ASCEnD-FreePDK45: An Open Source Standard Cell Library for Asynchronous Design

Carlos H. M. Oliveira, Matheus T. Moreira, Ricardo A. Guazzelli, Ney L. V. Calazans
GAPH - FACIN - Pontifical Catholic University of Rio Grande do Sul - Porto Alegre, Brazil
{carlos.oliveira.004, matheus.moreira,ricardo.guazzelli}@acad.pucrs.br, ney.calazans@pucrs.br

***Abstract*— An analysis of the state of art in asynchronous circuits reveals a lack of resources to support their design. When asynchronous cell libraries appear in the literature, they often accompany a demand from specific circuit designs, and are not proposed as general purpose resources to support semi-custom design styles. Moreover, currently proposed asynchronous libraries employ commercial technologies that cannot be distributed as open access. This work proposes and describes ASCEnD-FreePDK45, an open source standard cell library to support the design of asynchronous circuits. This first release of the library contains 30 different cells and is based on the FreePDK45 design kit, a predictive 45nm technology. Currently, the ASCEnD-FreePDK45 library supports both NCL and SDDS-NCL asynchronous design templates and is fully compatible with the NanGate FreePDK45 open cell library. ASCEnD-FreePDK45 relies on the ASCEnD-A design flow for its construction, which provides tools for transistor sizing, layout generation and electrical characterization. This article illustrates the use of the library with a semi-custom implementation of a 32-bit Kogge-Stone adder with cells from both the ASCEnD-FreePDK45 and the NanGate FreePDK45 libraries.**

I. Introduction

In the last decades, semiconductor technologies have reduced the minimum feature size of transistors and wires, steadily increasing the maximum transistor count allowable in a single chip. These advances for a long time enabled creating digital circuits with higher performance and lower power features in successive technology generations. However, higher PVT sensibility, excessive power dissipation in clock trees, reaching physical limitations and other issues have imposed many design challenges in the most recent technology nodes. To overcome such challenges, asynchronous design can be adopted, as it employs communication through local handshaking, removing the need for a global clock signal. Among available asynchronous design styles, quasi-delay insensitive (QDI) templates stand out, due to their flexibility regarding wire and gate delays. QDI design provides better robustness against PVT variations, while leading to simpler timing closure and analysis. However, current available EDA tools and flows are not fully compatible with asynchronous design templates, requiring adaptations or the development of new tools. Moreover, semi-custom design of QDI circuits often requires special logic gates not available in commercial standard cell libraries. This lack of tools and support for asynchronous design implies higher design effort, making it unattractive to mainstream commercial applications.

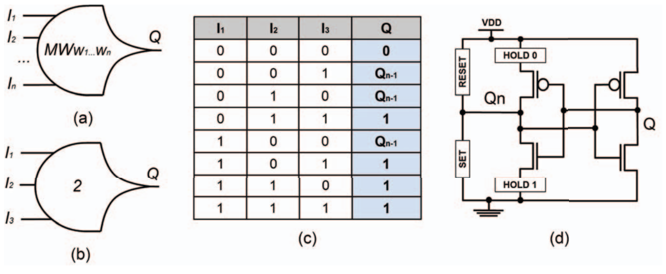
Recently, several approaches have been proposed to cope with the lack of support to asynchronous design, including some specific standard cell libraries. Among these, the ASCEnD library and the ASCEnD-A flow [1] provide support to QDI design, specially for NULL Convention Logic (NCL) and derived templates. Besides a cell library itself, ASCEnD-A is an automated flow for cell design, integrated with commercial EDA frameworks. It provides complex tools to support from cell specification through transistor sizing down to automated layout generation. ASCEnD currently addresses NDA-protected commercial technologies. This work presents ASCEnD-FreePDK45 an open access standard cell library. The library supports asynchronous design. The developed cell set supports NCL and SDDS-NCL designs in a predictive 45nm technology node. The new library is fully compatible with the NanGate-FreePDK45 library, based on the same process design kit (PDK).

II. The NCL and SDDS-NCL Design Templates

Modern asynchronous designs use one of either bundled-data or quasi delay-insensitive (QDI) template families. Despite the fact that bundled-data designs benefit from the use of conventional design tools, due to its similarity to synchronous circuits, these require extra care with the definition and verification of timing constraints between data and control signals. Accordingly, such circuits can be very sensitive to delay variations [2]. An alternative to avoid these issues is to encode control signals within data communication channels, which in part defines a QDI design template. Martin and Nyström report QDI as the most practical asynchronous template, due to its relaxed timing constraints and robustness to delay variations [2], which also makes the template suited for voltage scaling (VS) applications.

In the 1990s, Theseus Logic proposed the NCL logic family [3] to implement QDI asynchronous logic. Since then, NCL has been applied to deal with power problems, to design high speed circuits and fault tolerant applications, among other uses. A departure from the NCL logic family is NCL+, proposed in [4]. A fundamental difference between NCL and NCL+ gates is that the latter relies on a different handshake protocol called return-to-one (RTO) [5]. A recent work [6] showed that with a basic set of constraints for technology mapping it is possible to combine NCL and NCL+ gates in a single design advantageously. This combination constitutes a new design style called spatially distributed dual spacer NCL or SDDS-NCL, which can provide substantial improvements w.r.t. pure NCL or NCL+ alone [6]. The design of SDDS-NCL circuits requires the use of both NCL and NCL+ gates. MCL/MCL+ gates couple a threshold function with positive integer weights assigned to inputs to the use of a hysteresis mechanism. Many NCL/NCL+ gates are thus sequential circuits. Figure 1(a) shows a generic symbol for an NCL gate noted $MWw_1..w_n$-of-N, where M is the threshold function, N is the number of inputs and the w_is are input weights. Depending on the NCL gate function, each input can have a specific weight. Weight 1 is assumed wherever w_is are omitted. For example, Figure 1(b) shows the symbol of an NCL gate with $M = 2$, $N = 3$ and $w_1 = w_2 = w_3 = 1$. Given its specific function and naming style, we call this NCL2W111-of-3 or just NCL2-of-3.

The output of an NCL gate switches according to the following premises: (1) a high-to-low transition can only occur when all inputs are at logic 0; (2) a low-to-high transition occurs when the sum of weights for inputs at logic 1 is bigger than or equal to the threshold M. In case the input values and their weights do not combine to reach M, the output holds its previous state. For example, Figure 1(c) shows the NCL2-of-3 gate truth table. The output of this gate switches to 1 when 2 or more inputs are at 1, and it switches to 0 when all inputs are 0. For all other cases, the output keeps its previous state. Regarding transistor topologies, NCL gates can be realized with several distinct approaches, albeit this work only covers Sutherland et al. topologies [7]. Figure 1(d) illustrates a generic NCL gate with Sutherland topology, formed by four transistor blocks: SET, RESET, HOLD1 and HOLD0, an output inverter and a controlled feedback inverter. SET is responsible for producing logic 1 at the output, according to the functionality of the gate and RESET acts to force 0 on the output. HOLD1 and HOLD0 control the feedback loop, that can retain the output value when neither RESET nor SET blocks are active. In fact, the HOLD1 transistor arrangement is the complement of RESET, while HOLD0 is the complement of SET. NCL+ gates can be similarly defined. However, the assumption of the RTO protocol mandates the switching function of an NCL+ gate to be the reverse of its NCL counterpart: the output only switches to 1 when all inputs are at 1 and to 0 when threshold M is reached by the inputs at 0. For other input combinations, the output keeps its previous value. The symbol to represent NCL+ gates is the NCL symbol of Figure 1(a) with a "+" symbol added on its top right corner. To design NCL+ gates, the same topologies used for NCL are employed, as the only difference is how to implement blocks SET/HOLD0 and RESET/HOLD1. Classically, no negative function was supported in either NCL or NCL+. However, this limitation was overcome in SDDS-NCL [6]. In this way, to every classic NCL and NCL+ gate there is a corresponding negative unate version gate. This is useful for circuit synthesis optimizations, because internal inverters already present in these cells can be reused. Throughout this paper, negative unate gates will have the "I" prefix in their label to indicate their distinct functionality. For instance, the negative unate version of gate NCL2W111-of-3 is INCL2W111-of-3.



I_1	I_2	I_3	Q
0	0	0	0
0	0	1	Q_{n-1}
0	1	0	Q_{n-1}
0	1	1	1
1	0	0	Q_{n-1}
1	0	1	1
1	1	0	1
1	1	1	1

Fig. 1. NCL basics: (a) generic NCL gate symbol, (b) NCL2-of-3 gate, (c) truth table for the NCL2-of-3 gate and (d) generic NCL Sutherland topology.

As both NCL and NCL+ gates may need to include memorization circuits, their gate design follows constraints similar to those of latches and flip-flops from conventional libraries. Thus, worries about the isochronicity of forks inside NCL/NCL+ gates are not necessary during their design. Isochronic fork worries during circuit design using these gates are part of the supported QDI templates (NCL, NC+, SDDS-NCL, etc.), implying reference to the template design rules.

III. Library Development Infrastructure

ASCEnD-A is an environment adapted to semi-automatically generate cell libraries, with a special focus on cells to support asynchronous design. More details about the flow can be found in two PhD Thesis that targeted this flow as part of their development [1], [8]. Starting from a schematic specification, ASCEnD-A provides support for transistor sizing, layout generation, electrical characterization and library view generation. It employs tools from the Cadence commercial framework and complements it using various specific tools and scripts that integrate the new tools with the commercial framework. The new tools include the CeS/RoGen software for automatically dimensioning cell transistors, the LiChEn cell characterizer, which overcomes difficulties that appear when using the Cadence ELC characterizer for asynchronous cells like C-elements, and the ASTRAN automatic cell layout generator. Figure 2 shows an overview of the ASCEnD-A design flow with its four basics blocks: Cell Template Definition, Cell Sizing, Cell Layout Generation and Cell Characterization. Each of these blocks is broken up into several steps detailed elsewhere, see e.g. [1].

The Cell Template Definition block is the start of the flow. It defines the flow environment, technology parameters and the cell descriptions. Cells are defined by a transistor level schematic and follow a library template specification. This template allows the designer to specify certain cell parameters used in following blocks of the flow, which avoids interference among blocks.

The second block, Cell Sizing, dimensions cell transistors. This block takes as input cell templates from the first block and provides schematics with sized transistors. To determine the best transistor dimensions, ASCEnD-A incorporates two sizing tools: RoGen and CeS. RoGen generates a sizing simulation environment that provides performance/power information for each cell. The environment uses an exhaustive approach where all sizing combinations (within user-defined bounds) are simulated, following specific sizing parameters such as maximum transistor width, width step, etc. After simulating all sizing combinations, results can be evaluated by CeS coupled to a user-definable cost function, producing the best transistor sizing combination.

The Cell Layout Generation block is built around the ASTRAN tool [8], [9] to automatically produce the layout of each cell. ASTRAN is a netlist-to-layout tool which reads the output schematic from the Cell Sizing block and creates the

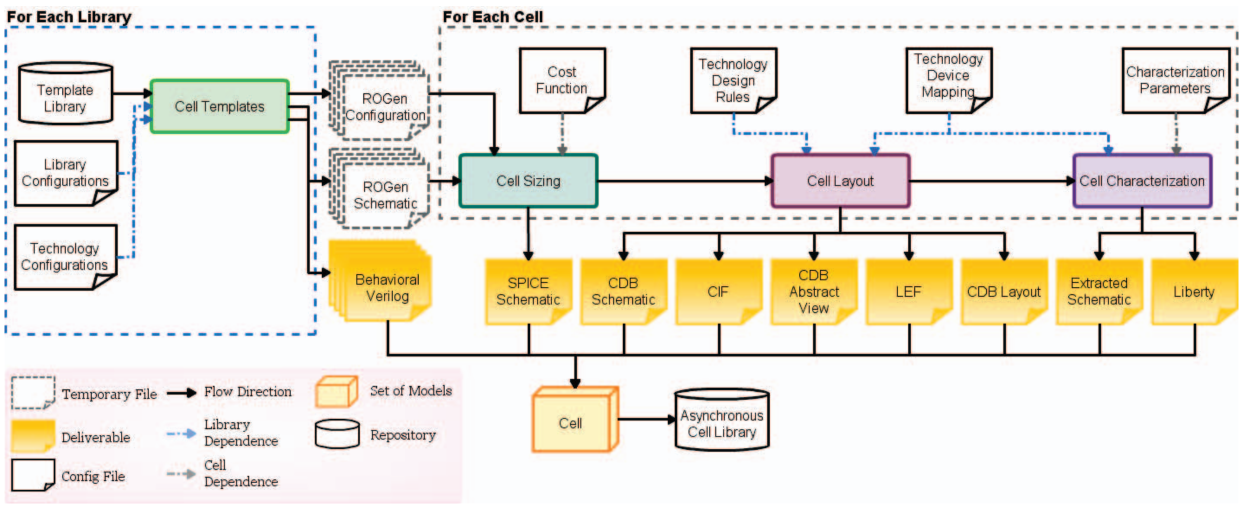


Fig. 2. An overview of the ASCEnD-A design flow. Cell Template, Cell Sizing, Cell Layout and Cell Characterization are the main parts of the flow.

layout. Next, the layout files from ASTRAN can be used for DRC/LVS checks and parasitic extraction, all automatically executed using commercial EDA tools.

Within the Cell Characterization block ASCEnD-A uses an in-house asynchronous library characterization tool called LiChEn [10]. This tool creates a simulation environment using the parasitic extraction files from the Layout Generation block. The simulation environment exercises all possible dynamic, static and internal arcs considering different input slopes, output loads and other library parameters. At the end, LiChEn exports all results to files in the open source Liberty format.

IV. The ASCEnD-FreePDK45 Library

ASCEnD-FreePDK45 is based on the open access FreePDK45 design kit, developed by Oklahoma State University (OSU) and North Carolina State University (NCSU) [11]. This design kit uses a predictive 45nm CMOS technology process and includes a technology library, technology files, display resources and has scripts to allow layout design and rule checking. Due to the fact that the design kit uses a predictive technology process, it is not possible to manufacture any circuit based on it, but it enables the free distribution of results without any restriction. This eases the process for educational institutions and VLSI researchers to obtain a design kit and share their work with others.

The ASCEnD-FreePDK45 library relies on the ASCEnD-A design flow, specifically proposed to support the construction of asynchronous cell libraries. The library implements 10 distinct functions from the NCL and NCL+ logic families, each one available in three driving strengths (X1, X2 and X4). Note that some cells have set and reset pins. Figures 3(b), 4(a) and 4(b) show the layout of three cells with different driving strengths and distinct functions. Currently, the library only provides 2-input cells. As previously mentioned, ASCEnD-FreePDK45 is fully compatible with the open access NanGate-FreePDK45 library (which counts more than 170 cells implementing 62 functions). This compatibility is important as conventional cells can be used to implement specific logic in QDI circuits. Layout design rules such as poly length and

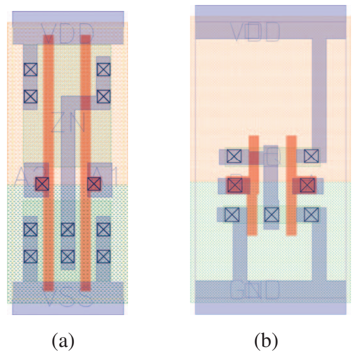


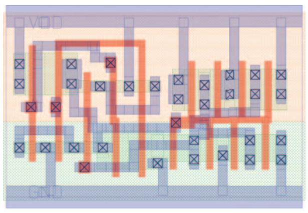
Fig. 3. Layout comparison between (a) Nangate-FreePDK cell NOR_X1 and (b) ASCEnD-FreePDK45 cell ST_INCL1W11OF2X1.

distance between two metal stripes come from the FreePDK45. ASCEnD-FreePDK45 cells follow the same architecture as the NanGate-FreePDK45 cells, as Figures 3(a) and 3(b) illustrate. Both have the same height (10λ), height of implant and doping layers.

The ASCEnD-FreePDK45 library is organized in two parts: front-end and back-end. The front-end comprises all timing and power consumption information and the logic function of each cell. Timing and power consumption information are described in the Liberty format, while logic function is provided in Verilog format in the ASCEnD-FreePDK45 datasheet. The back-end has a layout view in GDS, an abstract view in the LEF format, the SPICE cell description with and without parasitic extraction, and the schematic and layout in Open Access format (see details in https://corfu.pucrs.br/ascend-freepdk45/).

V. Case Study

A case study served to certify the usability of the ASCEnD-FreePDK45 library. This design consists in a 32-bit Kogge-Stone adder implemented in the SDDS-NCL template, with the same structure described in [6]. Two versions of the circuit were synthesized using the design method proposed and described in [12], one with strict timing constraints and


(a) ST_NCL2W11OF2X2

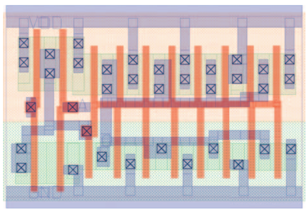

(b) ST_NCLP1W11OF2X4

Fig. 4. Layout samples for X2 and X4 ASCEND-FreePDK45 cells.

TABLE I
LIST OF ASCEND-FREEPDK45 FUNCTIONS.

Name	Family	Driving Strengths	Additional Logic
NCL2W11OF2	NCL	X1, X2 and X4	SET - RESET
INCL2W11OF2			
NCL1W11OF2			NONE
INCL1W11OF2			
NCLP2W11OF2	NCLP		
INCLP2W11OF2			

another with relaxed timing constraints. To do so, a target latency of 1.5 ns was defined for the former circuit while the latter underwent design under a 2.5 ns constraint. Note that the method in [12] allows automated synthesis and optimization of SDDS-NCL circuits using commercial STA tools and maximum delay constraints. This experimental setup enabled the validation of the compatibility of the models generated for the library cells and conventional tools and flows. More importantly, having these two different constraints scenarios enabled the assessment of the library capability to allow exploration of different design requirements.

Figure 5 shows the layout of the strictly constrained case study and Table II shows the results collected for each synthesis run. As Table II shows, the generated cells allow good exploration of the design space, enabling the designer to trade off latency, power and area figures. For example, the lower the latency, the larger the number of gates in the circuit and the larger its dissipated power. Moreover, as the same Table shows, the strictly constrained design had a reduced number of gates in the critical path, even if it increases the total number of gates. This demonstrates the compatibility of the generated models with commercial synthesis flows, as tools can perform latency optimizations by transforming logic paths at the cost of area and power overheads.

TABLE II
SIMULATION RESULTS FOR THE KOGGE STONE DESIGNS SYNTHESIZED CASE STUDIES. RESULTS ARE PRESENTED FOR THE FOLLOWING FIGURES: LATENCY CONSTRAINT (LAT.), GATES IN THE CRITICAL PATH (GCP), LEAKAGE POWER (LP), DYNAMIC POWER (DP) AND TOTAL POWER (TP).

Lat. (ns)	Gates	GCP	LP (uW)	DP (uW)	TP (uW)
1.5	2822	22	23.814	500.436	524.251
2.5	2604	34	22.126	482.808	504.934

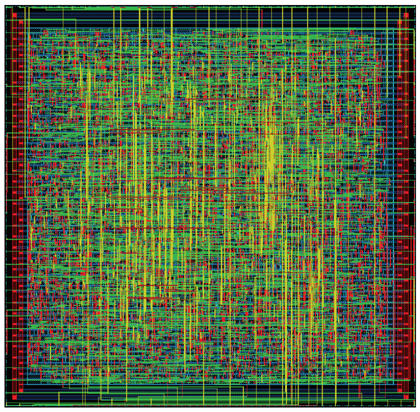


Fig. 5. Layout generated with the flow proposed in [12] for the strictly constrained version of the Kogge-Stone adder.

VI. CONCLUSIONS

This paper proposes the open access asynchronous cell library ASCEnD-FreePDK45, with 30 cells developed using the ASCEnD-A design flow. The library was validated through the synthesis of a 32-bit Kogge-Stone adder, showing its compatibility with commercial synthesis flows and with the Nangate FreePDK45 library. Currently, the library only comprises support for NCL, NCL+ and SDDS-NCL design. Its extension to support other QDI templates or bundled data design is ongoing work. The open access aspect of the library enables other research groups to easily contribute to the ASCEnD-FreePDK45 library with new cells. The ASCEnD-FreePDK45 library is available from http://www.inf.pucrs.br/g̃aph and comes with a public-domain license agreement which allows others to contribute and share results obtained with the library.

VII. ACKNOWLEDGEMENTS

This work was partially supported by the CNPq through grants no. 136603/2015-7 and 312556/2014-4.

REFERENCES

[1] M. T. Moreira, "Asynchronous Circuits: Innovations in Components, Cell Libraries and Design Templates," Ph.D. dissertation, Pontifícia Universidade Católica do Rio Grande do Sul, PUCRS, Brasil, 2016.
[2] A. Martin and M. Nystrom, "Asynchronous techniques for system-on-chip design," *Proc. of the IEEE*, vol. 94, no. 6, pp. 1089–1120, 2006.
[3] K. Fant and S. Brandt, "NULL Convention Logic™: a complete and consistent logic for asynchronous digital circuit synthesis," in *ASAP*, 1996, pp. 261–273.
[4] M. Moreira *et al.*, "NCL+: Return-to-One Null Convention Logic," in *MWSCAS*, 2013, pp. 836–839.
[5] ——, "Return-to-one protocol for reducing static power in C-elements of QDI circuits employing m-of-n codes," in *SBCCI, 2012*, 2012, pp. 1–6.
[6] ——, "Spatially distributed dual-spacer null convention logic design," *JoLPE*, vol. 10, no. 3, pp. 313–320, 2014.
[7] I. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, 1989.
[8] A. M. Ziesemer Jr., "Transistor Network Layout Automatic Synthesis," Ph.D. dissertation, Universidade Federal do Rio Grande do Sul, UFRGS, Brazil, 2014, in Portuguese.
[9] A. M. Ziesemer Jr. *et al.*, "Automatic layout synthesis with ASTRAN applied to asynchronous cells," in *LASCAS*, 2014, pp. 1–4.
[10] M. Moreira *et al.*, "LiChEn: Automated electrical characterization of asynchronous standard cell libraries," in *DSD*, 2013, pp. 933–940.
[11] J. Stine *et al.*, "FreePDK: An Open-Source Variation-Aware Design Kit," in *MSE*, June 2007, pp. 173–174.
[12] M. Moreira *et al.*, "Semi-custom NCL design with commercial eda frameworks: Is it possible?" in *ASYNC*, 2014, pp. 53–60.