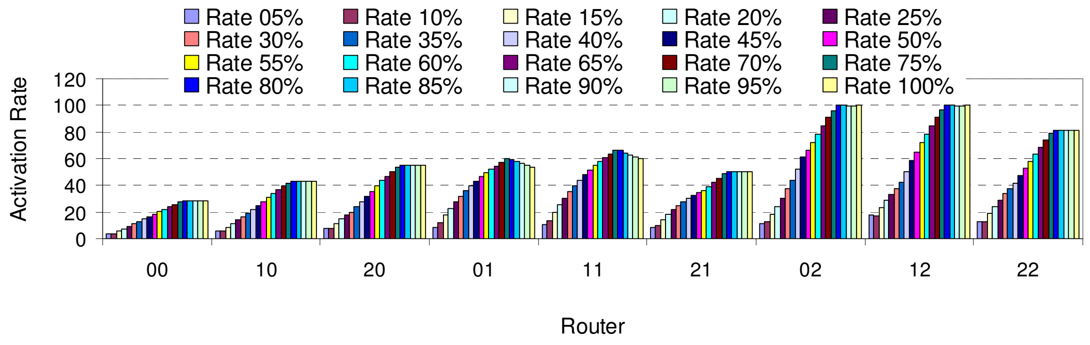


Figure 17 – Average activation rate of each router for flow (a) of TABLE XIII, with insertion rates varying between 5% and 100%.

Additionally, Figure 18 shows a comparison of the average activation rate on Hermes-GLP for two traffic scenarios presented on TABLE XIII. In this Figure it is possible to note that the activation rate can be significantly influenced by the arrangement of the IP Cores in the communication infrastructure.

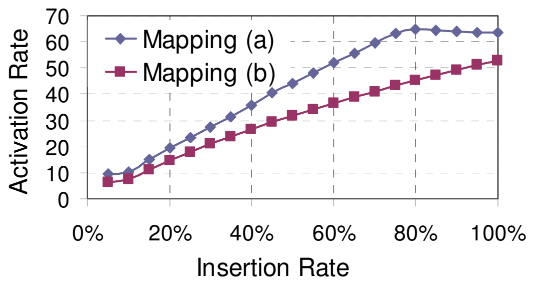


Figure 18 – Activation rate variation of Hermes-GLP NoC as a function of mapping and traffic insertion rate.

Figure 19 depicts the comparison of the maximum packet latency for flow (a) of TABLE XIII in the Hermes-G and Hermes-GLP. It is possible to notice that the Hermes-GLP does not introduce significant additional latency when compared to Hermes-G.

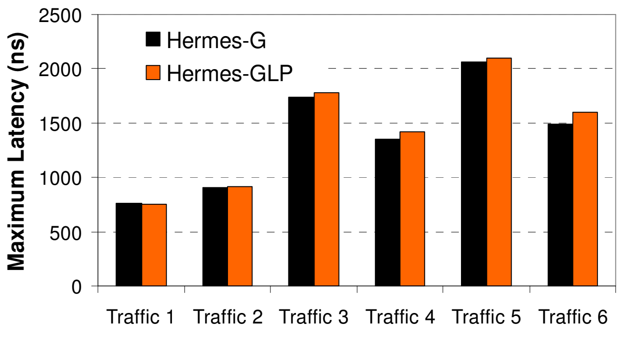


Figure 19 – Comparing packet maximum latency for Hermes-G and Hermes-GLP NoCs.

Hermes-GLP is a NoC developed to reduce power in SoCs. It is based on a simple DFS mechanism that allows significant power dissipation reduction with small latency penalty. Simulation results are encouraging that this strategy is useful and leads to low overhead not only in latency, but in area as well. One important point is that to ensure the robustness to the proposed DFS scheme, it is necessary to guarantee that the clock switching process is glitch free. The proposed NoC was prototyped in FPGAs. Moreover, ongoing work includes the development of versions of Hermes-GLP for ASIC implementation in CMOS technologies. One of the main points underlying the Hermes-GLP is the process of generating clocks to feed the NoC routers. Another ongoing work by other members of the Author's research group includes the development of local clock generation schemes that are sufficiently simple and powerful.

4.2 Hermes-A and Hermes-AA

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 XIV summarizes the main features for each of these, with the last row of the Table presenting the features for the Hermes-A router and NoC, proposed in [PON10a], work in which the Author took part. TABLE XIV 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. More efficient implementations of asynchronous devices in FPGAs than those cited in these works exist, as described in [PAN06]. These rely on use of FPGA layout and timing control tools to create asynchronous devices as FPGA hard macros. These are reasonably compact and may respect tight timing constraints. However, so far these have not been used for NoCs.

The remaining five NoCs/routers in TABLE XIV (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, 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 XIV – 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 [BAI02]	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 [FEL04]	2D Mesh 4 3GS/1BE VCs	XY / wormhole / credit-based	N.A.	QDI	1-of-4 DI / 8-bit flits	Simulation only
Nexus [LIN04]	Single 16x16 Crossbar	Source / BOP-EOP	A2S, S2A	QDI / 1-clock converters	1-of-4 DI / 36-bit phits	130nm, 780Gbits/s, ASIC
MANGO [BJE07]	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 [DOB09]	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. [QUA05]	Crossbar or Octagon	N.A.	Self-timed FIFOs	QDI	N.A.	FPGA, 56 Mflits/s
ANoC [BEI05]	2D Mesh (Irreg/Reg) / 2 VCs	Source / odd-even / wormhole	A2S, S2A FIFOs	QDI	34-bit flits	65nm, 550Mflits/s, ASIC
As. Proteo [WAN06]	Bidirectional Ring	Oblivious	OCP	QDI / 4-phase dual-rail	32-bit flits	FPGA, 202 Kbits/s
RasP [HOL06]	Framework / point-to-point (Irreg/Reg)	Source / bit serial	Ad hoc	Dual-rail	Point-to-point pipelined serial links	180nm, 700Mbits/s Simulation
ASPIN [SHE08]	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	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 (guaranteed Services or 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 [BJE07]. 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 Section.

4.2.1 Hermes-A

Unlike most other previously proposed 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. Figure 20 depicts the Hermes-A input port structure as a simplified asynchronous data-flow diagram [HSU05]. 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 20 two wires represent each bit. Thus, a 10-bit path is in fact a 20-wire bus.

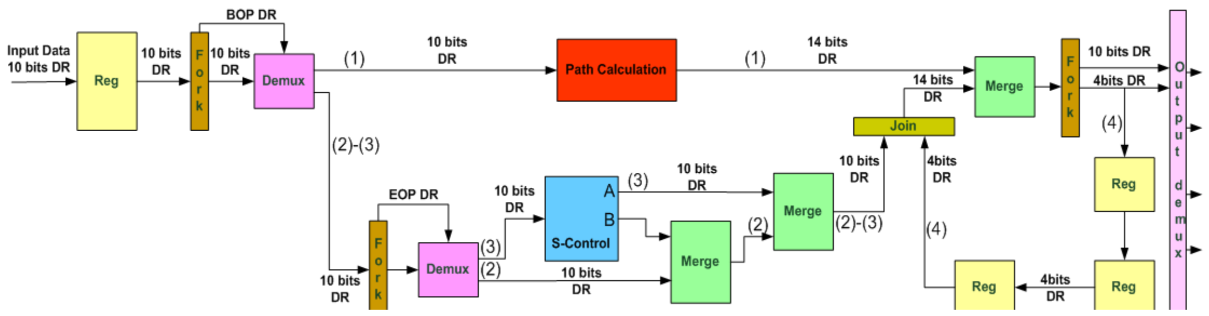


Figure 20 – Hermes-A router input port architecture. All paths employ dual-rail encoding.

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 20 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 [CRI10]. 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 a 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.

After computing the output port where to send the incoming flit, the rightmost module in Figure 20 (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 paragraphs cover the behavior of the Path Calculation and S-Control modules.

The basic route computation architecture is depicted in Figure 21. 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 dual-rail 4-bit packet destination code. It points the output port to use.

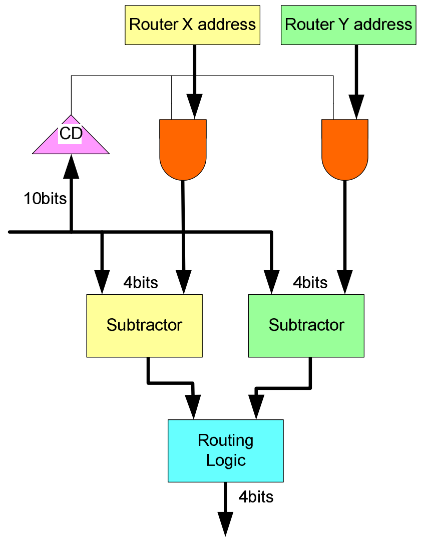


Figure 21 – Hermes-A path calculation circuit.

When the last flit of a packet is received (EOP=1), it is directed to the S-Control module. The S-Control protocol description appears in Figure 22.

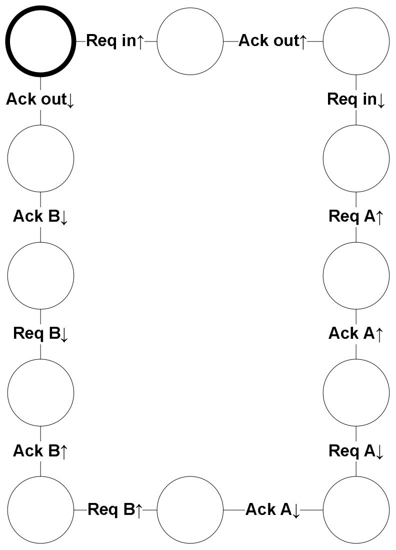


Figure 22 – State machine for the S-control module.

The function of this module is to send the last flit through the output marked A in Figure 20, 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 by 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 20). 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 22.

In the Hermes-A router each output port receives four data flows. For instance, Figure 23 shows the Local output port structure that receives data from input ports North, South, East and West.

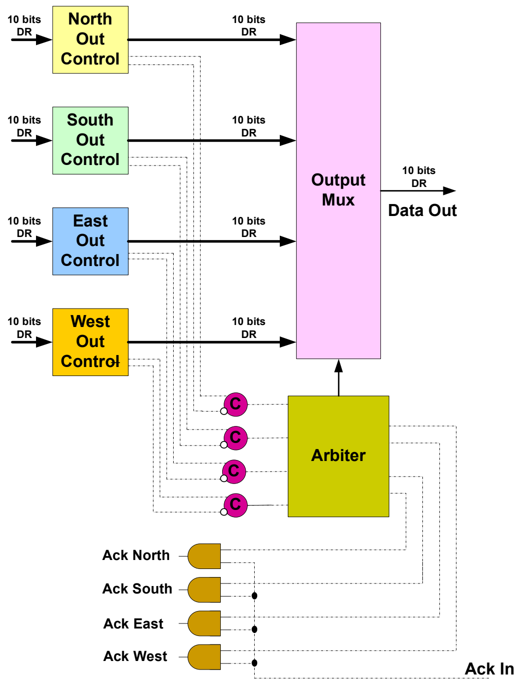


Figure 23 – Local output port structure. Dashed lines represent actual wires. Solid lines represent dual-rail encoded lines.

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 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 24 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.

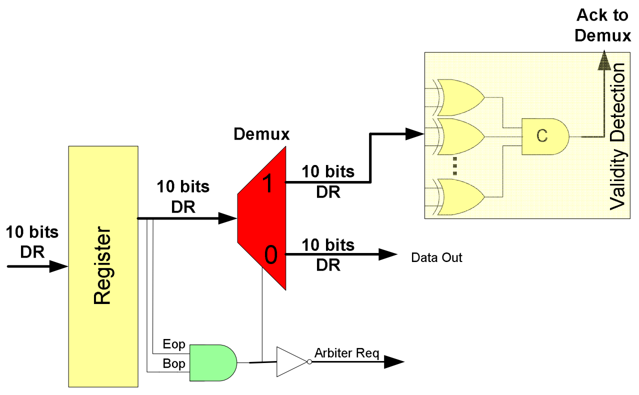


Figure 24 – Output control structure. All paths employ dual-rail encoding.

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 a 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 throughput higher than traditional synchronizers.

The synchronization mechanism adopted here is based on the SCAFFI asynchronous interface [PAN06]. 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 25. More details on this interface are available in reference [PAN06].

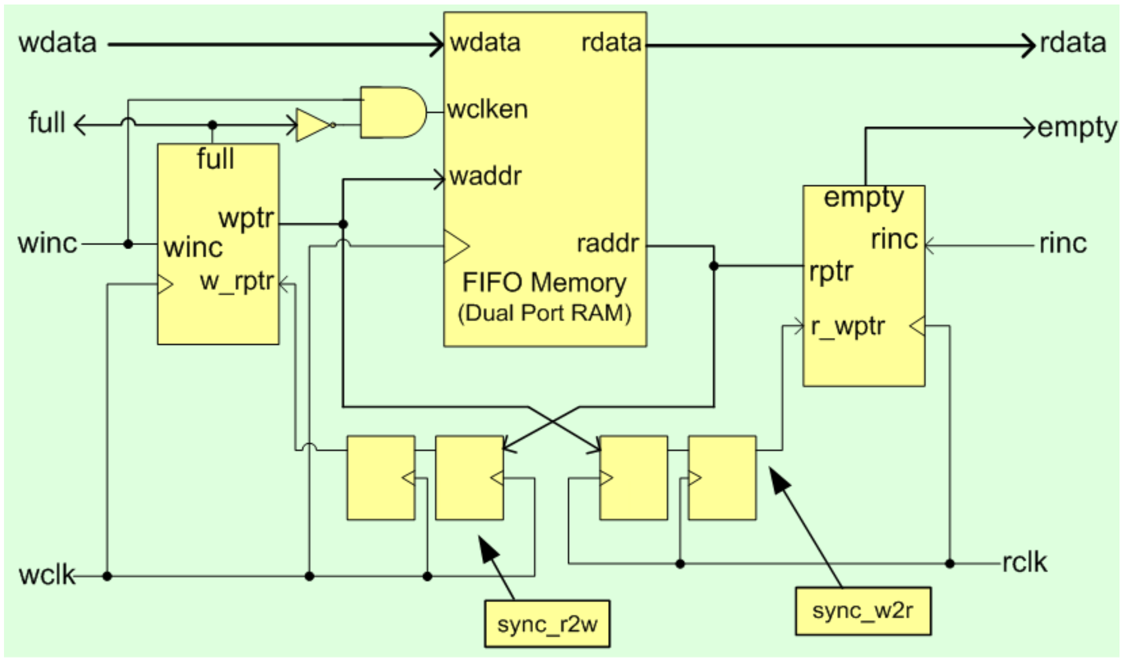


Figure 25 – SCAFFI network interface between a Hermes-A router and a synchronous PE. The interface employs clock stretching techniques to avoid metastability. The stretcher circuits are not represented in the picture.

Since traditional design kits do not usually contain asynchronous components, the Hermes-A ASIC implementation started with the implementation of an custom asynchronous digital cell library that later became the seed of the ASCEnD library previously described in this work. The library includes several versions of C-elements, metastability filters and control circuits, like sequencers. The first version of the asynchronous library used the XFab 180nm design rules and included liberty timing files (.lib), abstract views (.lef) and Verilog models using UDP primitives to enable timing annotated simulations. This 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 cell instances characteristics.

The results presented in the TABLE XV refer to the XFab 180nm ASIC implementation of the Hermes-A router. The operating conditions are 25°C, 1.8 Volts. Also, the library 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 XV 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 XV – ASIC implementation results for an 180nm XFab technology.

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

4.2.2 Hermes-AA

The Hermes-A router demonstrates that asynchronous circuits are useful as communication architecture for a high performance complex GALS SoCs. Another work [PON10b], proposed the Hermes-AA router. This new router is a modification of Hermes-A which implements an adaptive routing scheme. In this work, the target technology was the 65nm STMicroelectronics general purpose standard Vt using the ASCEnD-ST65 library.

Area and power results are taken from physical synthesis after RC extraction. The timing results are obtained from timing simulation. The power analysis was done based on the switching activity computed by simulation. The operating conditions used were 25°C, 1 Volt.

Figure 26 describes the relationship between throughput of a link of the Hermes-AA router and the IP frequency.

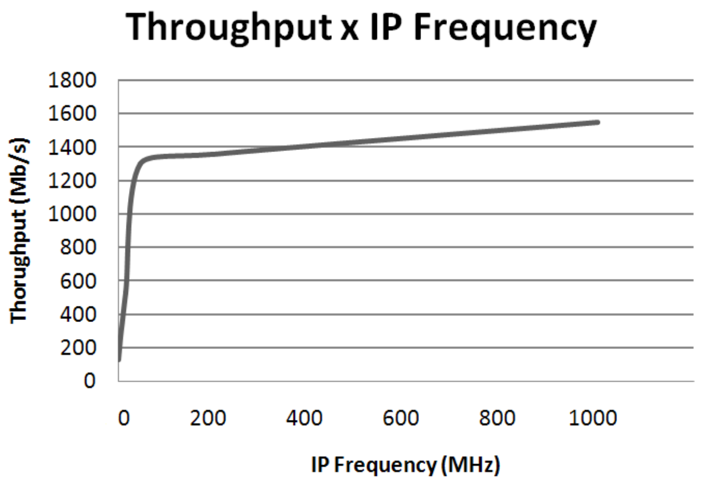


Figure 26 – Eight bit router throughput versus IP Frequency.

For the 8-bit router it was possible to extract the maximum throughput that a router link can de-liver, 1.55 Gbits/s. Since a single router can support up to five transmissions simultaneously, the maximum router throughput is 7.75Gb/s. This value can be improved with using asynchronous FIFOs at the input ports. Simulation results show that when the router operates with a FIFO, the throughput can reach up to 6.3Gb/s per port. It is important to note that the router throughput goes back to the saturation value (1.55 Gbits/s) when FIFOs are full. For the employed operating conditions the Input Port latency is 3.709ns and the Output Port latency is 1.351ns. Latency results were obtained when the router is empty. Thus it is a lower bound.

TABLE XVI compares area results of the Hermes-AA router (using a West First routing algorithm router) with an implementation using the XY routing algorithm, the same router published in [PON10a] but synthesized to the same 65nm technology. It shows that the use of the adaptive algorithm does not impact router area significantly.

TABLE XVI – West first and XY Hermes-AA routers comparison.

Algorithm	Standard Cell Area	Total Area
WF	76455 μm^2	116034μm^2
XY	75133 μm^2	114456 μm^2

Figure 27 shows a first traffic scenario built to display the benefits of using an adaptive algorithm. In this scenario, the Local and West ports send 9-flit packets in burst to two IPs located to the right and above the current router.

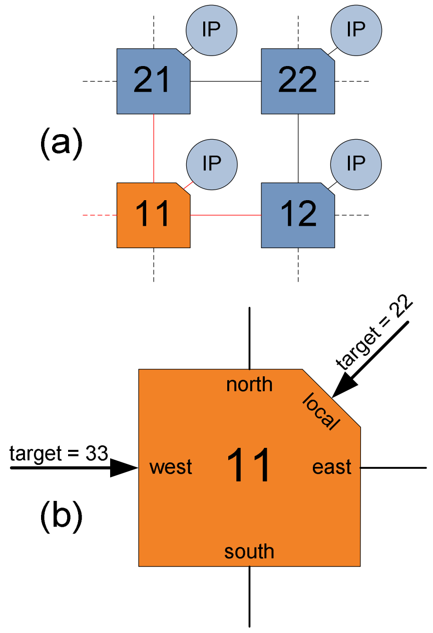


Figure 27 – Scenario for evaluating the Hermes-AA router throughput variation with the IP Frequency.

TABLE XVII shows that the West First algorithm presents a gain of about 71% over the XY in throughput.

TABLE XVII – West first and XY throughput comparison.

	XY	WF
Throughput	10Mpackets/s	17Mpackets/s

Power characterization was performed using a traffic scenario where all input and output ports are working in the saturation region of the router. Figure 28 shows the power dissipation of each module for the 8-bit phit Hermes-AA router.

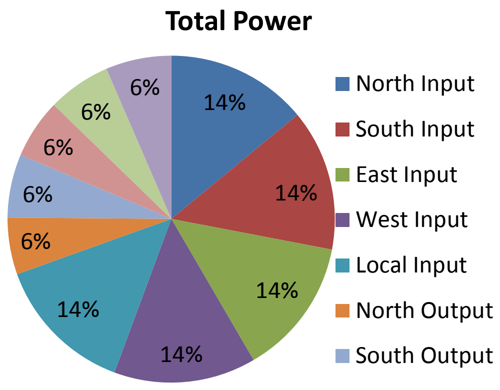


Figure 28 – Eight bit router modules proportional power.

Finally, Figure 29 evaluates the distinct components of power for the same scenario. For this particular scenario, the worst component is the internal power dissipation. However, the main concern refers to leakage power dissipation since in normal operation, with less congestion, the switching and internal power of the router should reduce but the leakage power dissipation will remain the same.

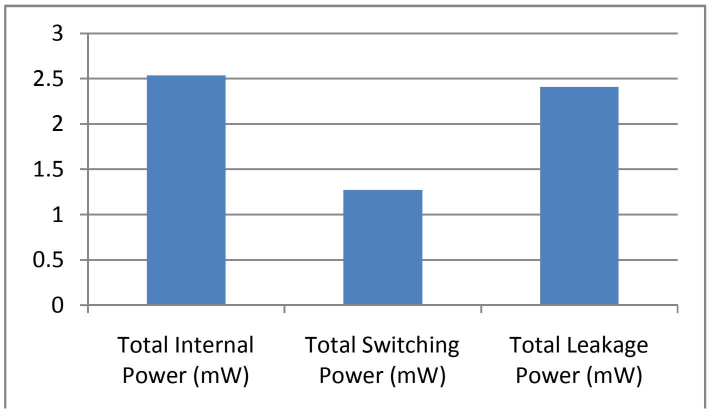


Figure 29 – Router Power characterization.

4.3 BaBaNoC

Most works the Author found on literature report the use of synchronous style design flows and tools for implementing asynchronous routers. This ends up generating a greater workload, where the focus of the work becomes the adaptation of the circuit, in order to guarantee asynchronous characteristics using tools designed for synchronous systems. Besides, asynchronous modules are handcrafted. This means that to implement handshake components, asynchronous control logic or even basic asynchronous components such as C-elements, manual design is omnipresent.

The Hermes-AA demonstrated that adapting typical tools and standard-cell libraries to design asynchronous circuits is a challenging task. Hermes-AA functional description took months to be implemented and validated. Moreover, its logic synthesis capitalized on much manual labor to generate a functional netlist. In contrast, BaBaRouter, proposed in [MOR12b], was described and validated in the Balsa Framework in less than one week and Balsa automatically generated a functional netlist. Both designs require special standard cells in their synthesis, which are available in ASCEnD. The complexity of physical synthesis for both Hermes-AA and BaBaRouter present the same degree of complexity, once the mapped netlist is available.

In this way, the work conducted by this Author in [MOR12b] stands off, by validating the use of a language to describe asynchronous circuits in intrachip network communication infrastructures. Results demonstrate BaBaRouter to be a good candidate module to support the construction of intrachip communication architectures for non-synchronous SoCs. This means that, albeit EDA support for asynchronous systems still lag behind their synchronous counterpart, it is already possible to implement competitive circuits using the former. Finally, this work compares the obtained router with a well known synchronous router, implemented with professional industrial tools. Results show that even though the tools used in the design flow of the proposed asynchronous router are in their early stages of development, they provided a circuit with competitive performance. This does not only points to advantages of asynchronous techniques used in intrachip networks, but also to improvements that could be obtained if these tools were enhanced.

The BaBaRouter was described using the Balsa language. The router can be implemented for different asynchronous templates, due to the fact that a Balsa description abstracts data encoding methods as well as communication protocols. Moreover, as the Hermes router, BaBaRouter is fully parameterizable. In fact, it has exactly the same functionality and overall structure.

As Figure 30 shows, BaBaRouter has four fundamental blocks, the input FIFOs and the input control, which are equivalent to the input port in Hermes, the switch control and the crossbar, present in the Control Logic block of Hermes. The blocks are formed by handshake components and each block is itself a handshake component. Consequently, blocks make use of handshake channels to communicate and synchronize. All channels in Figure 30 abstract existing request and acknowledge control signals. Also, five input and five output ports compose the router, East (0), West (1), North (2), South (3) and Local (4), each with input and output interfaces. As in Hermes, ports 0-3 interconnect NoC routers and port 4 to interconnect IPs to the router.

The input FIFO is a basic structure composed by sequentially connected buffers. As stated before, the router is parameterizable. In this sense, width and depth of the FIFOs are a choice of the designer. Neighbor buffers handshake to each other as data flows through the FIFO. When the FIFO is full, the first buffer will not consume data from the input port, until a new position is made available in the FIFO. The width of router channels is the same as the router flit width. Throughout this work, without loss of generality,the Author assumes the use of 8-bit flits and 8-flit deep FIFOs.

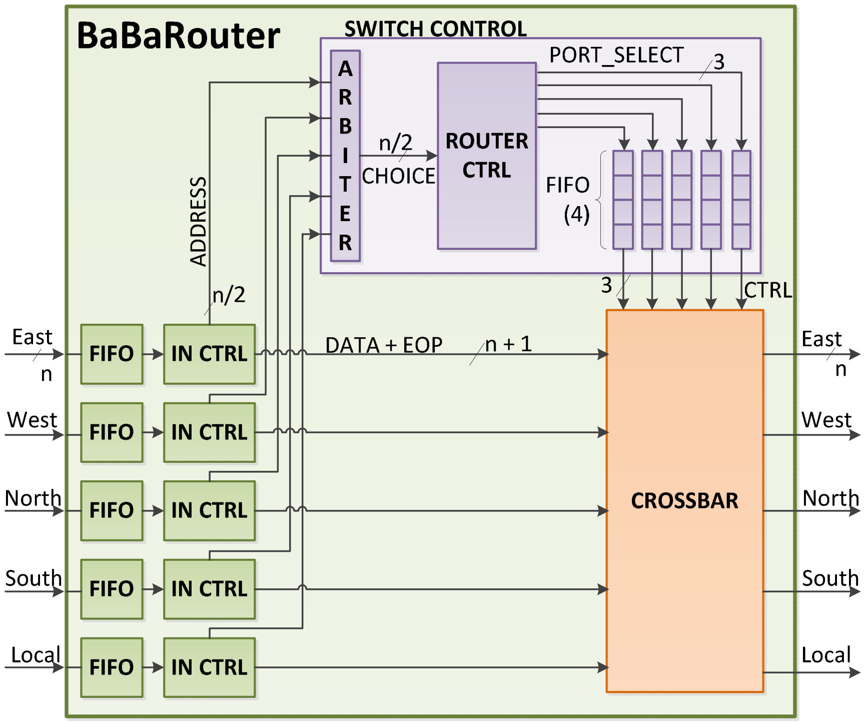


Figure 30 – BaBaRouter implementation block diagram. Communication and synchronization takes place through handshake channels. Control signals are implicit in the drawing. Flit size is denoted by n, which is used to specify the adopted internal channel widths.

The IN CTRL block is responsible of controlling packet information. A packet in BaBaRouter follows the Hermes format: the first flit has the address in its lower half, followed by the payload size, in the second flit, trailed by the payload, all following flits. When the IN CTRL of a given port detects a new communication, it sends the address to the switch control through an (n/2)-bit channel, where n is the flit width. Then, it sends the data, together with an end of package (EOP) signal of value ‘0’, to the crossbar, through an (n+1)-bit channel. The next flit is the payload size. IN CTRL sets an internal register to the value contained in this flit, resets its internal counter and propagates the flit to the crossbar with the ‘0’ EOP value. Next, IN CTRL increments its counter for each new flit received and propagates the flit to the crossbar with EOP=‘0’. When it detects the last flit, by comparing its counter to the internal register that keeps the packet size, it signals EOP=‘1’.

The switch control is responsible for calculating the destination of the packet received in a given input port using the address received from IN CTRL and the internal router address. All input ports share the single switch control to route packets. Therefore, requests from these ports to the switch must be arbitrated. This is required to solve conflicts. For instance, if two ports receive a new package at the same time and the delays of each path to the switch control are equivalent, two requests to use the switch control block arrive at the same time. To choose what packet will be routed first, the switch control arbitrates requests through the ARBITER block. This can be achieved safely at high level using the **`arbitrate`** Balsa construct, which also guarantees freedom of metastability.

Next, the ROUTER CTRL block computes the path that the arbitrated packet must follow. Currently, packets are routed using the XY algorithm. However, other routing algorithms can be easily implemented, due to the high modularity inherent to asynchronous circuits and in particular to Balsa descriptions. The information ROUTER CTRL produces is the destination output port, encoded in 3 bits. The switch control has a 4-position output FIFO with 3-bit slots for each router output port. The generated information decides to what output port the information of what input requested communication is propagated. For instance, consider that a router with address “11” receives a new packet in the East port with destination “11”. The switch control will compute that the packet must follow to the Local output port. Then, it will write in the local output FIFO the value of the East router input (0). In other words, it will inform that the local output port need to be reserved for the East input port. Each new request to the Local output port will be stacked in the output FIFO. The FIFO will be full when all input ports, except the Local port, request the local output port. Note that it is impossible for such FIFOs that any routing request is pending at any time to enter the FIFO.

FIFOs in the outputs are useful to avoid starvation while ensuring fair service to all ports, when multiple input ports request the same output port. A same input port can never request the same output port twice consecutively, if this output is also requested by another input port, because each request fills a position in the FIFO. This is an approach equivalent to the Round Robin arbiter presented in Hermes. The approach however provides fairer arbitration of priorities and costs little hardware. Implementing a pure Round Robin scheduler in an asynchronous circuit would be a much more complex task.

The crossbar binds an output port to an input port. This is done with the information generated by the switch control. When the switch control routes a packet to an output port, it signals to the crossbar through the CTRL channels which input port must be bound to a given output port. The crossbar then binds the ports and propagates the packets received from the IN CTRL until an EOP='1' is received. Then, the port can be bound to a next input port.

Only when the whole packet is transferred, the crossbar finishes the communication with the switch control for a given output port. Thus, a new communication for any of the remaining output ports can take place at any time. BaBaRouter can have data flowing from different input to different output ports concurrently, where the saturation point is when all input ports request communication to different output ports. In this case, five paths are simultaneously established. Figure 31 shows the crossbar block diagram.

The crossbar consumes a large amount of hardware, due to the fact that to guarantee correct operation, for each input port four channels must exist, one for each possible output port. The choice of what channel to use can be viewed as a DEMUX where the control input derives from information coming from the crossbar CTRL block. This block receives data from the switch control and binds input to output ports by generating control signals to the DEMUXes in each input. Channels of a same possible output port are merged into the actual output port. This was the best approach that the authors could find for a Balsa description of a crossbar, to guarantee concurrency of communication for different input/output port pairs.

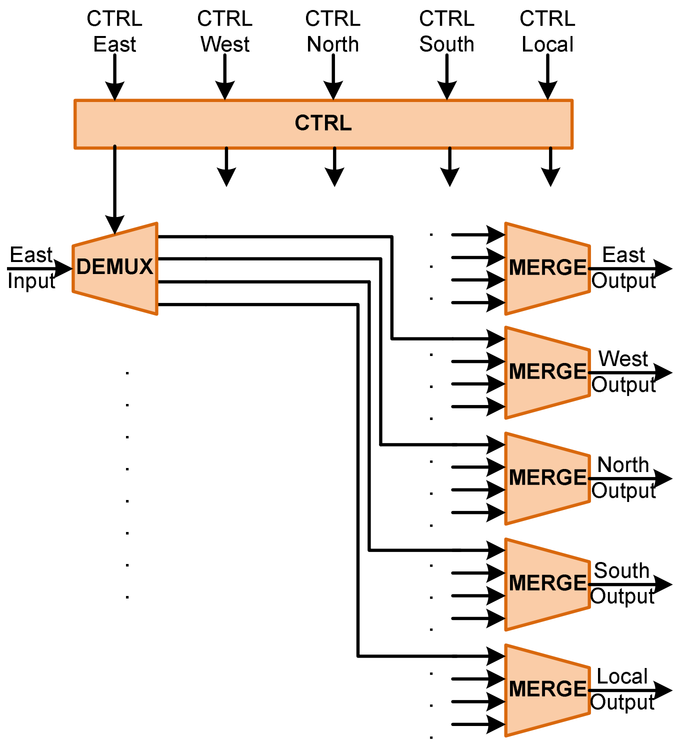


Figure 31 – BaBaRouter Crossbar block diagram.

As Figure 32 shows, when the first flit of a packet is received at an input of the router, it is propagated through the input FIFO and reaches the IN CTRL, which feeds the switch control and the crossbar. The switch control decides the path that the packet must follow and associates an output port to the input port. This can be done concurrently for all output ports, as the switch control generates routing information.

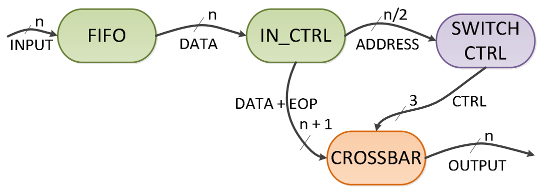


Figure 32 – BaBaRouter dataflow for first flits.

As Figure 33 shows, when the following flits are received at the input port, they follow straight to the output port until the last flit is received. This is due to the fact that the CTRL channel is locked until the IN CTRL block signals EOP='1'. When the crossbar detects this situation, it sends the last flit to the output and clears the CTRL channel. From this point on, a new transmission to that output port can take place.

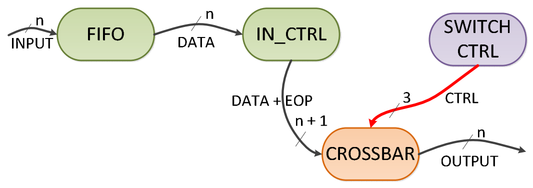


Figure 33 – BaBaRouter dataflow for second and following flits.

As explained before, BaBaRouter was described in Balsa and implemented in the STMicroelectronics 65nm CMOS technology. The Hermes Router was also implemented in the same technology, to compare the obtained circuits. The adopted design flow for BaBaRouter appears in Figure 34. From a Balsa description, the design is compiled into handshake components through the Balsa compiler (Balsa-c), generating a Breeze netlist. This netlist is composed by handshake components only and can be mapped to a specific technology. As mentioned before, this work employs the ASCEnD standard-cell library and the basic standard-cell library of the 65nm technology. Both were made fully compatible with the Balsa framework for synthesizing the QDI four-phase dual-rail router, through the use of in-house tools. This synthesis generates a gate level netlist, which is described in Verilog and is fully compatible with commercial back-end tools.

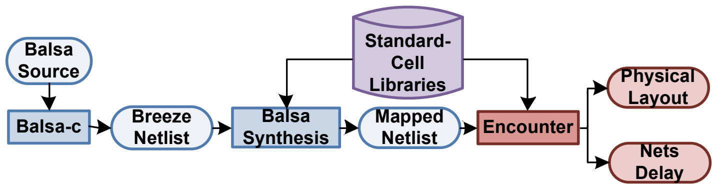


Figure 34 – BaBaRouter design flow.

The generated mapped netlist is imported into the Cadence Backend Framework (Encounter) to generate the physical layout. After place and route, the circuit is extracted and the delay of internal nets is annotated and exported to a standard delay format (SDF) file. This file is the source to conduct timing simulation, for validating the correct behavior of the physical BaBaRouter implementation.

For the implementation of the Hermes Router, the typical synchronous design flow using commercial tools from Cadence was employed, as Figure 35 shows. Router description in VHDL was automatically obtained through Atlas [MOR04]. Using RTL Compiler, the design was elaborated and mapped to the basic standard-cell library of the chosen 65nm technology.

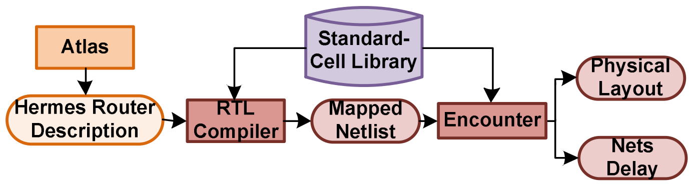


Figure 35 – Hermes Router design flow.

The mapped circuit was used in Encounter to generate the physical layout of the circuit. Similarly to the design flow adopted for BaBaRouter, after place and route, the circuit was extracted and the delay of its internal nets annotated. The correct operation of the circuit was validated through timing simulation.

The maximum operating frequency achieved after synthesizing Hermes was of 1.25GHz. This is due to the fact that its project is highly constrained. The internal logic of the router is sensitive to rising and falling clock edges and the generated critical path comprises registers sensitive to alternate clock edges. In this way each phase of the clock needs to be long enough to respect constraints. This resulted in a rather long clock period, 0.8ns.

After the validation of the physical design of Hermes and BaBaRouter, the obtained results were compared. A simulation scenario was defined in order to evaluate the circuits and conduct power analysis. All results were obtained for the STMicroelectronics 65nm CMOS technology, for a typical fabrication process and operational conditions of 1V and 25^{o}C.

The routers were physically implemented in the STMicroelectronics 65nm CMOS technology, using standard-cell libraries that employed the same devices (general purpose transistors with typical threshold). Moreover, both designs use 7 metal layers and a core density of 95%. TABLE XVIII shows physical information of the generated layouts.

Clearly, BaBaRouter is much more area consuming than Hermes, roughly 4.4 times. This is expected, due to the fact that EDA tools for synchronous designs are much more developed than the ones for asynchronous projects. Moreover, the standard-cell library employed in the design of Hermes is a complete set of typical cells, with a large variety of complex logic standard cells. Thus, the result is an optimized netlist. In contrast, Balsa still lacks some functionality in its synthesis approach. For example, the environment is not able to automatically choose adequate cell driving strength parameters, which potentially leads to more optimized designs. In other words, all cells of a same functionality have the same driving strength (capability of charging or discharging an output load). In this way, some cells can be underused. In some places of the design weaker cells could be employed. Furthermore, the library employed for the synthesis consists of C-Elements and basic gates only. Thus, generating complex asynchronous components requires multiple cells. For example, since the circuit employs dual-rail codification, dual-rail ORs and ANDs must be constructed using C-elements, OR and AND gates. A richer standard-cell library could certainly generate smaller circuits. This is left as future work. As for the I/O pins count, BaBaRouter employs dual-rail codification, therefore each data bit is represented by two wires, doubling the number of data pins used by Hermes. However, some control signals required by Hermes are not required by BaBaRouter, such as the request signal (for handshaking communication), which is encoded in its dual-rail data signals. In this way, the total number of I/O pins is not exactly the double.

TABLE XVIII – Comparison of BaBaRouter and Hermes Router physical implementations.

	BaBaRouter	**Hermes Router**
Standard Cells	10,007	1,822
Standard Cells Area	42,797 µm²	9,625.2 µm²
Standard Cells – Physical Cells Area	40,506.9 µm²	9,174.88 µm²
IO Pins	173	102
Total Wire Length	214.960 mm	46.196 mm

From the standard cell area of BaBaRouter design, subtracting physical cells (fillers and tap cells), 22,376.56µm² are required by C-elements. This means that over 55% of the required circuit area is designated for such cells, testifying the importance of the C-element in asynchronous designs. There are different ways to implement the functionality of a C-element in a standard cell, as [MOR12a] shows, each with its advantages and drawbacks. Therefore, by choosing different C-element implementations, area/power/speed tradeoffs could be obtained. Once more, if Balsa could select cells considering electrical behavior and area consumption for a same set of functionalities, a more optimized circuit could be obtained.

Albeit BaBaRouter consumes more area than Hermes, when compared to other QDI asynchronous routers it is much less area consuming. In fact, as far as the Author could verify, from the dual rail QDI routers implemented in 65nm available in literature, BaBaRouter is the least area consuming.

After physical synthesis, the circuit was extracted from the generated layout and its internal nets delays were annotated. These were the source to conduct timing simulations of the circuits. To do so, a scenario where the routers are operating at the highest possible throughput was described. The scenario, shown in Figure 36, consists in the simulation of a central router of a 3x3 network, with packets flowing in all its input and output ports. The targets specified for each input bind each input to a distinct output. As explained before, the maximum operational frequency for Hermes was of 1.25GHz. However, due to the handshake protocol implemented in the router, each flit, in the best case, takes 2 clock cycles to be transmitted. Therefore, each port may have a maximum throughput of 625Mflits/s. Since the router may have up to five concurrent communications, the maximum throughput of the whole router, in saturation, is 3.125Gflits/s.

For BaBaRouter, the maximum throughput cannot be measured by a clock frequency, because the router is asynchronous. It is measured as the average delay to transmit a flit. The conducted timing simulations showed that the router is capable of transmitting, in average, one flit each 3.164ns. This means that the throughput for the average delay of the router is 316Mflit/s per port or, in saturation, 1.58Gflits for the whole router. Results show that, in best cases, fastest paths, the router may transmit, in saturation, a maximum of 1.750Gflits/s.

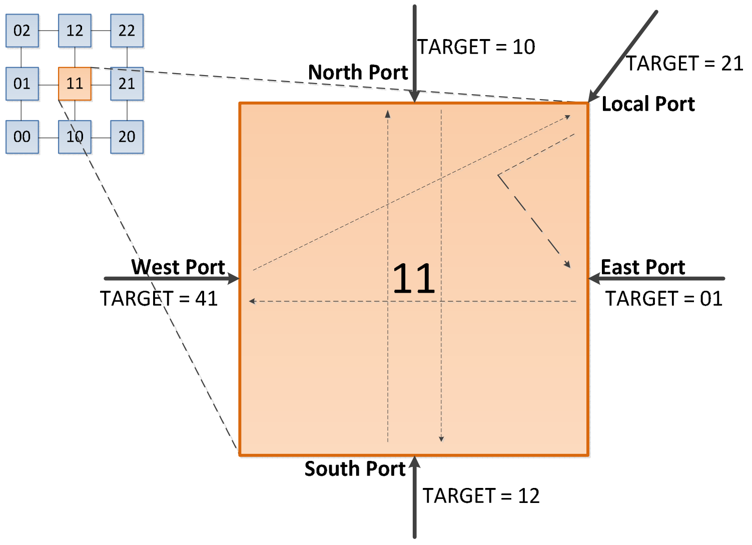


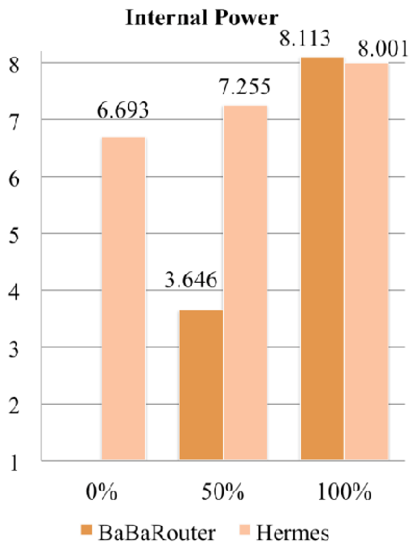
Figure 36 – Scenario to validate functionality of the routers.

Considering the average delay of BaBaRouter, it transmits roughly 50% the data that Hermes is capable of transmitting in a same period of time. Albeit this was not desired, it was expected. This is due to the fact that, as explained before, Balsa does not choose for different driving strengths cells for a same functionality. In this way, cells may end up over used. This means that, in order for the circuit to operate faster, faster cells should be employed in these cases. Still, as far as the Author could verify, BaBaRouter is the fastest asynchronous NoC router in absolute terms.

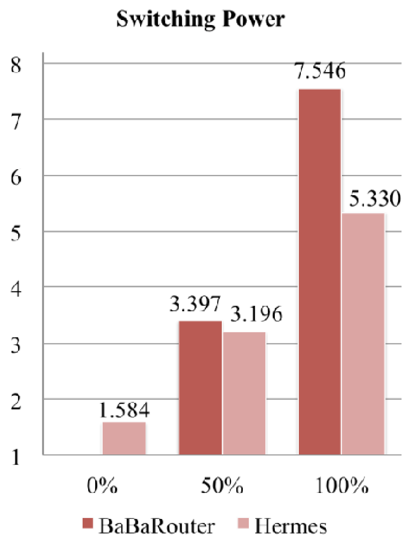
For the power analysis, three simulation scenarios were generated. In the first, routers were idle full time (0%). In the second, routers had data flowing as showed in Figure 37, 50% of the simulation time (50%). For the remaining time, routers were idle. For the third scenario, routers were operating in saturation for the whole simulation time (100%). For BaBaRouter, data was asynchronously inserted in the inputs at its maximum rate. For Hermes, a 732MHz clock was employed, to have a fair power consumption evaluation, since this is equivalent to the saturation zone operation of BaBaRouter. Each scenario was simulated for 100µs and the switching activity of the routers' nets was annotated and exported to a toggle count format (TCF) file. This file was the source to conduct power analysis. Additionally, all results displayed here were obtained for a typical process with typical operational conditions (25^{o}C and 1V), where PMOS and NMOS devices have typical speed.

Figure 37 shows the obtained power results. The charts demonstrate the internal power (a), switching power (b), leakage power (c) and total power (d) consumption for the three scenarios, 0%, 50% and 100%. As Figure 37(d) shows, when idle, BaBaRouter consumes less than 5% of the total power of Hermes. This is a characteristic of asynchronous circuits. Inactive parts of the circuit consume only static power due to leakage currents. However, this power consumption is very small. In contrast, in a synchronous circuit, even when idle, each clock cycle activates all registers in the circuit, causing excessive power consumption.

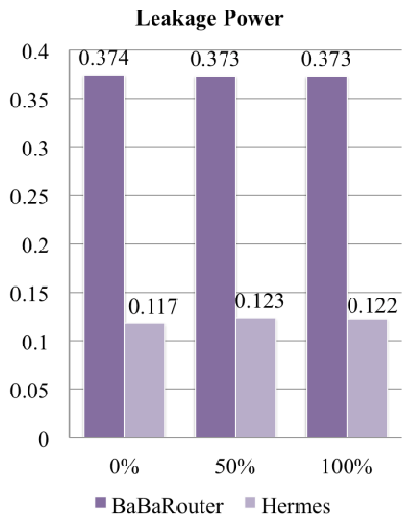
When routers are communicating 50% of the time, BaBaRouter still consumes roughly 30% less power than Hermes. This is due to the fact that it presents lower internal power consumption. As for the switching power, the consumption is slightly higher. This is due to the fact that the circuit has a bigger path from an input to the output to which it is bound, requiring more capacitances to be charged or discharged.



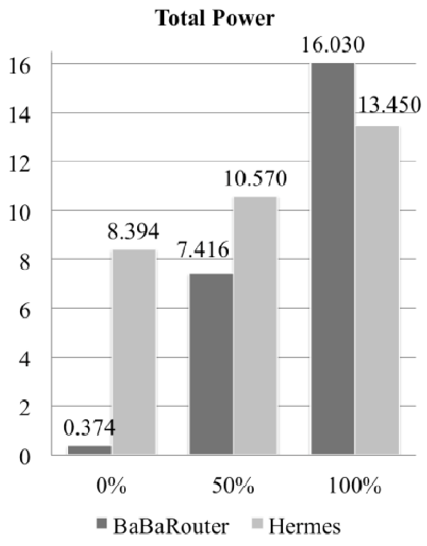
(a)



(b)



(c)



(d)

Figure 37 –Power results (in mW) comparison between BaBaRouter and Hermes: (a) internal power , (b) switching power, (c) leakage power and (d) total power. Three scenarios were evaluated, routers always idle (0%), routers transmitting data 50% of the time (50%) and routers always transmitting data (100%).

For the simulation where routers operate in saturation, Hermes consumes roughly 26% less power than BaBaRouter. This is due to the bigger area required to implement the last. For this scenario where the router is always operating, each part of the circuit remains quiescent for shorter periods of time, increasing dynamic power consumption significantly.

As for the leakage power consumption, which can be constraining in current technologies [EKE10], BaBaRouter consumes roughly 3.2 times more than Hermes. However, asynchronous components, C-elements, are responsible for only 22% of this consumption. The remaining is consumed by typical components. Moreover, if the area of the circuit was reduced, it could present much lower leakage power consumption.

The power consumption per module of BaBaRouter is presented in Figure 38. In the chart, inputs are assumed as the combination of the input FIFO and the IN CTRL. The block that consumes more power is the crossbar (40%), while each port consumes around 12% and the switch control less than 0.5%. Balsa synthesis still requires some improvements that could help reduce crossbar consumption, such as the availability of arbiters with more than two channels. Today, these are constructed through an associative approach of arbiter trees, which generates excessive hardware.

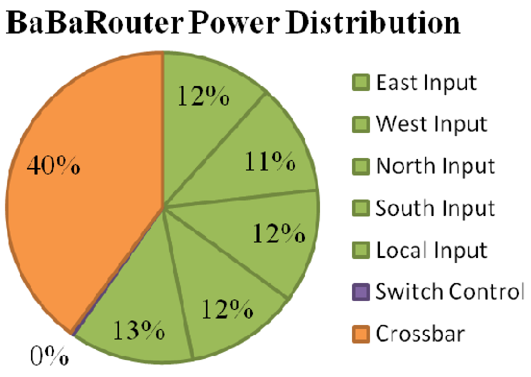


Figure 38 – Power distribution in BaBaRouter.

Figure 39 shows the power distribution in Hermes for the three evaluated scenarios. As the charts show, the clock tree, required by synchronous approaches, consumes up to 50% of the total power of the circuit. Moreover, sequential cells are also responsible for a major part of the power consumption. This is due to the fact that registers are always being activated, even when and where they are not required. In a fully asynchronous circuit, such as BaBaRouter, registers are activated only when and where required. This is the reason why BaBaRouter consumes less power for low and typical traffic scenarios.

Due to their nature, asynchronous circuits tolerate a wider variation in power supply voltage [CRI10]. Lower voltages reduce the operating speed and, consequently, power consumption. Therefore, for applications that require low throughput, lower voltages could be applied to a BaBaRouter in a very straightforward manner, resulting in lower power consumption. For synchronous routers, this technique is not easily adopted, due to timing constraints imposed by the use of a clock signal to control the circuit.

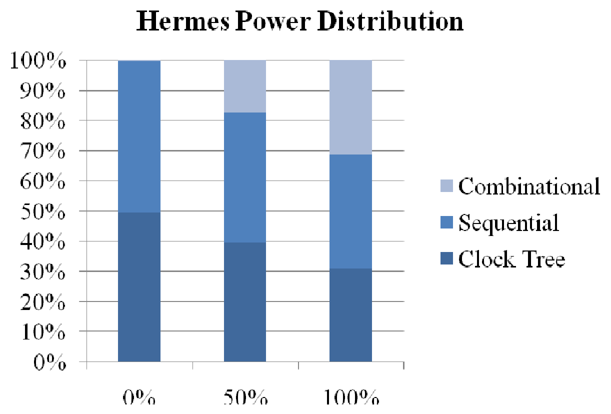


Figure 39 – Power distribution in the Hermes router.

Finally, BaBaRouter supports GALS or fully asynchronous SoCs, while Hermes supports only fully synchronous systems. If synchronizer interfaces were added to Hermes to support non-synchronous SoCs, power, operating frequency and area figures can be compromised. Moreover, albeit BaBaRouter operates at a slower rate than Hermes and consumes more power in some cases, previous results show that it is faster than Hermes-AA, less power consuming and, more importantly, its design complexity is extremely reduced.

As far as the Author could verify, BaBaRouter is the first asynchronous NoC router to be implemented using this approach. Results show that by using Balsa to implement asynchronous intrachip network interfaces, the design effort is highly reduced and competitive performance figures can be obtained, even when compared to a synchronous equivalent circuit. In fact, the implemented router proved to be faster than a previous, in-house designed asynchronous NoC router, which was designed through a full custom approach. In this way, not only better performance figures were obtained, but a lower complexity method to implement asynchronous NoC routers was validated.

When compared to a synchronous NoC router, BaBaRouter requires less power for scenarios where it works at full throughput at least 50% of the time, or less. In fact, these scenarios are close to reality. Scenarios where the router operates 100% of the time at full throughput are very unlikely in real applications. In other words, the designed router may present a solution for low power requirements in NoC-based embedded systems.

Although it does incur large area overhead, the designed router is naturally tolerant to process and operational variations when compared to a synchronous NoC router. This is due to natural characteristics of asynchronous circuits and is advantageous in current technologies, where small fabrication variations may compromise a whole chip. In addition, BaBaRouter presents a more straightforward possibility of implementing techniques to reduce power consumption, e.g. by varying power supply voltage. Further studies are under way to evaluate this and other optimization possibilities.

Finally, if Hermes were to provide all the listed advantages presented by BaBaRouter, area, operational frequency and power figures would be compromised. In this way, results point that advances in EDA tools for supporting the asynchronous paradigm can improve the quality of implementations similar to BaBaRouter, which would be helpful for coping with emerging CMOS technology problems. Future work includes the validation in silicon of BaBaRouter. Moreover, different routing algorithms can easily be added to the router. Also other asynchronous templates can be used to re-implement the router, such as bundled-data and 1-of-4 encoding, combined with 2-phase or 4-phase protocols. In addition, the Author envisage the construction of an automatic generator for BaBaRouter, along with a high level simulation and evaluation environment.

5 CONTRIBUTIONS AND CONCLUSIONS

The work of the Author on asynchronous circuits and GALS system design and automation comprises five types of contributions, which are listed below and related according the diagram of Figure 40:

1. EDA tools for supporting asynchronous design;
2. Design flows for asynchronous circuits and devices and GALS systems;
3. Basic asynchronous devices;
4. FPGA prototyping of asynchronous circuits and GALS systems;
5. Design and validation of ASIC versions of asynchronous circuits and GALS design.

As for the development of EDA tools to support design automation of asynchronous circuits, two tools were developed: the Ring Oscillator Generator (ROGen) and the Cell Specifier (CeS) [MOR11a]. These tools are part of a proposed design flow for implementing asynchronous standard cells. Through this flow, ASCEnD was generated for the STMicroelectronics 65nm CMOS technology and ongoing work comprises porting the library to other technologies, such as IBM 90nm and 180nm. The Author also contributed in the design and prototyping of the Hermes-G and Hermes-GLP in FPGAs. Moreover, through experiments with the Balsa synthesis, low complexity asynchronous circuits – such as asynchronous FIFOs were prototyped in FPGAs as well.

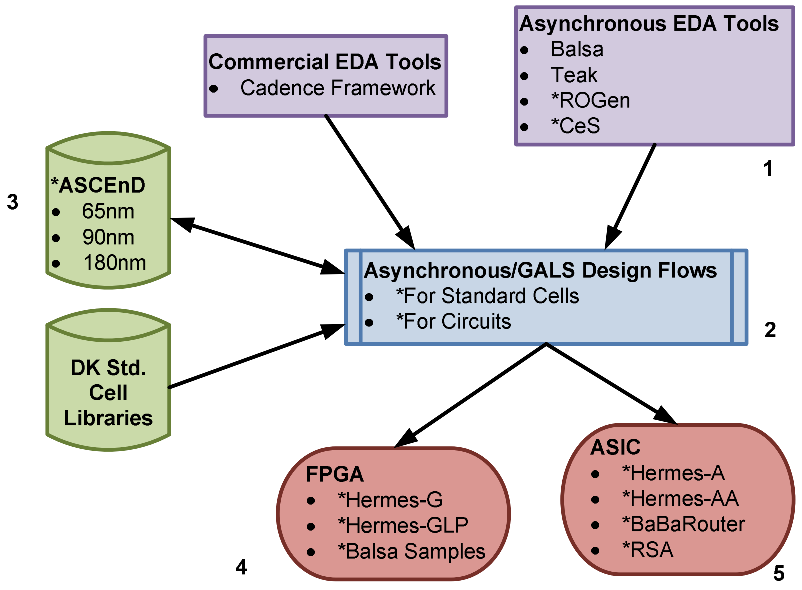


Figure 40 – Diagram expressing the contributions of the Author's work on the design and design automation of asynchronous circuits and GALS systems. A "*" indicates Author's contributions.

As explained before, in order to efficiently experiment with asynchronous circuits, two asynchronous NoCs, Hermes-A and Hermes-AA were designed targeting ASIC implementations. These were mapped to cells in ASCEnD and other standard cell libraries. However, the design required quite laborious manual work. Thus, an automated design flow was proposed, integrating Teak and ASCEnD, for generating asynchronous ASICs. The flow was validated with the implementation of an RSA cryptography core. Teak synthesis proved to be very inefficient.

Therefore, a new design flow was proposed, integrating Balsa synthesis and ASCEnD. Preliminary results indicate that this scheme generates higher quality asynchronous designs than the Teak-based scheme [MOR12b]. Finally, a new asynchronous NoC router was designed through this flow, the BaBaRouter. All ASIC implementations were validated through simulation after physical layout design.

REFERENCES

[ACH12] "Achronix Semiconductor Corporation". Available at http://www.achronix.com.

[AGY10] Agyekum, M.; Nowick, S.: "An error-correcting unordered code and hardware support for robust asynchronous global communication". In: Design, Automation and Test in Europe (DATE'10), 2010, pp. 765-770.

[APT12] "Advanced Processor Technologies Group". Available at http://intranet.cs.man.ac.uk/apt.

[ASC12] "ASCEnD: Asynchronous Standard Cells Enabling n-Designs". Available at https://corfu.pucrs.br/redmine/embedded/asynclib.

[BAI02] Bainbridge, J.; Furber, S. "Chain: a delay-insensitive chip area interconnect". IEEE Micro, 22(5), pp. 16-23, Sep.-Oct., 2002.

[BAR98] Bardsley, A. "Balsa: An Asynchronous Circuit Synthesis System". MSc Dissertation, Faculty of Science & Engineering, University of Manchester, 1998, 162p.

[BAR09] Bardsley, A.; Tarazona, L.; Edwards, D.: "Teak: A Token-Flow Implementation for the Balsa Language". In: Application of Concurrency to System Design (ACSD), 2009, pp. 23-31.

[BEI05] Beigné, E.; Clermidy, F.; Vivet, P.; Clouard, A.; Renaudin, M.: "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.

[BER92] Van Berkel, K.: "Beware the isochronic fork". In: Integration, the VLSI journal, vol.13(2), pp. 103-128, Jun. 1992.

[BER99] van Berkel, C. H.; Josephs, M. B., Nowick, S. M.: "Scanning the Technology: Applications of Asynchronous Circuits". Proceedings of the IEEE, 1999, pp. 223-233.

[BJE06] Bjerregaard, T.; Mahadevan, S. "A Survey of Research and Practices of Network-on-Chip". ACM Computing Surveys, 38(1), pp. 1-51, 2006.

[BJE07] Bjerregaard, T.; Stensgaard, M.B.; Sparso, J.: "A Scalable, Timing-Safe, Network-on-Chip Architecture with an Integrated Clock Distribution Method". In: Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07 , vol., no., pp.1-6, 16-20 April 2007.

[BOU07] Bouesse, F.; Ninon, N.; Sicard, G.; Renaudin, M.; Boyer, A.; Sicard, E. "Asynchronous Logic Vs Synchronous Logic: Concrete Results on Electromagnetic Emissions and Conducted Susceptibility". In: 6th International Workshop on Electromagnetic Compatibility of Integrated Circuits (EMC), 2007, 5p.

[CHA84] Chapiro, D. "Globally Asynchronous Locally Synchronous Systems". PhD Thesis, Stanford University, October 1984, 134p.

[CHO07] Chong, K.; Gwee, B.; Chang, J.S.: "A Simple Methodology of Designing Asynchronous Circuits Using Commercial IC Design Tools and Standard Library Cells". In: International Symposium on Integrated Circuits, 2007, pp.176-179.

[COR06] Cortadella, J.; Kondratyev, A.; Lavagno, L.; Sotiriou, C.P.: "Desynchronization:

Synthesis of Asynchronous Circuits From Synchronous Specifications". In Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions, vol.25(10), pp.1904-1921, Oct. 2006.

[CRI10] Cristófoli, L. F.; Henglez, A.; Benfica, J.; Bolzani, L.; Vargas, F.; Atienza, A.; Silva, F. "On the Comparison of Synchronous Versus Asynchronous Circuits Under the Scope of Conducted Power-Supply Noise". In: Asia Pacific Symposium on Electromagnetic Compatibility (APEMC), 2010, pp. 1047-1050.

[CUM02] Cummings, C. E.: "Simulation and Synthesis Techniques for Asynchronous FIFO Design". In: SNUG, 2002.

[DOB09] Dobkin, R.; Ginosar, R.; Kolodny, A.: "QNoC asynchronous router". In: Integration the VLSI Journal, 42(2), pp. 103-115, Feb 2009.

[EDW06] Edwards, D.; Bardsley, A; Janin, L.; Plana, L.; Toms, W. "Balsa: A Tutorial Guide". 2006, 157p.

[EKE10] Ekekwe, N. "Power dissipation and interconnect noise challenges in nanometer CMOS technologies". IEEE Potentials, 29(3), pp. 26-31, May 2010.

[ERI03] Eriksson, H.; Larsson-Edefors, P.; Henriksson, T.; Svensson, C.: "Full-custom vs. standard-cell design flow – an adder case study". In: Asia and South Pacific Design Automation Conference, 2003, pp. 507-510.

[FEL04] Felicijan, T. and Furber, S.: "An Asynchronous On-Chip Router with Quality-of-Service (QoS) Support". In: 17th IEEE Int. SoC Conf. (SOCC'04), pp. 274-277, 2004.

[FER06] Ferretti, M.; Beerel, P. A.: "High performance asynchronous design using single-track full-buffer standard cells" In: Solid-State Circuits, IEEE Journal of , vol.41(6), pp. 1444-1454, June 2006.

[GUL05] Gulati, G.; Brunvand, E.: "Design of a Cell Library for Asynchronous Microengines". In: GLVLSI, 2005, pp. 385-389.

[HEC12] Heck, G.; Guazzelli, R.; Soares, Rafael Iankowski; Moraes, F. G.; Calazans, N. "HardNoC: A Platform to Validate Networks on Chip through FPGA Prototyping". In: VIII Southern Programmable Logic Conference (SPL'12), 2012, pp. 36-41.

[HEN03] Henkel, J. "Closing the SoC Design Gap". IEEE Computer, 36(9), pp. 119-121, Sep. 2003.

[HOL06] Hollis, S.; Moore, S.: "RasP: An Area-efficient, On-chip Network," In: Int. Conf. on Computer Design (ICCD'06), pp. 63-69, 2006.

[HSU05] Hsu, C. L.; Wang, W. T.; Hong, Y. F.: [1]"Frequency-Scaling Approach for Managing Power Consumption in NoCs". IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences, 88(12), pp. 3580-3583, Dec. 2005.

[ITR08] International Technology Roadmap for Semiconductors. "ITRS 2008 Edition" http://www.itrs.net/, 2008.

[ITR09] International Technology Roadmap for Semiconductors. "ITRS 2009 Edition" http://www.itrs.net/, 2009.

[KIM05] Kim, D.; Kim, M.; Sobelman, G. E.: "Asynchronous FIFO Interfaces for GALS On-Chip Swit-ched Networks". In: SoC´05, pp. 186-189, 2005.

[LIN04] Lines, A.: "Asynchronous Interconnect for Synchronous SoC Design". In: IEEE Micro, 24(1), pp. 32-41, Jan-Feb 2004.

[MAR89] Martin, A. J.: "Formal program transformations for VLSI circuit synthesis". In: Formal Development of Programs and Proofs, E. W. Dijkstra, ed., Addison-Wesley, pp.59-80, 1989.

[MAU03] Maurine, P.; Rigaud, J. B.; Bouesse, F.; Sicard, G.; Renaudin, M.: "Static Implementation of QDI Asynchronous Primitives". In: PATMOS 2003, pp. 181-191.

[MOR04] Moraes, F. G.; Calazans, N. L. V.; Mello, A.; Muller, L.; Ost, L. "HERMES: an Infrastructure for Low Area Overhead Packet-switching Networks on Chip". Integration the VLSI Journal, 38(1), Oct. 2004, pp. 69-93.

[MOR10] Moreira, M. T. "Design and Implementation of a Standard Cell Library for Building Asynchronous ASICs". End of Term Work. Computer Engineering, FACIN-PUCRS, Jan. 2010, 139 p.

[MOR11a] Moreira, M. T.; Oliveira, B. S., Pontes, J. J. H., Calazans, N. L. V. "A 65nm Standard Cell Set and Flow Dedicated to Automated Asynchronous Circuits Design". In: IEEE International SoC Conference (SOCC'11). Taipei, 2011, pp. 99-104.

[MOR11b] Moreira, M. T.; Oliveira, B. S., Pontes, J. J. H., Moraes, F. G.; Calazans, N. L. V. "Adapting a C-Element Design Flow for Low Power" In: IEEE International Conference on Electronics, Circuits and Systems (ICECS'11). Beirut, 2011, pp. 45-48.

[MOR12a] Moreira, M. T.; Oliveira, B. S., Moraes, F. G., Calazans, N. L. V. "Impact of C-Elements in Asynchronous Circuits" In: International Symposium on Quality Electronic Design (ISQED'12), accepted for publication.

[MOR12b] Moreira, M. T.; Magalhães, F. G., Gibiluka, M.; Hessel, F.; Calazans, N. L. V. "Lowering the Design Effort for Efficient Asynchronous Intrachip Communication using the Balsa Framework". Submitted to Integration, the VLSI Journal.

[MOR12c] Moreira, M.; Prates, W.; Calazans, N.: "ASCEnD: A Standard Cell Library for Semi-Custom Asynchronous Design" In: VII Workshop on Semiconductors and Micro & Nano Technology. Accepted for publication.

[MYE01] Myers, C. J. "Asynchronous Circuit Design". New York: John Wiley & Sons, Inc. 2001, 422p.

[OGR07] Ogras, U.Y.; Marculescu, R.; Choudhary, P.; Marculescu, D.: "Voltage-Frequency Island Partitioning for GALS-based Networks-on-Chip". In: Design Automation Conference, 2007. DAC '07. 44th ACM/IEEE , vol., no., pp.110-115, 4-8 June 2007.

[OZD06] Ozdag, R. O.; Beerel, P. A.: "An Asynchronous Low-Power High-Performance Sequential Decoder Implemented With QDI Templates". Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , vol(9), pp.975-985, Sept. 2006

[PAN06] Panades, I.; Greiner, A.; Sheibanyrad, A.: "A Low Cost Network-on-Chip with Guaranteed Service Well Suited to the GALS Approach". In: Nano-Networks and Workshops, 2006. NanoNet '06. 1st International Conference on , vol., no., pp.1-5, Sept. 2006.

[PED05] Pedram, M. and Abdollahi, A. "Low power RT-level synthesis techniques: a tutorial". IEE Proceedings on Computers and Digital Techniques, 152(3), pp. 333-343, May 2005.

[PON07a] Pontes, J.; Soares, R.; Carvalho, E.; Moraes, F.; Calazans, N.: "SCAFFI: An intrachip FPGA asynchronous interface based on hard macros". In: Computer Design, 2007. ICCD 2007. 25th International Conference on , vol., no., pp.541-546, 7-10 Oct. 2007.

[PON07b] Pontikakis, B.; Hung Tien Bui; Boyer, F.-R.; Savaria, Y.: "A Low-Complexity High-Speed Clock Generator for Dynamic Frequency Scaling of FPGA and Standard-Cell Based Designs". In: Circuits and Systems, 2007. ISCAS 2007. IEEE International Symposium on , vol., no., pp.633-636, 27-30 May 2007.

[PON08] Pontes, J. J. H.; Moreira, M. T.; Soares, R. I.; Calazans, N. L. V. "Hermes-GLP: A GALS Network on Chip Router with Power Control Techniques". In: IEEE Computer Society Annual Symposium on VLSI Design (ISVLSI'08), Apr. 2008, pp. 347-352.

[PON10a] Pontes, J. J. H.; Moreira, M. T.; Moraes, F. G.; Calazans, N. L. V. "Hermes-A – An Asynchronous NoC Router with Distributed Routing". In: International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS'10), 2010, pp. 150-159.

[PON10b] Pontes, J. J. H.; Moreira, M. T.; Moraes, F. G.; Calazans, N. L. V. "Hermes-AA – A 65nm Asynchronous Router with Adaptive Routing". In: IEEE International SoC Conference (SOCC'10), 2010, pp. 493-498.

[QUA05] Quartana, J.; Renane, S.; Baixas, A.; Fesquet, L.; Renaudin, M.: "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.

[RAB03] Rabaey, J. M.; Chandrakasan A.; Nikolic, B. "Digital Integrated Circuits a Design Perspective". Upper Saddle River: Pearson Education, 2003, 761p.

[ROS05] Rostislav, D.; Vishnyakov, V.; Friedman, E.; Ginosar, R.: "An asynchronous router for multiple service levels networks on chip". In: Asynchronous Circuits and Systems, 2005. ASYNC 2005. Proceedings. 11th IEEE International Symposium on , vol., no., pp. 44- 53, 14-16 March 2005.

[SAL06] Saleh, R.; Wilton, S.; Mirabbasi, S.; Hu, A.; Greenstreet, M.; Lemieux, G.; Pande, P.P.; Grecu, C.; Ivanov, A.: "System-on-Chip: Reuse and Integration". In: Proceedings of the IEEE , vol.94, no.6, pp.1050-1069, June 2006.

[SHA98] Shams, M.; Ebergen, J. C.; Elmasry, M. I.: "Modeling and comparing CMOS implementations of the C-element" In: Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , vol.6(4), pp.563-567, Dec. 1998.

[SHE08] Sheibanyrad, A.; Greiner, A.; Miro-Panades, I.: "Multisynchronous and Fully Asynchronous NoCs for GALS Architectures," IEEE Design and Test of Computers, 25(6), pp. 572-580, Nov-Dec 2008.

[SIM04] Simunic, T.; Boyd, S.P.; Glynn, P.: "Managing power consumption in networks on chips". In: Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , vol.12, no.1, pp.96-107, Jan. 2004.

[SPA01] Sparso, J.; Furber, S. "Principles of Asynchronous Circuit Design – A Systems Perspective". London: Kluwer Academic Publishers, 2001, 337p.

[SUT89] Sutherland, I. E.: "Micropipelines". Communications of the ACM, vol.32, Jun. 1992, pp. 720-738.

[TOM06] Toms, W. B.: “Synthesis of Quasi-Delay-Insensitive Datapath Circuits”. PhD Thesis, University of Manchester, 2006, 237p.

[WAN06] Wang, X.; Ahonen, T.; Nurmi, J.: “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.

[WOR05] Worm, F.; Ienne, P.; Thiran, P.; De Micheli, G.: “A robust self-calibrating transmission scheme for on-chip networks”. In: Very Large Scale Integration (VLSI) Systems, IEEE Transactions on , vol.13, no.1, pp.126-139, Jan. 2005.

[ZHA07] Zhao, Chong. “Analysis and Design of Reliable Nanometer Circuits”. PhD Thesis, Electrical Engineering Dept. University of California, San Diego, 2007, 186 p.