Return-to-One Protocol for Reducing Static Power in C-Elements of QDI Circuits Employing m-of-n Codes

Matheus T. Moreira, Ricardo A. Guazzelli, Ney L. V. Calazans
Faculty of Computer Science - Pontifical Catholic University of Rio Grande do Sul – PUCRS - Porto Alegre, Brazil
{matheus.moreira, ricardo.guazzelli}@acad.pucrs.br, ney.calazans@pucrs.br

***Abstract*—The scaling of microelectronic technologies brings new challenges to the design of complex SoCs. For example, fully synchronous SoCs may soon become unfeasible to build. Asynchronous design techniques increasingly mingle within SoC design procedures to achieve functional and efficient systems, where synchronous modules are independently designed and verified. This is followed by module integration by means of asynchronous interfaces and communication architectures, forming a globally asynchronous, locally synchronous (GALS) system. Among multiple asynchronous design styles, the quasi delay insensitive (QDI) stands out for its robustness to delay variations. When coupled to delay insensitive (DI) codes like *m-of-n* and to four-phase handshake protocols, the QDI style produces the dominant asynchronous template currently in use. This work presents a technique to reduce the static power consumption of asynchronous QDI circuits using any *m-of-n* code and a four-phase handshake protocol, by proposing the utilization of a non-classical spacer encoding, namely all-1s. The article shows that the use of the traditional all-0s spacers may lead to static power consumption figures that are in some cases more than twice larger than the static power consumed by all-1s spacers in C-elements, the most common device used in asynchronous templates. Experiments demonstrate the new spacer reduces static power consumption without increase in complexity.**

Keywords-component; Asynchronous circuits, QDI, m-of-n codes, delay-insensitive codes, four-phase protocols, C-elements

I. INTRODUCTION

Asynchronous design techniques are becoming common in very large scale integration (VLSI) circuits. These help overcoming difficulties emerging in deep submicron (DSM) CMOS technology nodes like 45, 28nm and below that lead to over-constrained synchronous systems. In fact, according to the ITRS in its 2009 edition [1], a key challenge in modern IC design is the distribution of a single clock signal throughout the chip to control the whole circuit. Therefore, a shift on VLSI design paradigm seems inevitable.

Current technologies allow the implementation of systems on a chip (SoCs), comprising a large amount of intellectual property cores (IP cores) interconnected through specialized communication architectures. Each of these IP cores may employ particular standards and/or protocols and present 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. However, to guarantee correct communication between distinct frequency domains, asynchronous interfaces are mandatory. SoCs that employ different frequency domains with asynchronous interfaces to communicate at system level are called globally asynchronous locally synchronous (GALS) [2]. In addition to the use of point-to-point asynchronous interfaces, current literature pledges the use of systematically built SoC communication architectures, giving rise to the well accepted concept of networks on chip or NoCs [3]. Although NoCs can be built as synchronous subsystems, its chip-wide nature and time locality (every part of a NoC is typically active for low periods of time) indicate that asynchronous implementations can be advantageous in terms of performance and/or power, as already showed in some recent works, like e.g. in [4].

The QDI design style is attractive for the design of asynchronous circuits for many reasons, but especially for enabling simple timing closure and analysis [5]. Designing with QDI requires the use of delay insensitive (DI) codes [6]. Albeit a wide variety of such codes exist, just a few are practical in CMOS design [5]. Currently, the class of m-of-n codes is used with predominance. Specific codes used very often are the dual-rail (or 1-of-2 for each data bit) code and the 1-of-4 code for each data bit. It is necessary to add a selection of handshake protocol to an asynchronous design style and a code, in order to obtain an asynchronous design template. Such templates may substitute the synchronous design template. The most popular asynchronous handshake protocol is the 4-phase, due to its reduced design complexity and robustness. Given an asynchronous design template as a triple (style, code, handshaking protocol), it is possible to define a set of basic handshake components to design circuits with this template, just as RTL components (registers, multiplexers, demultiplexers, decoders, arithmetic operators, etc.) allow the design of circuits with the synchronous design template. As RTL components are built from logic gates, asynchronous templates also rely on logic gates, which quite frequently need to be enhanced with two other component types: metastability filters, to avoid or to delimit the occurrence of synchronization failures, and C-elements, that help dealing with independent events´ synchronization. While metastability filters appear only in very specific places, C-elements are widespread in QDI circuits, typically occupying more than 60% of the standard cell area of modules [7]. A successful example of asynchronous design is in networks-on-chip (NoC) based on asynchronous routers. These were recently proposed to support the design of GALS SoCs, as described in [8], [9],

978-1-4673-2608-7/12/$31.00 ©2012 IEEE

[10] and [11]. Most of these employ an asynchronous template based on a QDI style and some m-of-n code. As C-elements are a big portion of the total router silicon area, circuit quality directly relates to C-elements quality.

In this era of mobile, battery-based products, power consumption is a major figure. In the past, ignoring static power, due to its relative insignificance in face of the magnitude of dynamic power consumption, was sound. However, DSM nodes testify an exponential growth of static power consumption [12] and as well as a change in the static-dynamic power balance.

This work presents the results of a study that managed to reduce static power consumption of C-elements by more than 50% in asynchronous circuits employing an asynchronous template based on QDI m-of-n codes and 4-phase handshake protocol. Obtaining this reduction depends on a slight change in the handshake protocol only. Moreover, the change adds no design complexity and implies no loss of performance.

The rest of this paper is organized in four sections. Section II describes some basic concepts about asynchronous circuits. Section III discusses related work and presents the proposed technique to reduce static power consumption. Section IV describes simulations that validate the technique and discusses results. Finally, Section V presents conclusions and direction for further work.

II. ASYNCHRONOUS CIRCUITS CONCEPTS

A digital circuit is *synchronous* if its structure implies the use of a single clock signal controlling all circuit events. Otherwise it is called *non-synchronous*. As a special case, a digital circuit is *asynchronous* when no clock signal is used to control any sequencing of events. These employ explicit handshaking among their components to synchronize, communicate and operate [13]. Characterizing an asynchronous design requires the choice of: (*i*) a delay model, (*ii*) a code to represent information, (*iii*) a handshake protocol, and (*iv*) a set of basic components. Each of these is explored in the rest of this Section.

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 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 QDI circuit class. Here, signal transitions occur at the same time only at each end point of the mentioned forks, which are called *isochronic forks*. Usually, the set of basic components of an asynchronous design is created to incorporate all isochronic forks needed to guarantee delay insensitivity. Thus, design may ignore delays altogether.

There are different ways to encode information to adequately support delay models that communicate through handshake protocols. 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, timing restrictions between control and data signals need to be guaranteed at every handshake point, making design of large circuits non-scalable. As an alternative, DI encodings [6] are robust to wire delay variations, because request signals are embedded within data signals. An example is the class of *m-of-n* codes [6]. Given n and m, with $m<n$, an *m-of-n* code consists in the set of all n-bit code words with Hamming weight (i.e. the number of bits in 1 in the code word) equal to m. For example, the well-known *one-hot* codes are *1-of-n* codes. The use of *m-of-n* codes coupled to a protocol that establishes how valid codes succeed one another in a data flow allows obtaining communication with absolute insensitivity to delay variations in individual wires.

Handshake protocols can be either *2-phase* or *4-phase* [13] both illustrated in Figure 1. Usually 2-phase protocols operate faster, but require more hardware than 4-phase protocols. The latter requires that after each data transmission wires return to a fixed logic state, the so-called spacer, which is not a valid code word in the used code. While a 4-phase protocol increases the time to propagate values, it reduces control complexity. This is the most commonly used protocol.

One approach to achieve delay insensitivity consists in representing each data bit in a circuit by a pair of wires, in what is called dual-rail (DR) encoding (i.e. each bit is represented by a *1-of-2* code). Let *d.t* (data true or *1*) and *d.f* (data false or *0*) be the names of two wires representing a single data bit. One example of 2-phase handshake using a 1-bit dual-rail encoding appears in Figure 1(a). Here, a transition in wire *d.f* signals a logical 0 value, which is recognized by a transition in the signal acknowledge (*ack*). A transition in *d.t* signals a logical 1 value, which is again acknowledged by a transition in ack. Other 2-phase protocol implementations exist [13]. Note the protocol requires that *d.t* and *d.f* never transition at the same time, and that a subsequent transition in a wire can only occur after a transition in the *ack* signal. Also, in this 2-phase protocol, the data encoding varies in time. Transitions on specific wires represent data, in a clearly glitch-sensitive scheme. This requires careful logic design and because of this may incur in significant hardware overhead.

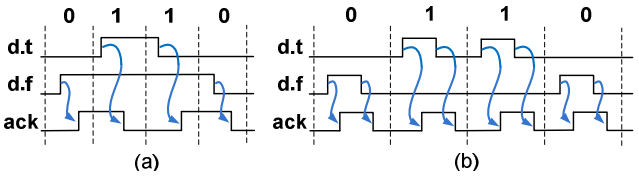


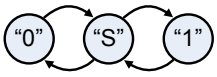
Figure 1. Examples of 2-phase (a) and 4-phase (b) handshake protocols.

Figure 1(b) shows an example 4-phase protocol using dual-rail encoding. Logical levels in wires uniquely identify data bit values. A widely adopted subclass of m-of-n codes is dual-rail (DR). Each dual-rail code is in fact a proper subset of some m-of-2m code. This code uses two wires to represent each bit of an m-bit code word. Again, let *d.t* and *d.f* be the names of two wires representing one bit of some DR code. Valid bit values are always valid code words of the 1-of-2 code ("01" for bit 0 and "10" for 1). However, after a value is acknowledged, all wires must return to a

predefined value, here the all-0s spacer. The spacer itself is an invalid DR code word. This protocol is denominated return to zero or RTZ. In Figure 1(b), the first communicated data value is a logical 0, encoded by *d.t=0* and *d.f=1*. After the value is acknowledged by a low-to-high transition in the *ack* signal, a spacer is issued, in this case *d.t=0* and *d.f=0*. Next, the ack signal switches to 0, signaling reception of the spacer, and a new transmission may occur. Any 4-phase protocol requires spacers when using *m-of-n* codes. Figure 2 shows the RTZ conventions.

Value	*d.t*	*d.f*
Spacer	0	0
Logical 0	0	1
Logical 1	1	0
Invalid	1	1

(a)



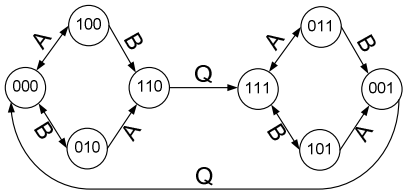
(b)

Figure 2. Example of (a) 4-phase DR encoding and (b) values transition.

Asynchronous design templates can greatly benefit from the availability of basic components other than ordinary logic gates and flip-flops available in current standard cell libraries. These include e.g. special registers, event fork, join and merge devices, as well as metastability filters. Although most of these may be built from logic gates, this is inefficient. A fundamental device that enables to build several such elements more effectively is the C-element. The importance of C-elements is the fact that they may help in the synchronization of independent events. Figure 3(a) depicts the truth table and Figure 3(b) shows a transition diagram for an ordinary 2-input C-element. Its output switches only when all inputs have the same logical value. When inputs A and B are equal, output Q assumes this same value. However, when inputs are different, the output keeps the previous logic value. The asynchronous state transition diagram of Figure 3(b) for the C-element has vertices containing values of inputs and output in the order ABQ_i.

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

(a)



(b)

Figure 3. Simple 2-input C-element specification: (a) truth table, and (b) asynchronous state transition diagram.

III. The Return-to-one Protocol

A. C-elements in QDI asynchronous templates

The quality of circuits built using asynchronous templates depends greatly on the quality of the C-elements provided to implement it. For simplicity sake and without loss of generality with regard to any other *m-of-n* code, this paper assumes the use of DR codes. In QDI asynchronous DR circuits based on 4-phase handshake (QD4), registers are based on C-elements [14] [15], as Figure 4 shows.

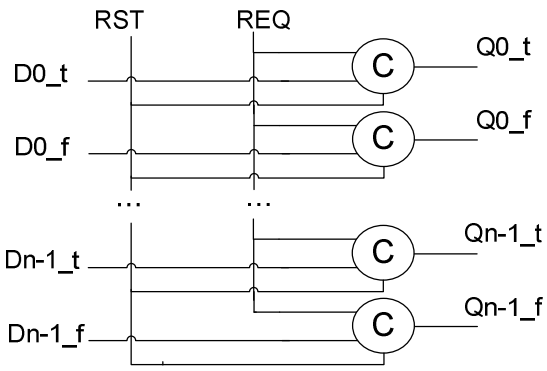


Figure 4. Example of a C-element based DR register.

The RST signal ensures that all outputs are initiated at a well defined logical value, typically the spacer, often represented by the all-0s value. Then, available data, represented in wires *Di_t* and *Di_f*, for *0≤i≤n-1*, is only propagated to the outputs *Qi_t* and *Qi_f* after the REQ signal is issued. For instance, assume that the depicted register has a spacer in its output. According to Figure 2(a), all output signals are at logical 0. Now, imagine that a valid data bit *i* appears at the inputs. If it is a logical 0, the *Di_f* wire is at logical 1, which propagates as soon as the REQ signal is set to logical 1. However if the data bit is a logical 1, *Di_t* is the wire that will switch to logical 1 and its value is propagated once REQ assumes a logical 1 value.

To implement Boolean functions without losing the delay insensitivity property, different component schemes can be employed for asynchronous circuits in general and specifically for QD4 circuits. One of the most used, due to its simplicity, is the delay insensitive minterm synthesis (DIMS) [13]. In this approach, all minterms of the input variables are generated by C-elements and are then combined to perform a given function, similar to two-level logic implementations used e.g. in PLAs. For example, Figure 5 presents a QD4 DIMS half adder schematic.

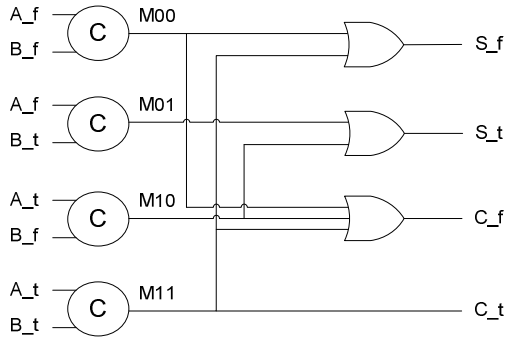


Figure 5. Example of DIMS half adder.

All minterms are generated by the C-elements and the outputs are computed through OR gates. *S_f* must be at logical 1, identifying a logical 0 value in the sum, only when both inputs A and B have the same value, which mean that either M00 or M11 will be at logical 1. When inputs have different values, identifying a logical 1 value in the sum, S_t

will be at logical 1. The carry signal is computed in a similar way. Although DIMS implies a large amount of hardware to compute each minterm, it is widely used. In DIMS, C-elements do play a major role in circuit logic.

B. Problem Statement and Proposed Technique

During the development of ASCEnD [16], a standard cell library for asynchronous circuits in 65nm CMOS technology, the static power consumption of three different implementations of a 2-input C-element was characterized for each combination of input values. The implementations were Martin's, proposed in [14], Sutherland's, proposed in [17] and the van Berkel C-element, proposed in [18]. For all these components, the static power consumption for a logical 0 in the output was at least 70% larger than the one measured when the output was fixed at logical 1. In view of this observation, it is expected that an asynchronous circuit in idle state, i.e. with all C-elements with a spacer in its outputs, may have the static power consumption of all C-elements reduced significantly if the spacer is encoded by all wires at logical 1, instead of all wires at logical 0. The modification does not imply in increased design complexity nor reduces circuit performance[1].

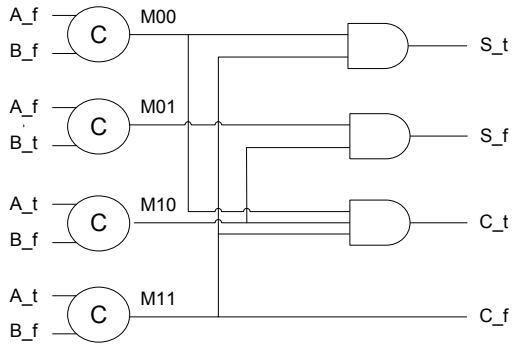


Figure 6. Example of a modified DIMS half adder.

For example, the register showed in Figure 4 can be easily modified to accept the all-1s spacer, if its RST signal sets the output to logical 1 and the REQ signal is active low. Moreover, the DIMS half adder showed in Figure 5 can also be easily modified by swapping the true and false wires of the carry and sum bits and replacing the OR gates by ANDs, as Figure 6 illustrates. A spacer composed by the all-1s encoding represents the proposed Return-to-One (RTO) protocol. Due to the fact that asynchronous circuits are only active when and where required, when the circuit is active, some blocks are computing and some blocks are quiescent. This means that even when the circuit is operating, a portion of its C-elements will have spacers on their outputs. Thus, the use of RTO reduces not only the power consumed by the circuit in idle state, but also while the circuit operates.

C. Related Work

As far as the authors could verify, there is no work in the literature making use of a different value of spacer to lower power consumption of (asynchronous) circuits. However, the authors found three works where the all-1s spacer is used to achieve robust cryptographic hardware.

In [19], the authors propose a dual-spacer protocol, where each data value is between two different spacers: the classic all-0s spacer and an all-1s spacer. According to [19], the single spacer scheme proved to fall short in balancing the switching activity between rails, which leads to a cryptographic core vulnerable to side-channel analyses, like power and electromagnetic attacks. Results showed high robustness to these attacks can be obtained by using the dual-spacer scheme. The drawback is the increased overhead in area and power consumption. In [20], a similar work was conducted. Authors present results measured in a prototype AES cryptographic core designed with standard EDA tools, which employ the same dual-spacer technique. Results show that the technique provides high robustness to attacks and a high cost in circuit area. Similarly, in [21] authors use the all-1s encoding to represent an alarm state to obtain a balanced implementation.

IV. EXPERIMENTS AND DISCUSSION

An analysis of the electrical characterization of three implementations of C-elements conducted during the design of the ASCEnD library revealed that when all inputs are at logical 0, static power dissipation was at least 70% larger than when they were at logical 1. Moreover, in some cases it was 100% as large. Table I shows the static power of C-elements measured by electrical simulation for a 65nm CMOS process in typical fabrication corner (25°C, 1V) for two different scenarios: inputs at logical 0 and at logical 1.

TABLE I. STATIC POWER CONSUMPTION IN C-ELEMENTS WHEN INPUTS ARE AT THE SAME LOGICAL LEVEL.

	Static Power (nW)		Savings
C-element	**Logical 0**	**Logical 1**	
Sutherland	61.703	30.894	50%
Martin	66.858	39.107	42%
van Berkel	57.926	28.425	51%

In this experiment, the three employed C-elements had the same driving strength. As apparent from the results, a spacer signaled by all-0s presents, in the worst case, an overhead of over 100% in static power consumption when compared to a spacer signaled by all-1s.

As explained before, most methods for implementing Boolean functions in QDI asynchronous circuits rely on C-elements. Registers, in turn, need C-elements with a control signal that guarantees that their output will have a fixed and predefined value in the initial state, usually the spacer value. The structure of the evaluated C-elements or any other C-element variation can usually be divided into the blocks (0), (1) and (2), as displayed in Figure 7. Block (0) is the *inverted logic function*, block (1) is the *output inverter* and block (2) is the *state keeper*. Block (1) is responsible for driving (charging/discharging) the output load, while block (0) is responsible for setting the value of the internal node, which feeds the output inverter. Block (2) implements the memory mechanism that holds previous values.

[1] Note that as usual, the ASCEnD library was designed to provide symmetric excursion between 0 to 1 and 1 to 0 transitions in all of its cells, to provide predictable performance figures.

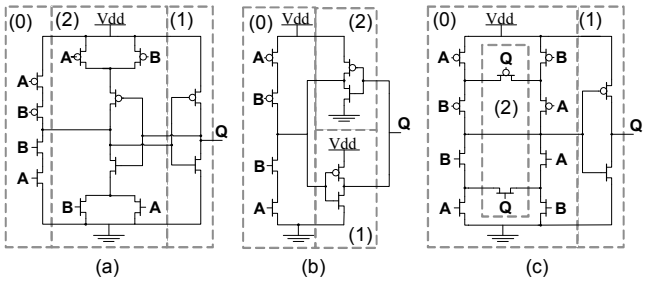


Figure 7. CMOS schematic of the employed C-elements: (a) Sutherland's, (b) Martin's and (c) van Berkel's.

To generate the initial state for an RTZ protocol-based circuit, a resettable C-element is employed. This component can be easily built by adding an NMOS and a PMOS transistor in a typical C-element. Connecting external PMOS and NMOS transistors as Figure 8(a) shows to blocks (0) and (2), a resettable C-element is obtained.

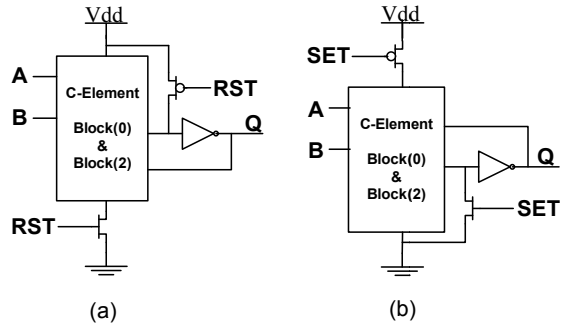


Figure 8. Initialization schemes for C-elements: (a) resettable C-element; (b) settable C-element.

To generate a logical 1 in the C-element output as initial state, for RTO, a settable C-element is required. This can be obtained by connecting PMOS and NMOS transistors with blocks (0) and (2) as Figure 8(b) depicts.

A comparison of the effects of the RTZ and RTO protocols in the power consumption of resettable and settable C-elements was performed, by simulating these components in the same 65nm CMOS technology used in the previous evaluation, for a typical fabrication process at 25°C and 1V. Results appear in Figure 9, where resettable and settable C-elements were simulated for both RTO and RTZ protocols. As the chart shows, in fact, a resettable C-element is in most cases more efficient in terms of static power consumption than a settable C-element for the RTZ protocol. The settable C-element, in turn, consumes less static power than its resettable counterpart for the RTO protocol. Furthermore, a resettable C-element in RTZ protocol consumes at least 50% more static power than a settable C-element in an RTO protocol. These results are summarized in Table II. As Table II also shows, employing settable C-elements for an RTO protocol leads to at least 35% of savings in static power consumption, when compared to a resettable C-element for an RTZ protocol. Moreover, in the best cases savings may reach nearly 50%. Because in *m-of-n* codes-based circuits registers are basically composed by C-elements, and this static power reduction can lead to substantial savings in the register power consumption.

TABLE II. A STATIC POWER CONSUMPTION COMPARISON FOR RESETTABLE (RST) C-ELEMENTS IN RTZ PROTOCOLS AND SETTABLE (SET) C-ELEMENTS IN RTO PROTOCOLS.

	Static Power (nW)		Savings
C-element type	RST / RTZ	SET / RTO	
Sutherland´s	60.101	31.260	48%
Martin´s	71.975	46.637	35%
van Berkel´s	66.603	35.466	47%

A 50% reduction in the static power consumption may represent a large economy in total power consumption, particularly in current technology nodes, where static power can be constraining [21]. For instance, in the case study presented in [7], C-elements consumed over 60% of the total area of a DR asynchronous cryptographic RSA core. Moreover, the static power consumption of the circuit reached 30% of the total power consumption. In this context, reducing the static power consumption would have a substantial impact in the total power consumption of the circuit. Furthermore, given the proportion of the total area required by C-elements, if the proposed RTO protocol was applied, the static power at the core level could be significantly reduced. In this way, total power consumption of the integrated circuit (IC) would also be reduced.

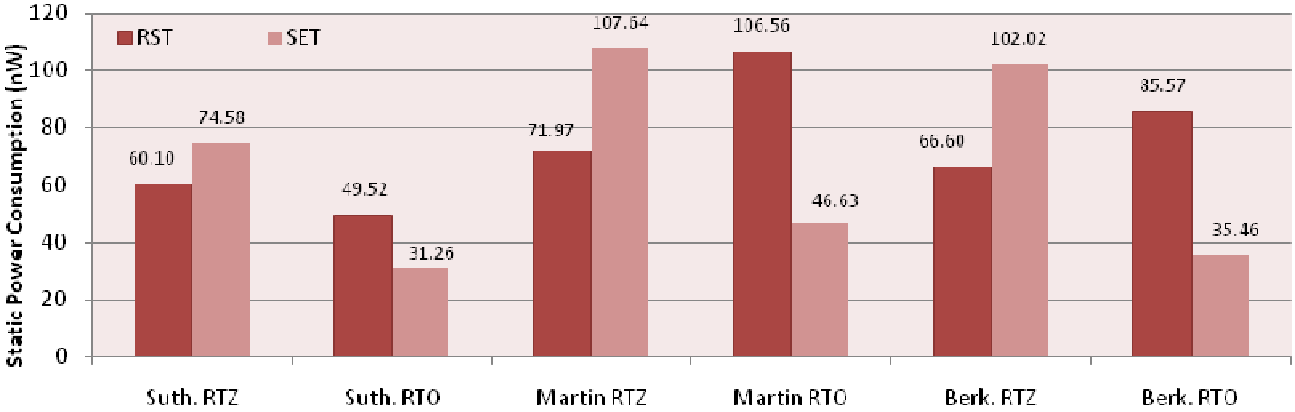


Figure 9. Static power consumption for three implementations of a C-element with set and reset control signals for RTO and RTZ protocols. Suth. stands for Sutherland's and Berk. for van Berkel's. RST stands for resettable and SET for settable C-elements.

Finally, as explained before, in the proposed RTO protocol no complexity is added to circuit design and there are no losses in performance in terms of speed. More importantly, dynamic power consumption is not increased either. This is because there is no change in the transitions for valid signals. In both protocols, RTZ and RTO, valid data is issued by two events: a high-to-low and a low-to-high transition in the output. The order of these events depend on the protocol but do not change the total dynamic power consumed or the total propagation delay of each valid signal communication. For instance, in the RTZ protocol, valid data is issued by a low-to-high transition in the output of e.g. a register. Once it is acknowledged, follows a high-to-low transition in the same output of the register, to indicate the spacer. In the RTO protocol, communication happens the other way. Valid data is issued by a high-to-low transition. Once acknowledged, valid data is followed by a low-to-high transition in the same wire to represent the spacer. Therefore, valid code words in m-of-n codes that employ the RTO protocol are issued with high-to-low transitions.

V. CONCLUSIONS

This work proposed a method able to reduce static power of C-Elements by more than 50%. This represents solid power savings in current technologies, where static power consumption takes an increasing portion of total power consumption. Besides, in asynchronous circuits, where even when the circuit is operating some parts of it are quiescent, static power consumption represents a large portion of the overall power.

All presented results were obtained through electrical simulations of previously designed C-elements in a 65nm CMOS technology validated after physical layout extraction. C-elements are basic blocks in asynchronous circuit design and represent up to 60% of the total area required by standard cells in a complex module. In more advanced nodes like 45nm and 28nm, power savings in asynchronous modules are expected to be larger.

Future work includes detailed analysis of complex asynchronous integrated circuits that employ the proposed technique, in order to measure its efficiency at system level. Moreover, different benchmark circuits that employ these components will be evaluated, to identify where they are more critical and how to achieve greater total power reductions using the RTO protocol. Another future work is associating RTO with other *m-of-n* codes and evaluating the resulting implementations.

ACKNOWLEDGMENTS

This work is partially supported by the CAPES-PROSUP and FAPERGS (under grants 11/0455-5 and 11/1445-0). Ney Calazans acknowledges CNPq support under grant 310864/2011-9. Authors acknowledge support granted to the INCT-SEC (National Institute of Science and Technology – Critical Embedded Systems – Brazil), process no. 573963/2008-8.

REFERENCES

[1] International Technology Roadmap for Semiconductors. “Design Section”, 2009, available at http://www.itrs.net.

[2] D. Chapiro. “Globally Asynchronous Locally Synchronous Systems”. PhD Thesis, Stanford University, 1984, 134p.

[3] A. Agarwal, C. Iskander, and R, Shankar. “Survey of network on chip (NoC) architectures & contributions”. Journal of Engineering, Computing & Architecture, vol. 3(1), 2009, 15p.

[4] I. Miro-Panades, F. Clermidy, P. Vivet, and A. Greiner. “Physical implementation of the DSPIN network-on-chip in the FAUST architecture”. In: NOCS´08, 2008, pp. 139-148.

[5] W. J. Bainbridge, W. B. Toms, D. A. Edwards, and S. B. Furber. “Delay-insensitive, point-to-point interconnect using m-of-n codes”. In: ASYNC´03, 2003, pp. 132- 140.

[6] T. Verhoeff. Delay-insensitive codes- an overview. Distributed Computing, vol. 3(1), 1988, pp. 1-8.

[7] M. T. Moreira, B. S. Oliveira, J. J. H. Pontes, F. G. Moraes, and N. L. V. Calazans. “Impact of C-elements in asynchronous circuits”. In: ISQED´12, 2012, pp. 438-444.

[8] R. Dobkin, R. Ginosar, and A. Kolodny. “QNoC asynchronous router”. Integration the VLSI Journal. Vol. 42(2), 2009, pp. 103-115.

[9] E. Beigné, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin. “An asynchronous NoC architecture providing low latency service and its multi-level design framework”. In: ASYNC´05. 2005, pp. 54-63.

[10] S. Hollis and S. Moore. “RasP: an area-efficient, on-chip network”. In: ICCD´06, 2006, pp. 63-69.

[11] J. J. H. Pontes, M. T. Moreira, F. G. Moraes, and N. L. V. Calazans. “Hermes-AA: a 65nm asynchronous NoC router with adaptive routing”. In: SOCC´10, 2010, pp. 493-498.

[12] B. Deepaksubramanyan and A. Nuñez. “Analysis of subthreshold leakage reduction in CMOS digital circuits”. In: NVSD´07, 2007.

[13] J. Sparsø and S. B. Furber. “Principles of asynchronous circuit design – a systems perspective”. Kluwer Academic Publishers, Boston, 2001, 360 p.

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

[15] E. Yahya and M. Renaudin. “QDI latches characteristics and asynchronous linear-pipeline performance analysis”. TIMA Technical Report TR 06/06-03, 2006, 11 p.

[16] M. T. Moreira, B. S. Oliveira, J. J. H. Pontes, F. G. Moraes, and N. L. V. Calazans. “Adapting a C-element Design Flow for Low Power”. In: ICECS´11, 2011, pp. 45-48.

[17] I. Sutherland. “Micropipelines”. Communications of the ACM. Vol. 32, 1992, pp. 720-738.

[18] K. van Berkel. “Beware the isochronic fork”. Integration, the VLSI Journal, vol. 13(2), 1992, pp. 103-128.

[19] W. Cilio, M. Linder, C. Porter, J. Di, S. Smith, and D. Thompson. “Side-channel attack mitigation using dual-spacer dual-rail delay-insensitive logic (D3L)”. In: SoutheastCon´10, 2010, pp. 471-474.

[20] J. Murphy and A. Yakovlev. “An alternating spacer AES crypto-processor”. In: ISSCC´06, 2006, pp. 126-129.

[21] S. Moore, R. Anderson, R. Mullins, G. Taylor, J. J. A. Fournier. “Balanced self-checking asynchronous logic for smart card applications”. Microproc. and Microsys., vol. 27, 2003, pp. 421-430.

[22] N. Ekekwe. “Power dissipation and interconnect noise challenges in nanometer CMOS technologies”. IEEE Potentials, vol. 29(3), 2010, pp. 26-31.