Design of NCL Gates with the ASCEnD Flow

Matheus T. Moreira ‡†, Carlos H. M. Oliveira ‡, Ricardo C. Porto ‡, Ney L. V. Calazans ‡

‡Pontifical Catholic University of Rio Grande do Sul
Faculty of Computer Science – Porto Alegre, Brazil
{matheus.moreira, carlos.oliveira.004,
ricardo.porto}@acad.pucrs.br, ney.calazans@pucrs.br

†University of Santa Cruz do Sul
Department of Informatics – Santa Cruz do Sul, Brazil
matheustrevisan@unisc.br

Abstract— **Silicon technologies advances brought the possibility of integrating billions of transistors in a die. However, as transistors get smaller, some of the aspects that were negligible in previous technologies emerge as difficulties for the design in current and future technology nodes. In this context, fully synchronous circuits are harder to be built, as timing closure constraints become difficult to be met, and the asynchronous paradigm gains interest in the research community for its ability to cope with current technologies issues. ASCEnD was proposed as a standard cell library for supporting standard-cell based design of asynchronous circuits and comprises a design flow for asynchronous components. This work presents the use of the ASCEnD flow to design NCL gates, which enable design improvement opportunities for some asynchronous templates. A total of 14 different NCL gates were designed at the layout level and had their electrical behavior characterized. As a result, electrical and physical models of these gates are now part of the ASCEnD library.**

Keywords- standard cell library, asynchronous circuits, null convention logic.

I. INTRODUCTION

The synchronous paradigm prevails for integrated circuits design. This is due to the high abstractions levels achieved when assuming that time is a discrete variable controlled by a *clock* signal, the main characteristic of this paradigm. However, as silicon technologies evolved into deep-submicron nodes, synchronous design starts to become more constrained [1]. In fact, the ITRS [2] in its 2011 edition suggests that further advances in VLSI systems need a shift of the paradigm. By 2020, 40% of the intrachip communication is expected to use asynchronous mechanisms. Thus, asynchronous techniques start to gain relevance.

However, there is still little design automation for asynchronous circuits. EDA tools adapted to deal with asynchronous design form a very limited set of resources. Even worse, basic components required to build these circuits are usually not available in typical standard cell libraries. In this way, semi-custom asynchronous design is not available and designers have to rely on full-custom approaches to build asynchronous systems.

The ASCEnD library [3] [4] is a freely available standard cell library of asynchronous components designed to help supporting semi-custom asynchronous circuits and systems design. The library was first designed in the STMicroelectronics 65nm CMOS technology (ASCEnD-ST65), and is currently being ported to the IBM 130nm CMOS technology. Coupled to the library, there is a design flow, called the ASCEnD flow, used to design the library cells. This flow is semi-automatic, based on a set of specially designed tools. Currently, the only manual step is the cell layout drawing.

This work presents an extension of ASCEnD-ST65 to support the Null Convention Logic (NCL) style [5], with the goal of providing increased support to additional asynchronous templates and allows better design space exploration for applications. A set of 14 different NCL gates was designed at the layout level through the ASCEnD flow. Besides, post-layout simulations were performed to validate the designed gates. A mixed signal simulation environment was the source to conduct a precise timing analysis and power characteristics of the NCL template and to compare it with the delay insensitive minterm-synthesis (DIMS) template. Results show substantial improvements in terms of power consumption and operating speed, which illustrates the design improvement opportunities obtained for semi-custom designs by adding NCL to the ASCEnD library.

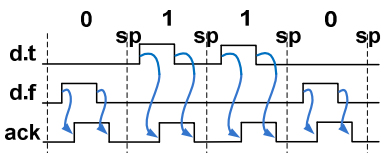
II. ASYNCHRONOUS CIRCUITS

A digital circuit is synchronous if its operation is controlled by the use of a single clock signal. Otherwise it is called asynchronous. Asynchronous circuits employ explicit handshaking among their components to synchronize, communicate and operate [1]. Characterizing an asynchronous design style 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.

The most robust and restrictive delay model is the delay-insensitive (DI) model [6], 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) class [7]. Here, signal transitions must occur nominally at the same time only at each end point of the mentioned forks. QDI circuits are, currently, the most used class of asynchronous circuits [8].

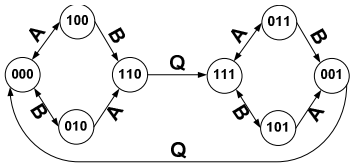
In QDI circuits, data is encoded through DI codes, to guarantee their robustness to delay variations. The most used DI codes belong to the m-of-n class [1] [6] [8]. These consist of all n-bit code words where exactly m bits are at logic ‘1’ and all other (n-m) bits are at logic ‘0’. In circuits that use m-of-n codes, the request signal to communicate data is encoded in the data itself and, therefore, requires extra hardware for detection. A specific case of m-of-n code is the dual-rail (DR) code or 1-of-2. This code uses two wires to represent an information bit, here called d.t and d.f.

978-1-4673-4900-0/13/$31.00 ©2013 IEEE



Value	d.t	d.f
Spacer	0	0
Logic 0	0	1
Logic 1	1	0
Invalid	1	1

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



(a) (b) (c) (d)

Figure 1 – 4-phase DR data transmission (a) and wire encoding (b); basic C-element truth table (c) and state diagram (d) (values inside states are ABQi).

Regardless the data encoding scheme, handshake protocols can be classified in 2-phase or 4-phase, as discussed in detail in [1]. The first is often based on wire transitions identifying values, while 4-phase handshaking usually assumes that wire logic level combinations define data values and transitions are required for synchronization. Usually, 2-phase protocols enable faster speeds but consume more hardware than 4-phase protocols.

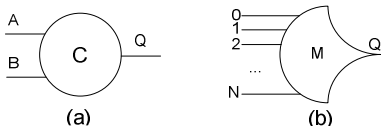
When DR codes are associated to a 4-phase handshake protocol, all communications start with all wires at a predefined value (usually logic '0'), called *spacer*. Next, after each valid value is acknowledged by a receiver, all wires must return to the spacer value. Figure 1 (a) shows an example of data transmission with this protocol, where typical values are represented in Figure 1 (b). In the displayed waveform, the first propagated value is a logic '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, represented in this case by d.t=0 and d.f=0. Next, the acknowledge signal switches to logic '0', and a new transmission can initiate.

To synchronize data transmission, asynchronous circuits using m-of-n codes require devices other than ordinary logic gates and flip-flops available in current commercial standard cell libraries. These include e.g. asynchronous registers, event fork, join and merge devices [1]. Although most of these may be built from logic gates this is often 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 event synchronizers. Figure 1(c) depicts the truth table in and state diagram for a basic 2-input C-element appears in Figure 1 (d). Figure 2 (a) shows the associated symbol.

The output of a C-element may only switch when all inputs are at the same logic value. For instance, in a 2-input C-element, 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 at the output. This enables ordering the occurrence of input events. Moreover, C-elements allow the implementation of Boolean logic without losing delay insensitivity through e.g. the DI minterm synthesis (DIMS) logic style [1].

III. Null Convention Logic

Threshold gates with hysteresis, also called M-of-N gates, form a class of circuit elements that have important application in NCL [5], which have been successfully employed for implementing asynchronous systems on silicon. Figure 2 (b) shows their associated symbol, where N is the number of inputs and M the threshold of the gate. The output will switch to logical 0 when all inputs are at logical 0 and to logical 1 when at least M (the threshold) of its N inputs is/are at logical 1. Otherwise, it will keep its state, similarly to a C-element with different values at its inputs. In fact, for N>1, an N-of-N gate is a N-input C-element.



(a) (b)

Figure 2 – (a) Symbol of a 2-input C-element and (b) symbol of an M-of-N gate.

Figure 3 (a) shows the CMOS schematic of an N-of-N threshold gate, or an N-input C-element. Figure 3 (b) shows the CMOS schematic of a 2-of-3 gate, to demonstrate more generally the structure of M-of-N threshold gates. Output "Q" will only be logical 0 if the inputs are all at logical 0. However, as the schematic shows, if any two or more inputs are at logical 1, the output becomes logical 1. These gates allow more efficient circuit implementations in terms of operating speed and power consumption than those obtained by using the DIMS logic style.

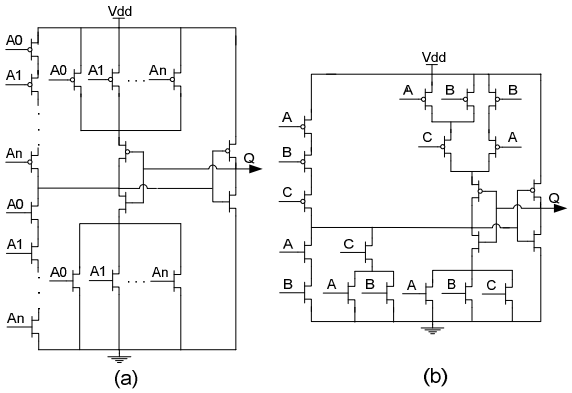


Figure 3 – Examples of NCL gates: (a) an N-of-N NCL gate, or a C-element; (b) 2-of-3 NCL gate [5].

IV. The ASCEnD Flow and Library

The ASCEnD flow was proposed to automate the design of the ASCEnD library standard cells. It has 3 basic steps: specification, design and validation, as detailed in [3].

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. To do so, a specially designed tool called RoGen is employed to generate a simulation circuit described in SPICE. This tool is parameterizable through a configuration file, which allows it to be used in any CMOS technology and provides good design space exploration capabilities to the designer. The circuit generated

by RoGen is then simulated, power and timing measurements are extracted and annotated to a text file. Next, another specially designed tool called CeS automatically selects the best transistor dimensions, according to a cost function.

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.

The last step 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 the layout, two other views are produced: an abstract one, used for place and route, and a behavioral one, used in HDL-level simulations.

Currently, the ASCEnD-ST65 library contains only C-elements, which limits its automated use to DIMS logic styles. However, DIMS logic is classically known as being too area and power consuming and by presenting inflated delay in combinational logic blocks. This is due to the fact that there is a need of employing C-elements for computing each minterm and OR gates to select values, before generating the output. In this context, NCL gates offer a greater possibility for improving design quality.

The design of NCL gates with the ASCEnD flow is straightforward. To do so, RoGen is configured as if the N-of-M gates were an M input C-element. The output inverter is dimensioned according to the sizes of the transistors of a similar drive inverter of the core library. The feedback inverter employs minimum-size transistors, to reduce resistive effects. The transistors of the logic stack have their dimensions automatically varied by RoGen. The resulting circuit is simulated and results are analyzed by CeS. To select the best transistors dimension, the employed cost function is given as the maximum frequency (Fr) at which the cell can operate divided by its average dynamic power consumption (DynPwr). In other words, transistors are dimensioned to the best Hertz per Watt compromise. Using this flow, a set of 14 NCL gates which allow the implementation of a wide range of functionalities was designed. Figure 4 shows the physical layout of an example M-of-N threshold gate, drawn to fulfill the parameters obtained by CeS.

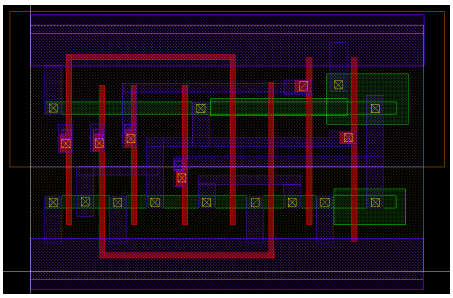


Figure 4 – Layout of a 1-of-3 NCL gate.

V. EXPERIMENTS AND RESULTS

To validate the new set of NCL components of ASCEnD and to evaluate the improvements that can be obtained for asynchronous logic when employing it, a case study of a dual-rail 4-phase handshake 32-bit ripple carry adder was implemented and simulated in two versions: DIMS and NCL. The choice is justified by the fact that the adder employs only combinational logic, to what NCL logic is mostly used. Also, for the DIMS-based logic, three different types of C-elements were employed, generating three versions of adders: van Berkel, Sutherland and Martin. All of these are available in the ASCEnD-ST65 library and details of their implementation can be found in [9].

Figure 5(a) and Figure 5(b) show the gate level schematics of DIMS-based half and full adders, respectively. Figure 5(c) and Figure 5(d), in turn, show the gate level schematics of the corresponding NCL circuits. The 4 versions of these adders, 3 DIMS and 1 NCL, were implemented in SPICE instantiating the standard cells of ASCEnD-ST65 and the core library, according to the schematics of Figure 5, after layout extraction. These were employed to generate the 32-bit ripple carry adders, all in SPICE.

Inspection of Figure 5 shows that NCL logic clearly requires fewer gates than DIMS logic. The transistors count of the NCL half and full adders are 80 and 128, respectively. For the DIMS adders, transistor count depends on the choice of C-element. For the van Berkel, Sutherland and Martin implementations the half and full adder transistors count is, respectively: 68 and 264, 68 and 168 and 52 and 120. All adders employed general purpose standard threshold transistors from the STMicroelectronics 65nm CMOS technology.

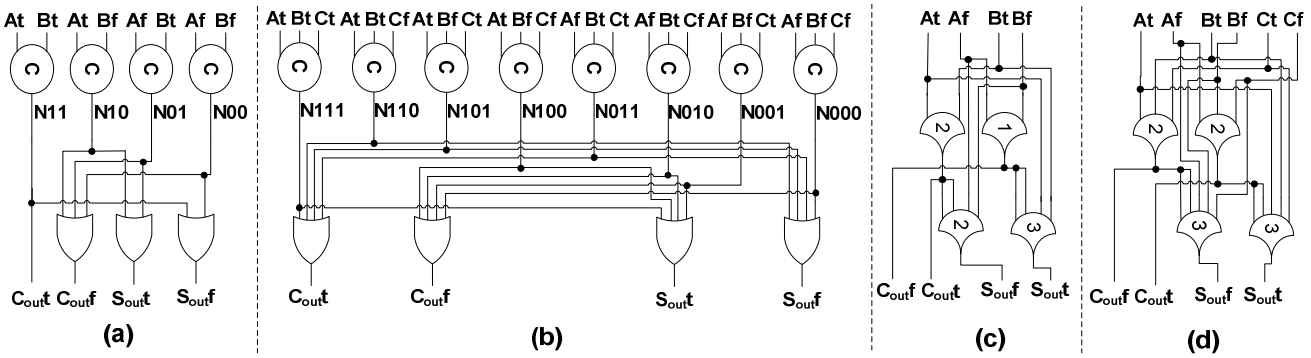


Figure 5 – Schematics of adders: (a) DIMS half adder; (b) DIMS full adder; (c) NCL half adder; (d) NCL full adder.

Simulation included a mixed-signal environment with a VHDL testbench instantiating adders described in SPICE and a verification block described in SystemC. A VHDL-AMS description was used for converting analog to digital signals and vice-versa, allowing the test bench to communicate with the SPICE circuits. This environment is useful for employing the high abstraction of SystemC to verify the electrical behavior of adders, while maintaining the precision of electrical simulation. The employed simulators for digital and analog circuits were Cadence Incisive and Spectre, respectively.

The SystemC verification block consists of a random data generator for feeding the inputs of the adders and a data checker to verify its correct behavior. Also, this block measured average, best and worst case delays for performing the sum of two values for forward delay and transmission delay, as well as the throughput (operations per µs). Dynamic and static power consumptions were measured by analog simulation. The former is given as the average power consumption of the adder when operating and the latter as the average power consumption when idle (filled with spacers). Results are summarized in Table I.

As Table I shows, the dynamic power consumption measured for the NCL adder is, in the worst case, roughly 3.5 times bigger than that presented by the DIMS-based adders. However, the throughput of the NCL adder is roughly 4 times larger in the best case. This leads to smaller energy per operation, roughly 0.981 for the NCL and 1.401 for the DIMS in the worst case. This represents an energy reduction of 30% for NCL. Also, this adder presents a throughput per µW relation of 1.02, while the DIMS adders present in the best case 0.883. This type of comparison is very useful for asynchronous circuits, because it normalizes delay and power in a cost function.

Table I – Simulation results for the adders, DIMS-VB (van Berkel), DIMS-SU (Sutherland) and DIMS-MA (Martin) and one NCL-based.

	DIMS-VB	DIMS-SU	DIMS-MA	NCL
Dynamic Power (µW)	156.349	155.561	197.43	541.37
Leakage Power (nW)	13.368	13.370	15.264	3.857
Throughput (operations/µs)	138	137	141	552
Throughput/µW	0.883	0.881	0.714	1.02
Energy per operation (pJ)	1.133	1.135	1.401	0.981
Best case forward delay (ps)	2822	2840	3092	295
Average forward delay (ps)	2842	2877	3138	395
Worst case forward delay (ps)	2897	2913	3183	968
Best case transmission delay (ps)	7063	7075	6887	1037
Average transmission delay (ps)	7149	7220	7070	1553
Worst case transmission delay (ps)	7351	7368	7249	3174

As for the forward delay, the delay for valid data to be computed (without considering the spacer delay), NCL presents an average of 395ps, while the best case average forward delay for the DIMS adders is of 2842ps, more than seven times bigger, and the worst case is 3092ps, roughly 7.83 times bigger. For the transmission delay, i.e. the delay required for computing valid data and spacer for each transmission, rates are roughly 4.55 and 4.65, respectively. In addition, for the NCL adder, the forward delay represents only 25% of the full transmission delay in average, while for the DIMS adders, in the best case, this relation is roughly 40%. Having small forward delay is very important for an asynchronous logic block because the faster valid data propagate through it, the faster it will be acknowledged. These results display the design improvement opportunities that are enabled by integrating NCL gates to the ASCEnD flow and library.

VI. CONCLUSIONS

This paper demonstrated that NCL gates can be designed through the ASCEnD flow, which automates all design steps but physical layout drawing. Such gates can be used to improve an asynchronous design in terms of power consumption and operating speed. ASCEnD-ST65 was enriched with a set of 14 NCL gates, which allow design improvements for semi-custom asynchronous designs. Future work includes prototyping an NCL-based design in silicon.

ACKNOWLEDGEMENTS

This work is partially supported by the CAPES-PROSUP (under grant 11/0455-5). Ney Calazans acknowledges CNPq support under grant 310864/2011-9. Authors acknowledge support granted by the INCT-SEC, process no. 573963/2008-8. Matheus Moreira acknowledges UNISC support.

REFERENCES

[1] P. Beerel, R. Ozdag M. Ferretti. “A Designer’s Guide to Asynchronous VLSI”. Cambridge University Press, 2010, 337 p.

[2] ITRS, “Design Section”, 2011, available at http://www.itrs.net.

[3] M. Moreira, B. Oliveira, J. Pontes, N. Calazans. “A 65nm Standard Cell Set and Flow Dedicated to Automated Asynchronous Circuits Design”. In: SOCC’11, 2011, pp. 99-104.

[4] M. Moreira, B. Oliveira, J. Pontes, F. Moraes, N. Calazans. “Adapting a C-Element Design Flow for Low Power”. In: ICECS’11, 2011, pp. 45-48.

[5] K. M. Fant, S. A. Brandt. “NULL convention logic: a complete and consistent logic for asynchronous digital circuit synthesis”. In ASAP’96, 1996, pp. 261-273.

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

[7] A. J. Martin. “The limitations to delay-insensitivity in asynchronous circuits”. In: AUSCRYPT’90, 1990, pp. 263-278.

[8] 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.

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