Adapting a C-element Design Flow for Low Power

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

***Abstract* - The interest in non-synchronous design of digital circuits is growing due to technology scaling into deep submicron transistor geometries and to the problems this scaling causes to keep synchronous design advantageous. To enable most non-synchronous styles, the C-element is a fundamental device that has to be available as logic primitive. A recently proposed design flow improved a standard cell library, adding to it a set of typical asynchronous cells. However, the original flow did not address low power cells explicitly, which is a requirement in many modern applications. This paper proposes the extension of the flow so that it can expand the cell set with low power components. To achieve this, the paper adds a new degree of freedom to cell design. The new standard cell set encompasses over 500 different C-element implementations. The cell set employs a 65nm commercial CMOS process and is fully compliant with the foundry standard cell library. A fully asynchronous RSA crypto core was designed with the new cells, producing savings of more than 35% in total power and more than 69% in leakage power.**

Keywords - asynchronous design; low power; C-element.

I. INTRODUCTION

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. For example, they estimate that ICs based on global clocks, 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 and globally asynchronous, locally synchronous (GALS) ICs. This is a consequence of the scaling down of CMOS technologies.

A problem to adopt non-synchronous design paradigms is that most EDA tools assume a single clock signal globally controls an entire IC or even an entire system. Efficient implementations of non-synchronous circuits are hard to obtain using such tools. Moreover, non-synchronous paradigms are in fact not a single alternative to their synchronous counterpart, but a set of methods, each with its own needs and constraints. Thus, efficient EDA tools for non-synchronous circuits must support different design methods. Several tools exist, which provide solutions for designing non-synchronous circuits than full custom- or schematic-based approaches.

The C-element is a fundamental primitive for building asynchronous logic and supporting the local synchronization, required by several handshaking protocols, since it acts as an event synchronizer. Several variations of this device exist, each useful for specific design features. Variations involve device behavior, logical, physical and electrical requirements. In a previous work [2], the authors described a new standard cell set and its design flow. These allow enhancing traditional standard cell libraries with asynchronous primitives, especially C-elements. But that work overlooked power dissipation in the new cells. This work suggests adding a new degree of freedom in designing C-element cells, namely the choice between **best for speed** and **best for power** versions for each cell. Note this work *is not* about automated circuit generators. It describes the process of producing standard cells that are to become part of a static library that includes foundry-provided standard cells as well as specific cells to support asynchronous design.

II. NON-SYNCHRONOUS CIRCUITS

A digital circuit is *synchronous* if its design implies the use of a single clock signal to control all of its events. Otherwise the circuit is called *non-synchronous*. As a special case of the later, a digital circuit is *asynchronous* when no clock signal is used to control any sequencing of events. These employ explicit handshaking between their components to synchronize, communicate and operate [3]. 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) choice of a *delay model*; (ii) a *data encoding* method; and (iii) a set of basic *devices.* This Section explores the first two items; Section III discusses the last item.

Asynchronous circuits can be classified according to several criteria. One important criterion is the sensibility to delays in wires and gates. The most robust and restrictive delay model is the *delay-insensitive* (DI) model. DI circuits operate correctly regardless of any gate or wire delays. Unfortunately, this class is too restricted. The addition of an assumption on wire delays in some carefully selected forks enables the *quasi-delay-insensitive* (QDI) model and 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* [3] are used in specific contexts. This work deals only with QDI design.

There are different ways to encode data to support different delay models. The use of *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 constrains between control and data signals need to be guaranteed at every handshake point, making design of large asynchronous modules unfeasible. As an alternative, *DI encodings* are robust to wire delay variations, because control signals are embedded within data signals. An example is *dual-rail* encoding, where request signals are computed at the data destination from the data lines. All experiments in this work employ dual-rail data encoding. More efficient DI encodings exist, as described e.g. in [4] and [5].

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

(a)

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

(b)

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

(c)

Figure 1 – Examples of C-element behaviors and symbols. (a): a 2-input, symmetric C-element; (b) and (c): two 2-input, asymmetric C-elements.

III. THE 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, C-elements, etc. Although most of these may be built from logic gates this is often quite inefficient. C-elements and ordinary logic gates enable to effectively build most asynchronous elements [3]. C-elements are sequential logic devices that operate as event synchronizers. Figure 1 depicts some types of C-elements. The Figure shows truth tables and associated symbols for 2-input implementations only. Figure 1 (a), for instance, shows a *symmetric* C-element, where the output only switches when all inputs are at the same logic value. When inputs A and B are the same, output Q assumes this value. When inputs are different, Q keeps its previous value.

Versions with distinct behavior exist, where one or some of the inputs of the C-element may interfere only in the 1→0 or the 0→1 transition of the output. The importance of such variations in the synthesis of asynchronous circuits are discussed e.g. in [6]. These C-elements may be called *asymmetric*, *unbalanced* or *generalized*. Figure 1 (b) and 1 (c) show examples of such devices. In Figure 1 (b), input A can only interfere in the 0→1 switching of the output (1→0 transitions occur in response to a 1→0 transition in input B). This behavior is indicated in the symbol by a "+" in the asymmetric input. Next, Figure 1 (c) shows a device where input B can only interfere in the 1→0 switching of the output, where the "-" symbol in this input denotes the asymmetric behavior. This means that the output will switch to 0 if input A switches to 0, regardless of the value in input B.

IV. C-ELEMENTS DESIGN FLOW AND CELL SET

The newly proposed standard cell design flow for C-elements counts 3 steps: specification, design and validation.

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 (and the rate of their sizes), in order to meet timing and power requirements for the cell. As an example, Figure 2 presents three CMOS schematic implementations for the 2-input C-element cell from Figure 1(a). Figure 2 (a) shows the static C-element presented by Sutherland in [7], (b) presents the pseudo-static version proposed by Martin in [8], and (c) displays the schematic proposed in [9] by van Berkel. Each implementation is advantageous under specific requirements. A detailed discussion about these C-element topologies can be found in [11]. The structure of these or any other C-element variation can be divided into blocks (1), (2) and (3), displayed in Figure 2. Block (3) is the state keeper, block (2) is the output driving inverter and block (1) is the inverted logic function.

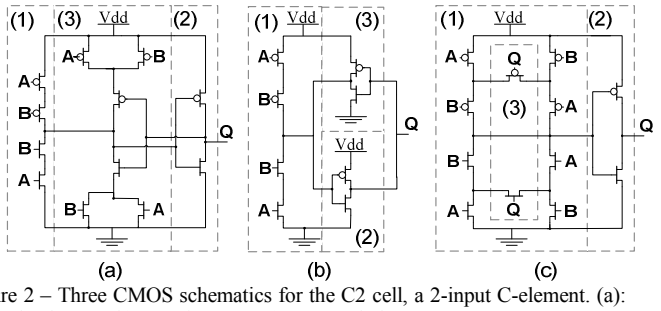

Figure 2 – Three CMOS schematics for the C2 cell, a 2-input C-element. (a): Sutherland's [7]; (b): Martin's [8]; (c): van Berkel's [9].

For each block, the flow proposes a distinct transistor sizing technique. The transistor sizing methods details can be found in [2].

The most complex sizing process refers to block (1) transistors, which requires exhaustive electrical simulations and produces a large amount of results. The flow proposes tools to automate transistor size generation, simulation and transistor sizing choice processes. The original flow uses a tool called *Cell Specifier* (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). As one of its major contributions, this paper suggests an enhancement by modifying CeS to output another implementation choice. The tool now selects the best speed and power consumption tradeoffs and presents it in a chart. In this way designers can choose for the BS or the implementation more adequate for low power consumption, called **Best for Power** (BP).

Figure 3 presents an example output of the new CeS for an instance of a 2-input Martin's C-element with driving strength X4. In this case, a total of 1219 simulation scenarios were automatically generated and simulated, each obtained by varying dimensions of block (1) (see Figure 2) transistors for this C-element. 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.

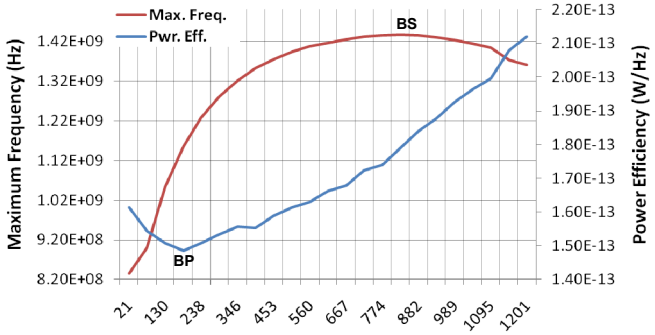

Figure 3 – Example of CeS output for the example 2-input Martin C-element.

As Figure 3 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 I 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 I. 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 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 chosen process. Figure 4 (a) and (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.

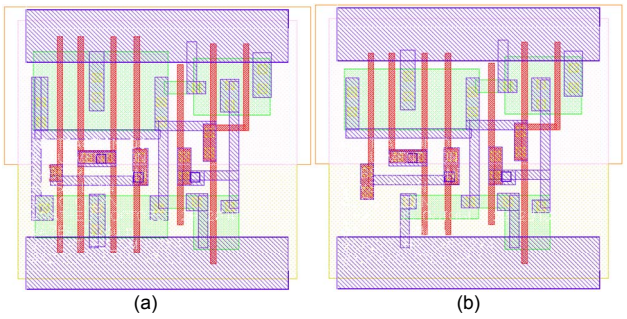

Figure 4 – Physical layouts for: (a) one BS and (b) one BP implementation of the 2-input, X4 Martin C-element (See Figure 2(b)).

The last step, validation, checks if information produced during electrical characterization is the same as that defined in the cell specification. Moreover, timing simulation is carried for a design composed by a single cell to check if the delay of the electrical characterization is correctly annotated and behavior is correctly implemented. If the cell passes this step, it can be used in a design. Besides layout, two other views are produced: an abstract one, used by place and route tools to assemble a circuit from cells; and a behavioral one, used in HDL-level simulations. Both are automated steps of the flow.

A. *The Cell Set*

A cell set containing 504 C-elements was created for the STMicroelectronics (STM) 65nm, general purpose, standard V_t technology. Figure 5 presents the composition of this cell set. Each cell is the result of choices on six parameters: driving strength, number of inputs, number/kind of asymmetric inputs, CMOS transistor topology, speed/power trade-off and choice of initialization control signals.

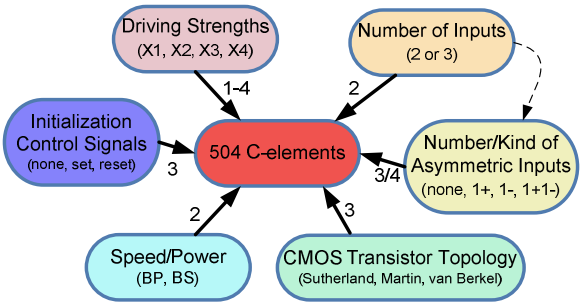

Figure 5 – Composition of the implemented set of C-elements cells. Each external rounded corner rectangle represents one parameter choice. Edges are labeled with the number of choices the parameter implies. The dotted edge represents dependencies between parameters choices.

Most parameters are orthogonal to each other. An exception is the influence of the number of inputs on the choice of asymmetric inputs. 2-input cells may have none, one + or one – (3 choices) asymmetric inputs (see Figure 1). 3-input cells may also have one + and one - asymmetric inputs (4 choices).

The output driving strength (amount of load that a cell is able to charge/discharge) varies from X1 to X4, with power consumption, area and driving capability increasing from X1 to X4. This is compatible with the STM standard cell library.

The cell set enables the choice of the three static topologies displayed in Figure 5. Dynamic topologies have limited application and are, accordingly, ignored here. The different static topologies are available to trade power consumption, area and operating speed.

Each C-element cell can present none, one **set** or one **reset** initialization signal. As the main contribution in this paper, each cell may now be either BP or BS.

V. CASE STUDY

A 32-bit public key RSA crypto core was described in the Balsa language [3] and implemented using the new cell set. The choice of this circuit was due to the fact that the underlying algorithm employs several distinct arithmetic operations, as well as control circuits. The former operations are implemented through delay insensitive minterm synthesis (DIMS) logic, which is basically composed by C-elements [3]. In this way, the impact of the proposed cell set on the circuit is more significant and can thus be efficiently evaluated.

The circuit was synthesized with the design tool Teak [10] for two C-element versions, BP and BS. The C-element topology choice was the van Berkel in both cases. 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 STM and those of the proposed cell set. Table II 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 II. STANDARD CELL AREA AND WIRE LENGTH FOR THE ASYNCHRONOUS IMPLEMENTATIONS OF THE RSA IN STM 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 these 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 STM 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 the 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 through the simulation of an oscillator ring, in Section IV, 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 III 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 25º 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 processes [1]. 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 III. POWER CONSUMPTION FOR THE ASYNCHRONOUS IMPLEMENTATIONS OF THE RSA IN STM 65NM TECHNOLOGY EMPLOYING BS AND BP VERSIONS OF C-ELEMENTS.

	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

VI. CONCLUSIONS AND FUTURE WORK

C-elements are fundamental primitives in asynchronous circuits and need to be supported at a low level of abstraction. BS and BP versions of the cell set proposed here 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 much more 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. This is ongoing work. Another ongoing work is prototyping circuits with this enhanced cell set in order to validate and evaluate the new cell set on silicon.

VII. ACKNOWLEDGEMENTS

This work is partially supported by the CAPES-PROSUP, the CNPq-PIBIC Program, the CNPq (under grants PNM 551473/2010-0, 301599/2009-2), and by the FAPERGS (under grants 11/1445-0 and 10/0814-9). Authors would also like to acknowledge the National Science and Technology Institute on Embedded Critical Systems (INCT-SEC) for the support to this research.

REFERENCES

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

[2] Moreira, M. et al., "A 65nm Standard Cell Set and Flow Dedicated to Automated Asynchronous Circuits Design," in SoCC, pp.99-104. 2011.

[3] J. Sparsø and S. Furber, "Principles of Asynchronous Circuit Design – A Systems Perspective," Kluwer Ac. Pub., Boston, 354p., 2001.

[4] T. Verhoeff, "Delay-insensitive codes: an overview" Distributed Computing, 3(1), pp.1-8, Mar. 1988.

[5] M. Agyekum and S. Nowick, "An error-correcting unordered code and hardware support for robust asynchronous global communication," in DATE, pp.765-770, 2010.

[6] W. B. Toms, "Synthesis of Quasi-Delay-Insensitive Datapath Circuits," PhD thesis, Univ. of Manchester, 237 p., Feb. 2006.

[7] I. E. Sutherland, "Micropipelines," Communications of the ACM, 32(6), pp.720-738, Jun. 1989.

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

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

[10] A. Bardsley et al., "Teak: A Token-Flow Implementation for the Balsa Language," in ACSD, pp.23-31, Jul. 2009.

[11] M. Shams et al., "Modeling and comparing CMOS implementations of the C-element," IEEE Trans. VLSI Syst., 6(4), pp.563-567, Dec. 1998.