Impact of C-Elements in Asynchronous Circuits

Matheus Moreira, Bruno Oliveira, Fernando Moraes, Ney Calazans
Faculdade de Informática
Pontifícia Universidade Católica do Rio Grande do Sul
Porto Alegre, Brazil
{matheus.moreira, bruno.scherer}@acad.pucrs.br {fernando.moraes, ney.calazans}@pucrs.br

Abstract

Asynchronous circuits are a potential solution to address some of the obstacles in deep submicron (DSM) design. One of the most frequently used devices to build asynchronous circuits is the C-element, a device present as a basic building block in several asynchronous design styles. This work measures the impact of three different C-element types. The paper compares the use of each implementation to build a real case asynchronous circuit, an RSA cryptographic core, and reports results of precise electrical simulations of each C-element. Findings in this work show that previous results in the literature about C-element implementation types must be re-evaluated when using C-elements in DSM technologies.

Keywords

asynchronous circuits; standard cell; C-element; deep submicron

I. Introduction

The interest in non-synchronous circuits is increasing. The International Technology Roadmap for Semiconductors (ITRS) in its 2008 edition [1] describes a clear need for asynchronous communication protocols for control and synchronization in integrated circuits (ICs) for the next decades. However, one major problem for adopting the asynchronous design paradigm is that most commercial EDA tools assume that a single (or a few) clock signal(s) globally controls an entire IC. Moreover, basic components required to implement asynchronous communication, like the C-element, are not available off-the-shelf in typical standard cell libraries.

The C-element is a fundamental primitive for building asynchronous logic and implementing the synchronization required by most handshaking protocols, which provide the basis for asynchronous communication. Three static CMOS implementations are the Sutherland pull-up pull-down, the weak feedback and the van Berkel[1] implementation. Some previous work comparing C-element types also mention a fourth type, the dynamic C-element. Since the objective of this work is to investigate the use of C-elements in the general scope of asynchronous implementations, where no systematic refresh of dynamic circuits is generally available, this latter type is ignored here.

This work presents a comparison between the three first C-element mentioned implementations. To do so, these were designed and implemented as standard cells in the STMicroelectronics (STM) 65nm CMOS technology. Two different scopes served to compare the implementations, the single cell level and the application, or core, level. For the former scope, delay and power consumption were measured through electrical simulations at the standard cell level after electrical extraction of the physical layout. For the latter scope, the different C-elements were used to build asynchronous implementations of an oscillator ring and an RSA cryptographic core. A comparison of performance and required silicon area of the case studies made it possible to scrutinize the systemic effect of using different C-element implementations when designing asynchronous circuits in DSM technologies.

The rest of the paper is organized in seven sections. Section II describes basic concepts on asynchronous circuits and Section III presents a discussion on related works. Section IV presents details on the CMOS design and implementation of C-elements. Section V approaches the comparison of each implementation at the cell level. Section VI presents the case study circuits using each of the three C-element implementations. Finally, Section 0 draws some conclusions and directions for further work.

II. Asynchronous Circuits

A digital circuit is *synchronous* if its design 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 [2]. The resulting behavior is similar to a synchronous system where registers are clocked only when and where necessary. Characterizing an asynchronous design style requires: (i) the choice of a delay model, (ii) an information encoding method and (iii) a set of basic devices. Each of these are 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 *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 [2] are still used in specific contexts. This work assumes the use of QDI as target model.

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

[1] Some works in the literature call the van Berkel C-element *symmetric*, because of its special circuit topology and its transition effects. This paper avoids this nomenclature to prevent confusion with the behavioral variations of C-elements, which are named *symmetric* and *asymmetric or generic*.

used to synchronous techniques, the timing relationship between control and data signals need to be guaranteed at every handshake point, making design of large asynchronous modules difficult and hard to scale. 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. Throughout this work, circuits will employ dual rail encoding. More efficient DI encodings exist and are discussed in detail in several publications e.g. in [3].

Most of the asynchronous design techniques proposed to date require devices other than 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 this is inefficient. A fundamental device that enables to build such elements more effectively is the C-element. Its importance comes from the fact that C-elements operate as an event synchronizer. Figure 1depicts the truth table and state diagram for a C-element with symmetric behavior. Its output only switches when all inputs have the same logical value. When inputs A and B are equal, the output Q assumes this same value. However, when the 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 [4]. On the rest of this paper the discussion restricts attention to different CMOS implementation of the Figure 1 C-element.

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

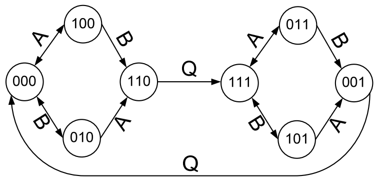


Figure 1 - A basic C-element truth table and state diagram for symmetric behavior with regard to the inputs.

III. Related Work

As far as the Authors could verify, only four works propose a comparison of different CMOS implementations of C-elements. In the first two works, Shams et al. [5] [6] present a comparison of four CMOS implementations of the C-element. Initially, first-order models are proposed to quickly approximate delay and energy dissipation of such C-elements. Next, CMOS implementations are designed in a 0.8 μm technology. Measurements of energy dissipation and delay through simulation showed to be in good agreement with the analytical predictions. The authors conclude that the Van Berkel C-element is the best option for energy-efficient, high-speed and high density circuits. However, the conducted study did not take into account layout considerations, and results obtained through analytical models do not agree with the more precise results presented here. Moreover, Shams et al. ignored wire loads and their effect on circuits during simulations, which is not acceptable in DSM technologies.

In [7], Elissati et al. conducted a study comparing four different CMOS implementations of the C-element for a specific case of application, self-timed rings. The work compares the performance of rings using each C-element implementation and through simulation for a 65nm technology. Results show that the van Berkel implementation seems to be the best trade-off between low power and operating speed. However, the work does not take into account any layout consideration either. Besides, the results presented are valid mostly for self-timed rings only. In [8], Bastos et al. evaluated transient-fault effects on four C-element implementations. The work shows that the weak-feedback C-element is the most transient-fault robust implementation. Moreover, results show that the van Berkel C-element is the fastest and lowest power and area consuming implementation. Albeit the work provides concrete results on the robustness of each implementation for transient faults, speed, power consumption and area results were not evaluated through a layout-aware approach. Wire loads and parasitic were also not taken into account during simulations.

This paper stands off by scrutinizing the electrical behavior of each C-element CMOS topology after the extraction of the physical implementation. Moreover, it presents a layout- and application-aware comparison of them in generic asynchronous circuits. Comparisons presented here are impartial, because the physical design of the C-elements is based in a parameterizable flow that allows the generation of different cells with the same driving capability. Results showed that, differently from previous works concluded, the van Berkel is not the best option for low area, low power and high-speed designs. An in-depth discussion on electrical design and simulation of C-elements is conducted to demonstrate that each C-element type is advantageous in some context. This is, as far as the Authors could verify, the first work to compare C-elements taking into account parasitic devices and wire loads, guaranteeing a fairer comparison.

IV. C-element CMOS Implementation

Three different static CMOS topologies of the C-element were implemented and compared in this work. Figure 2(a) shows the Sutherland pull-up pull-down implementation, proposed by Sutherland in [9] and employed in his micropipeline design method. Another C-element type, shown in Figure 2(b), was proposed by van Berkel in [10]. Finally, Figure 2(c) shows the weak feedback C-element, proposed by Martin in [11] and used extensively in the asynchronous microprocessors designed at Caltech. These three implementations were implemented in the scope of the present work as standard cells for a specially designed library [12].

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 (capability of charging/discharging a given load). To efficiently implement C-elements as standard cells in a real technology (in this case, STMicroelectronics 65nm), this work employs a specific design flow, detailed in [13]. The flow comprises three main steps, briefly described herein: specification, design and validation. Furthermore, it is parameterizable, depending on the required driving strength.

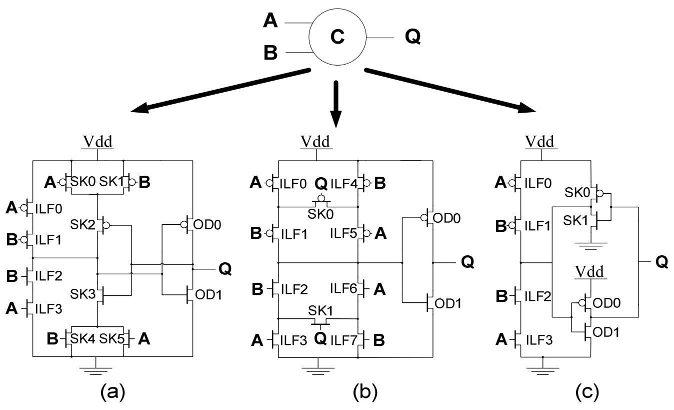


Figure 2 - CMOS implementations of the C-element: (a) Sutherland's pull-up pull-down, (b) van Berkel's, (c) Martin's weak feedback.

Specification includes defining the precise cell functionality and its electrical requirements. The goal of this step is to find optimal dimensions of PMOS and NMOS transistors to meet timing and power requirements for the cell. The flow proposes tools to automate transistor size generation, simulation and transistor sizing choice processes. The design phase, second step of the flow, consists in taking the selected transistor dimensions and producing a physical view (layout) that fulfills this specification. This includes drawing the cell in a layout editor for the chosen process, extracting parasitics and electrically characterizing the resultant circuit for the given fabrication process. The last flow step, validation, consists in checking if the information generated during electrical characterization is equivalent to that defined in the cell specification. Moreover, timing simulation must be carried for a design composed by a single cell to check if the delay of the electrical characterization is correctly annotated and the behavior is correctly implemented.

After verification, the cell can be used in a design. Besides the layout, two other views are produced: an abstract view employed by place and route tools and assemble the circuit on a chip; and a behavioral view, destined to support high level simulations. Both correspond to automated steps in the proposed flow.

V. Standard Cell Comparison

Each of the three implementations of the C-element was designed through the design flow mentioned in Section IV. 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 [14], which builts asynchronous circuits from a high level description language, Balsa. Therefore the complexity of designing asynchronous ICs is significantly reduced. Figure 3 shows an example layout for each of the designed standard cells.

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

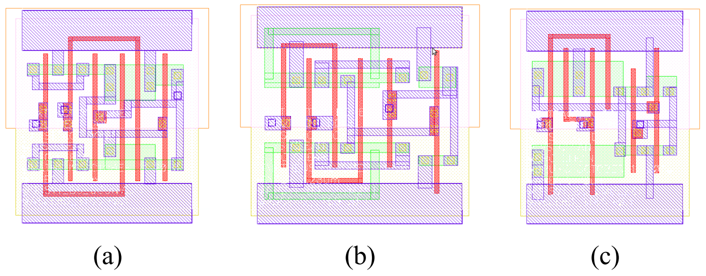


Figure 3 – Example physical layout at the cell level of the designed C-elements: (a) Sutherland's pull-up pull-down, (b) van Berkel's, (c) Martin's weak feedback.

TABLE I – Area and parasitic capacitance required by CMOS implementations of the C-elements (cap. obtained through RC extraction).

C-element Implementation	Cell Area (um^2)	Parasitic Cap. (fF)
Sutherland	6.24	6.066
van Berkel	7.28	9.383
Weak feedback	5.72	4.469

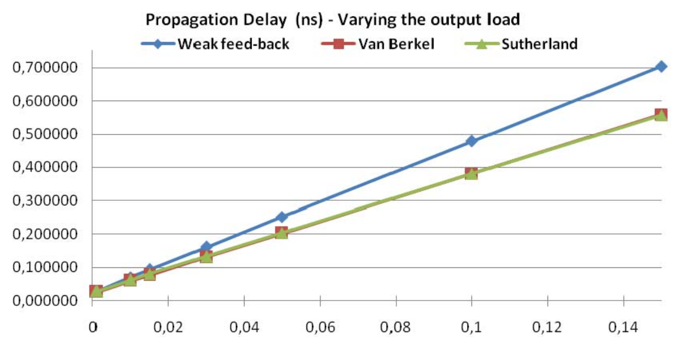
After electrical extraction, each cell was characterized for a typical fabrication process corner, with typical delay for the NMOS and PMOS transistors, for an operational condition of 25 degrees Celsius and 1 V power supply. Table II shows electrical results obtained through the electrical characterization. The weak feedback implementation presents the highest capacitance on its inputs. In fact, in comparison with the van Berkel implementation, which presents lower 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 3 shows.

As for the internal power required to switch the output of the standard cell, the Sutherland and the 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, weak feedback implementations have the highest leakage power consumption, when compared to the others. This is also a consequence of transistor size.

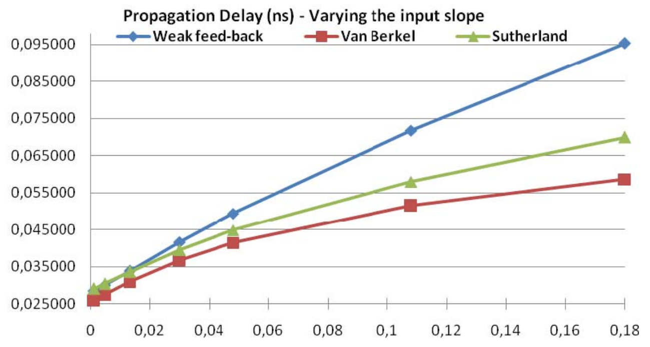
TABLE II - Capacitance of the input pins and power consumption of designed standard cells, after electrical extraction.

C-element Implementation	Input Capacitance (fF)		Average Internal Power[2] (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
Weak feedback	5.633	5.849	4.889	21.930	30.591

[2] 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.



(a)



(b)

Figure 4 - Propagation delay of the designed standard cells after electrical extraction, as a function of the (a) output load capacitance and (b) input slope. In (a), results were obtained by fixing the input slope in 1.2ps and varying the output load from 0.001pF to 0.15pF. In (b), results were obtained by fixing the output load in 1fF and varying the input slope load from 0.0012ns to 0.180ns.

The average propagation delay of each cell, measured as the average delay of rise and fall transitions, was also obtained through electrical characterization. Figure 4 shows the obtained results for two scenarios. In Figure 4(a), 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 the van Berkel implementations 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 implementations. This behavior shows that the weak feedback is also the implementation most sensitive to output load variations.

Figure 4(b) shows the electrical behavior of each implementation when the output load is fixed in 1fF and the input slope varied from 0.0012ns to 0.180ns. In this scenario, the fastest implementation is the van Berkel, followed by the Sutherland and next the weak feedback. The obtained results show that for small input slopes (0.0012ns to 0.0132ns), the speed of Sutherland and weak feedback C-elements is equivalent. However, as the input slope gets more significant, the propagation delay of the weak feedback C-element gets worse. In this way, the weak feedback implementation is, also, the most sensitive to input slope variations. As Figure 4(b) 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 [10], which conducted an analysis of the threshold of C-elements.

VI. C-ELEMENTS ON ASYNCHRONOUS CIRCUITS

A. Oscillator ring case study

As an initial comparison of the impact of each C-element implementation on asynchronous circuits, a low complexity circuit was described in the SPICE language, an oscillator ring. As Figure 5 shows, 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.

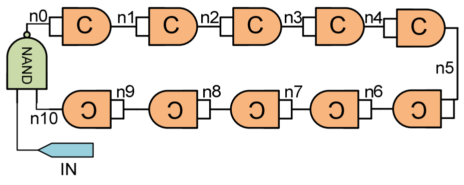


Figure 5 - Oscillator ring employed for an initial comparison of the impact of each C-element implementation in an asynchronous circuit.

After simulating each oscillator ring, power consumption and operational frequency, were obtained through the SPICE ".measure" function, as Table III 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 III - 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
Weak feedback	0.808	0.28	119.20

As Table III 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 the Sutherland C-element. The weak feedback implementation presents higher dynamic and leakage power, as expected. Power figures enforce the statement that the weak feedback is the most power consuming and the Sutherland and van Berkel present similar power consumption figures.

It would be expected that at least two of the rings, the ones composed by the Sutherland and the van Berkel implementation, presented equivalent operating frequency. However, the one generated with van Berkel C-elements operates roughly 32% and 42% faster than the one composed by Sutherland and weak feedback C-elements, respectively. As Table II shows, the sum of the pin capacitances of the Sutherland C-element is 8.975fF, while for the van Berkel implementation 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-elements 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 6 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.

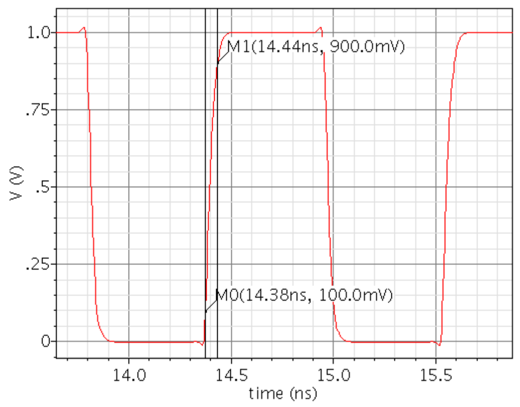


Figure 6 - Input slope of a single cell of an oscillator ring composed by Sutherland C-elements.

For the weak feedback C-element, the operational frequency range is even worse. That is due to the fact that this implementation presents not only higher input capacitance, which contributes for an elevated slope in the input of each cell of the ring, but it is also much more sensitive to input slope variations than the Sutherland C-element. Its performance would be yet worsened if, for instance, each C-element was required to drive a load bigger than 0.015pF. See Figure 4(a), where the delay of the weak feedback implementation starts to get worse than the other two. In this case, the sum of its input pins, load that each cell (except the one that drives the NAND) must drive, is 11.482fF.

B. *RSA cryptographic core case study*

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 [2] and synthesized through the Teak System [14]. Teak automatically maps the Balsa description into a specific set of cells, a group of C-elements with different functionalities, generating asynchronous QDI circuits. ASCEnD [13], a standard cell library composed by asynchronous components, was designed to support Teak synthesis for the 65nm STMicroelectronics CMOS technology. The library contains the required C-elements for the three CMOS topologies compared in this work.

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 [2]. 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. Physical cells are the cells used to connect the power lines to the substrate, tap cells, and the filler cells employed in the core. 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. This result shows the importance of this device in typical asynchronous design. For these examples, roughly 40% of the required logical standard cells were C-elements.

Table IV 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 IV - 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. Thesr comprised multiple cryptographic operations for each netlist, collecting performance results. Employed operational conditions were 25ºC, 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 implementation 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 delay 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 V 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 smaller portion of the total power than the dynamic power consumption. Notably, weak feedback was the less power efficient implementation.

TABLE V - 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 presented worse dynamic power consumption than the van-Berkel C-element. However, in that case, each cell was driving a single two-input cell, while in the circuit generated by Teak, the cells were required to drive multiple nets and, consequently, higher loads. In this scenery, the van Berkel dynamic power efficiency was compromised.

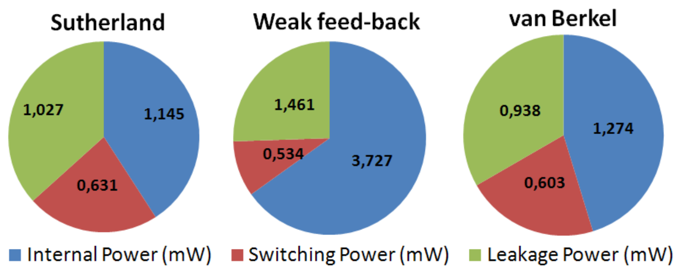


Figure 7 - C-elements power consumption for each asynchronous RSA cryptographic core implementation.

Figure 7 shows details the power consumption of the C-elements from the total power consumed by the placed and routed netlists. The total power consumed by the Sutherland, the weak feedback and the van Berkel C-elements, in their respective netlists, was 2.803mW (54%), 5.722mW (73%) and 2.815mW (54%), respectively. These results show that, in a realistic application, 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 that the transistors are in short circuit when switching the output of the C-element.

VII. CONCLUSIONS AND FUTURE WORK

This paper presented a comparison of 3 different CMOS implementations of C-elements: Sutherland, van Berkel and weak feedback. The comparison considered layout effects on a DSM technology, together with the systemic effect of using these C-elements in building asynchronous cores. 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 bigger portion of the total power 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 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. Future work includes prototyping the circuits designed as case studies in the STM 65nm technology in order to evaluate and compare the C-elements on silicon. In this way, information about the robustness and the effect of process variations, for the different implementations can be obtained and compared. Moreover, a study is under way to scrutinize the use of different C-element implementations in a single design in order to generate hybrid and optimized designs.

ACKNOWLEDGMENTS

This work is partially supported by the CAPES-PROSUP, the CNPq-PIBIC Program, and by the FAPERGS (under grants 11/1445-0 and 10/0814-9). Professors Moraes and Calazans also acknowledge the CNPq support under grants 301599/2009-2 and 309255/2008-2, respectively. Authors would like to acknowledge the support granted by CNPq to the INCT-SEC (National Institute of Science and Technology – Critical Embedded Systems – Brazil), process no. 573963/2008-8.

References

[1] Semiconductor Industry Association, "The International Technology Roadmap for Semiconductors", ITRS 2008 Edition, 2008.

[2] J. Sparsø and S. Furber, "Principles of Asynchronous Circuit Design – A Systems Perspective", Kluwer Academic Publishers, Boston, 360 p., 2001.

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

[4] W. B. Toms, "Synthesis of Quasi-Delay-Insensitive Datapath Circuits", PhD Thesis, University of Manchester, 237 p., Feb. 2006.

[5] M. Shams, J. C. Ebergen, and M. I. Elmasry, "A comparison of CMOS implementations of an asynchronous circuits primitive: the C-element", in: International Symposium on Low Power Electronics and Design (ISLPED), pp. 93-96, Aug. 1996.

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

[7] O. Elissati, E. Yahya, S. Rieubon, and L. Fesquet, "Optimizing and Comparing CMOS Implementations of the C-element in 65nm Technology: Self-Timed Ring Case", in: International Workshop on Power and Timing Modeling (PATMOS), pp. 137-149, Sep. 2010.

[8] R. P. Bastos, G. Sicard, F. Kastensmidt, M. Renaudin, and R. Reis, "Evaluating transient-fault effects on traditional C-element's implementations", in: International On-Line Testing Symposium (IOLTS), pp. 35-40, Jul. 2010.

[9] I. E. Sutherland, "Micropipelines", Communications of the ACM, 32, pp. 720-738, Jun. 1989.

[10] K. van Berkel, "Beware the isochronic fork", Integration, the VLSI journal, 13(2), pp. 103-128, Jun. 1992.

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

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

[13] M. T. Moreira, B. S. Oliveira, J. J. H. Pontes, and N. L. V. Calazans, "A 65nm Standard Cell Set and Flow Dedicated to Automated Asynchronous Circuits Design", in: 24th IEEE International SoC Conference (SoCC), pp. 99-104, Sep. 2011.

[14] A. Bardsley, L. Tarazona, D. Edwards, "Teak: A Token-Flow Implementation for the Balsa Language", in: International Conference on Application of Concurrency to System Design (ACSD), pp. 23-31, Jul. 2009.