LiChEn: Automated Electrical Characterization of Asynchronous Standard Cell Libraries

Matheus T. Moreira[1], Carlos H. M. Oliveira[1], Ney L. V. Calazans[1], Luciano C. Ost[2]

[1]Pontifical Catholic University of Rio Grande do Sul – Porto Alegre, Brazil
matheus.moreira@acad.pucrs.br, ney.calazans@pucrs.br
[2]Université de Montpellier II (UM2) – Montpellier, France
ost@lirmm.fr

Abstract — **Semi-custom design flows are a key factor for the rapid growth of integrated circuits and systems. They lower design complexity through the use of pre-designed and pre-characterized functional components called standard cells, instead of assuming that designers have to draw, place and connect each transistor. In this way, modeling of complex systems is easier. As CMOS technologies evolve into deep submicron nodes, asynchronous techniques gain relevance in the research community, due to their ability to cope with problems that are hard to solve with the synchronous paradigm. However, several specific components required in asynchronous designs are not available in commercial standard cell libraries, which constrains asynchronous design to use approaches close to full-custom ones. This limits modularity and increases design complexity. Thus, one of the possibilities for enabling further advance of the asynchronous paradigm is the availability of asynchronous standard cell libraries. Albeit industrial tools provide reasonable support to asynchronous standard cells physical design, the characterization of these cells using standard tools is usually quite laborious. This work proposes the Library Characterization Environment (LiChEn), an open source tool applicable to automatically characterize typical asynchronous standard cells. The tool managed to successfully characterize a standard cell library with over five hundred asynchronous components.**

Keywords-component; Asynchronous circuits, C-elements, electrical characterization, standard cell, design automation

I. INTRODUCTION

Current system-on-chip (SoC) designs use semi-custom design flows extensively as an answer to time to market pressures. Such flows speed up the design phase of medium to high performance integrated circuits (ICs) and are often referred as the key success factor for the rapid growth of integrated systems [1]. One of the reasons for this is the use of a standard library of pre-defined cells, provided by chip or intellectual property (IP) vendors. These libraries most often contain a collection of standard gates and flip-flops [2]. Cells are pre-designed and pre-verified, and have electrical behavior characterized by models described in text-based files, such as the Open Source Liberty Format [3]. When coupled to specific electronic design automation (EDA) tools, standard cell libraries facilitate the automation of IC implementations enormously.

The evolution of technology nodes allows building circuits with billions of transistors. In such a scenario, the design of fully synchronous systems is a daunting challenge. Guaranteeing clock integrity in highly complex chips by means of a single clock distribution network is a hard task, even with extensive automation provided by EDA tools. Furthermore, the power required by clock networks is increasing [4], which is not adequate for currently relevant market niches such as battery-based embedded systems. Also, as current technologies allow the implementation of multi-processor systems-on-a-chip (MPSoCs), a large amount of IP cores following particular standards and/or protocols may reside in a same chip. The requirements of particular IP cores often determine the use of specific operating frequencies, making the design of SoCs easier when using multiple clocks.

According to the International Technology Roadmap for Semiconductors [4], a shift on the VLSI design paradigm is inevitable in future technology nodes with regard to clocking strategies. Thus, the use of asynchronous techniques [5] [6] is gaining relevance in the VLSI research community. Such techniques can guarantee correct communication between distinct frequency domains or even completely eliminate the clock signal of a chip, or of some chip modules. Globally asynchronous locally synchronous (GALS) systems are those that employ different frequency domains with asynchronous interfaces to provide global communication [7]. On the other hand, fully asynchronous circuits are more challenging and consequently are less often used. However, there are some successful examples reported in recent literature, e.g. the Opus2 family of asynchronous DSP multiprocessors from Octasic Inc. [8] and high speed FPGAs from Achronix, Inc. [9].

Wider adoption of asynchronous techniques is constrained by the fact that such circuits still have limited design automation support [6]. There are a few tools for automating semi-custom asynchronous circuits design, but basic standard cells for implementing these circuits are not available off the shelf. In this way, asynchronous design is practically limited to full-custom approaches, where asynchronous logic components are specific to each design, and implemented through logic design approaches.

Building a generic standard cell library with components to support the asynchronous paradigm can be challenging. First, asynchronous circuits can be implemented using several different styles [6], each of which requiring a specific component set. Moreover, albeit typical industrial tools from vendors like Cadence, Mentor or Synopsys may be employed in the physical design of asynchronous components, the electrical characterization of such components using these tools is not straightforward. This is due to the fact that these tools aim primarily the synchronous paradigm. Characterizing sequential cells typically starts by treating control signals, especially the clock, and their assumptions. The drawback is that asynchronous sequential components are not controlled by a single global clock signal, but by signaling of specific, local events. Thus, using current commercial tools to characterize asynchronous cells entails laborious manual work.

This work proposes the Library Characterization Environment (LiChEn), deemed to allow an automatic and precise electrical characterization process for asynchronous standard cells. LiChEn was developed to support any CMOS technology and is built around SPICE simulations. It supports the characterization of propagation and transition delays, internal and switching power, measured for different input slopes and output loads and modeled using non-linear table models. Also, it enables capturing input and output pins capacitance and the leakage power for each of the cell internal states and for each power source. Results present the efficiency of LiChEn in terms of the total time required to characterize asynchronous standard cells. A cell subset that contains 18 different cells illustrates the characterization process. Additionally, the tool was recently used to characterize the electrical behavior of cells in the ASCEnD library [10] [11], composed currently by over five hundred asynchronous components.

The rest of this paper is organized in five sections. Section II discusses basic asynchronous components applicable at the gate level in asynchronous systems. Section III discusses challenges for the characterization of asynchronous standard cells, including a case study discussion. Section 0 presents the LiChEn tool, discusses its implementation, and shows the characterization environment. Section V presents some performance data for the characterization of standard cells using LiChEn. Finally, Section VI draws some conclusions and provides selected directions for further work.

II. ASYNCHRONOUS COMPONENTS

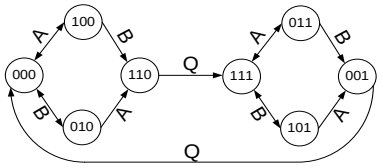
Asynchronous circuits can be implemented through a wide variety of schemes [5] [6] [12]. These schemes usually rely on a delay model, a data encoding, a handshake protocol and a set of basic components available at the standard cell level or as full custom cells.

A fundamental device that enables the design of several asynchronous templates is the C-element. The importance of C-elements is the fact that they help in the synchronization of independent events. Figure 1(a) depicts the truth table and Figure 1(b) shows a transition diagram for an ordinary 2-input C-element. Its output switches only when all inputs have the same logical value. When inputs A and B are equal, output Q assumes this same value. However, when inputs are different, the output keeps the previous logic value. The asynchronous state transition diagram of Figure 1(b) for the C-element has vertices containing values of inputs and output in the order ABQ_i. Albeit this component can be build using typical logic gates available in most standard cell libraries, this is inefficient.

The most popular transistor level topologies used for C-elements´ implementation appear in Figure 2: (a) Martin´s [13], (b) Sutherland´s [12], and (c) van Berkel´s [14]. Basically, the choice for a C-element implementation comprises a power-area-speed tradeoff, as discussed in detail in [15].

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

(a)



(b)

Figure 1 – Simple 2-input C-element specification: (a) truth table and (b) asynchronous state transition diagram.

Each C-element topology can be adapted to support more than two inputs and differentiated functionality. Such functionalities may include set and reset control inputs and inputs with asymmetric behavior. Set and reset inputs force the output of the component to logical 1 and logical 0, respectively. Asymmetric inputs, in turn, are inputs that interfere only in the low-to-high or in the high-to-low transition of the output, as discussed in [6].

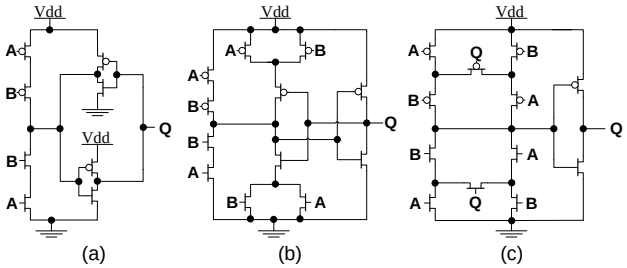


Figure 2 – The three basic C-element transistor topologies: (a) Martin´s [13], (b) Sutherland´s [12] and (c) van Berkel´s [14].

Specific asynchronous templates may require other components. Examples are the pre-charged half buffer or the pre-charged full buffer templates, discussed in detail in [6]. The use of these templates can be advantageous. However, their drawback is that there are currently little automation for the synthesis of asynchronous systems implemented with them.

The focus of this work is on C-elements, which enable asynchronous design that relies on standard cells using automated flows such as that proposed by Thonnart et al. [16]. Yet, these can also serve as a basis to build more sophisticated, template-based flows.

III. THE STANDARD CELL CHARACTERIZATION PROBLEM

The standard cell design method assumes that all cells are pre-designed and pre-validated circuit blocks. After interconnecting place and route these in a given design, it is possible to analyze power and delay figures. Enabling electrical analysis of the design requires a database containing detailed information on the timing and power consumption for each library cell. Different industrial tools are available to generate this standard cell timing and power database.

One example tool is called Encounter Library Characterization (ELC), commercialized by Cadence. It basically requires a SPICE or Spectre transistor level description of the standard cell, along with the technology models and a set of configuration parameters. These parameters are: process corner, operating voltage and temperature, and non linear model tables, which define input slew and output load vectors. With this information, the tool can recognize the logic behavior of the standard cell and generate the database. Next, it performs a set of simulations and outputs a Liberty file containing power and timing characteristics. In this way, the designer just produces a set of scripts and the tool automatically performs all required computation, with no manual intervention involved, except for the script generation. Considering that such libraries usually contain hundreds of components, this flow considerably simplifies the production of power and timing models.

However, its use in the characterization of asynchronous standard cells proved to be laborious, as the tool was not developed for this specific purpose. The problem is that when the logic behavior of the standard cell is not automatically identified by ELC, a designer needs to manually specify it inside the database. This means that the characterization process must be stopped, while the modification is done. This requires the generation of some text files by the tool, which are then modified by the designer, according to a specific syntax, defined within ELC. Once the correct logic behavior is specified, the designer must update the database and the tool can continue its characterization flow. Albeit some scripts could help automating the corrections required to use ELC, these are usually dependent on internal node labels that are only available after layout extraction and that vary from cell to cell. Thus, even using such an enhanced approach would lead to extra manual labor, where scripts need to be configured for each C-element that ELC fails to recognize.

To illustrate the above mentioned issues, Figure 3 shows what happens when trying to characterize the 2-input Sutherland C-element (see Figure 2(b)) with ELC. The tool identifies all transistors, but cannot compute the cell logic function. When it detects the loop that forms the memory mechanism of the C-element, it stops the characterization process and prompts the user to solve the problem.

TABLE I presents the topologies that have their logic recognized by ELC and can thus be automatically characterized by the tool, and the ones that the tool fail to recognize.

```
==============================
      DESIGN : CELEMENTX2
==============================
- loop node ( Q ) is found
- loop node ( 0 ) is found

=> no simulation

==============================
  stimulus generation  summary
==============================
Name             #MOS    #DVEC   #RVEC
-----------------------------------------
CELEMENTX2       12      0       0       *
-----------------------------------------
                         0       0
 Reading setup file : elc.st
 -   CELEMENTX2 (BLOCK) - nom_1.00V_25C - 2013
-06-28 00:17:37 (2013-06-28 03:17:37 GMT)

elc>
```

Figure 3 – Example of unrecognized C-element in ELC.

The Table presents results for 2-input C-elements with and without asymmetric inputs. Mention to **1d** and **1u** refer to the cells that have one input that only interferes in high-to-low (**1d**) and low-to-high (**1u**) transitions of the output. The Martin topology is recognized in all its variations: 2- and 3-input, asymmetric inputs and settable (**set**) and resettable (**rst**) C-elements. However the Sutherland and the van Berkel topologies were not automatically recognized, which required manual specification and verification of their function to characterize them.

TABLE I STANDARD CELLS THAT ARE RECOGNIZED BY ELC.

C-element	Martin	Sutherland	van Berkel
2-input	OK	FAILED	FAILED
2-input 1d	OK	FAILED	FAILED
2-input 1u	OK	FAILED	FAILED
2-input rst	OK	FAILED	FAILED
2-input set	OK	FAILED	FAILED
3-input	OK	FAILED	FAILED

A serious problem with his behavior is that the Sutherland and the van Berkel topologies are those that typically present the best power/speed/area tradeoffs [15]. Therefore, they are often preferred in asynchronous design. Also, considering that ASCEnD comprised over five hundred C-elements, from which 66% were based on Sutherland and van Berkel topologies, an alternative choice for ELC was required.

We suspect that ELC and similar characterization tools are designed to extract the logic behavior of transistor topologies with no feedback connections and from a few topologies with feedback connections in specific configurations, are found in typical synchronous components like latches and flip-flops. An example of such configuration is the two-inverter loop found in CMOS latches and RAM memory bits. This is exactly what the Martin topology C-element uses as output stage and is its only source of sequential behavior. However, the other topologies (Sutherland and van Berkel) use more complex feedback paths.

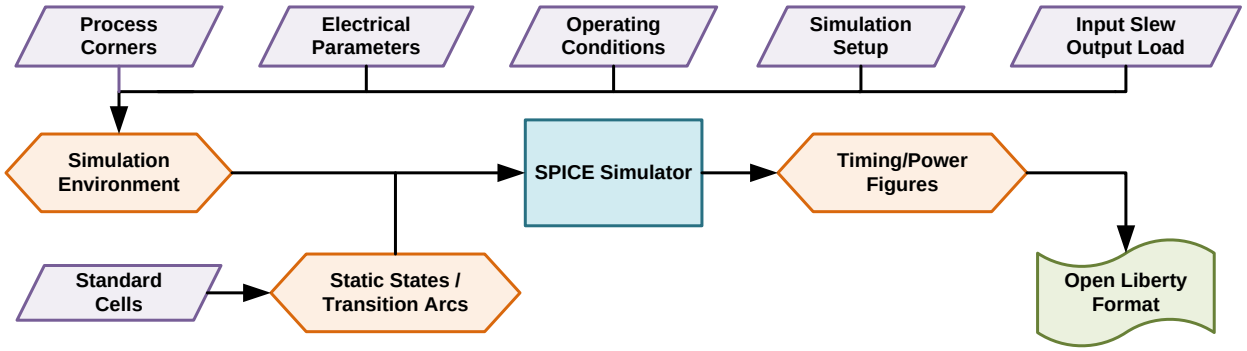


Figure 4 – The LiChEn electrical characterization flow.

LiChEn was designed to overcome the difficulties of characterizing the C-elements of the ASCEnD standard cell library [10], deemed to support a semi-custom approach and multiple asynchronous templates. During the construction of this library many issues were faced when dealing with electrical characterization by commercial tools, as these failed to recognize the C-elements logic functionality for topologies other than Martin´s. Circumventing the problems required extensive manual work to generate the cell power and timing database .

IV. The LiChEn Library Characterization Environment

LiChEn is implemented using the C/C++ language and is open source. It is based on the generation of a SPICE simulation environment where each standard cell has all its arcs and states exercised and its power and timing figures are characterized and exported to the Open Source Liberty format [3], compatible with most commercial EDA tools. All commands are given through a command line interface, and text-based scripts can help the automation of standard cell libraries characterization. The electrical characterization flow employed by LiChEn is represented in Figure 4. It comprises three main steps, represented by diamonds: (*i*) producing a cell simulation environment, (*ii*) finding static states and transition arcs, and (*iii*) generating timing and power figures through SPICE simulation.

A. Simulation Environment

In order to characterize asynchronous standard cells, the simulation environment must be configured. This requires to employ technology-specific data, providing corner selection, operating conditions, electrical specifications, and simulation parameters.

First, technology models are furnished to the tool, together with the specific corner to use during simulation. Next, electrical parameters for collecting information must be given such as minimum logical 1 and maximum logical 0 voltages, low-to-high and high-to-low transition thresholds. The precision of the results generated by LiChEn for the provided technology depends on the quality of choice of these parameters. The tool also requires information about operating conditions, nominal voltage and temperature, and global power nets like **vdd** and **gnd**, which feed the standard cells. Afterwards, parameters to control the simulation must be furnished, including maximum simulation time, instant when to start measuring the information and simulation minimum step.

Pin-to-pin propagation, transition delays and dynamic power consumption depend on the input slew rate and on the output capacitance load. Hence, these delay and power figures are modeled in non-linear tables, where each value depends on a combination of an input slew rate and an output load capacitance. However, to do so, input slew rate and output load capacitance vectors are required, to generate the characterization simulation environment. These vectors can then be independently used to generate the models for each standard cell. Moreover, since characterization relies on non-linear table models, the quality of the results is a function that strongly depends on the precision of the provided input slew rate and output load capacitance vectors.

B. Static States / Transition Arcs Search

Once the simulation environment configuration is satisfactory, the tool can deal with specific standard cells. These are available as SPICE netlists containing the transistors schematic along with all parasitics. Also, the standard cell logic function and output and input pins names must be defined, to guide the tool in its search for transition arcs and static states, which will be used to conduct the simulations required for the standard cell characterization. In LiChEn, logic functions can employ the following logic operations: conjunction (*), disjunction (+) and complement (~). Moreover, parenthesis can be used to express hierarchy.

For instance, the basic C-element with inputs A and B and output Q has the following logic function:

$$Q = (A*B) + (A*Q) + (B*Q) \quad (1)$$

In this function, it is clear the feedback loop formed by output Q. As explained before, characterizing a standard cell that executes such a logic function with conventional tools requires laborious manual work. Also, as a standard cell library typically contains hundreds of cells, it is advantageous to have a highly automated characterization process.

LiChEn employs a branch and bound algorithm to find all transition arcs and static states (SSs), even in functions that employ a feedback loop. Basically, the tool initially sets all input and output pins to logical 0 and computes the resulting value of the output according to the logic function, obtaining the first static state.

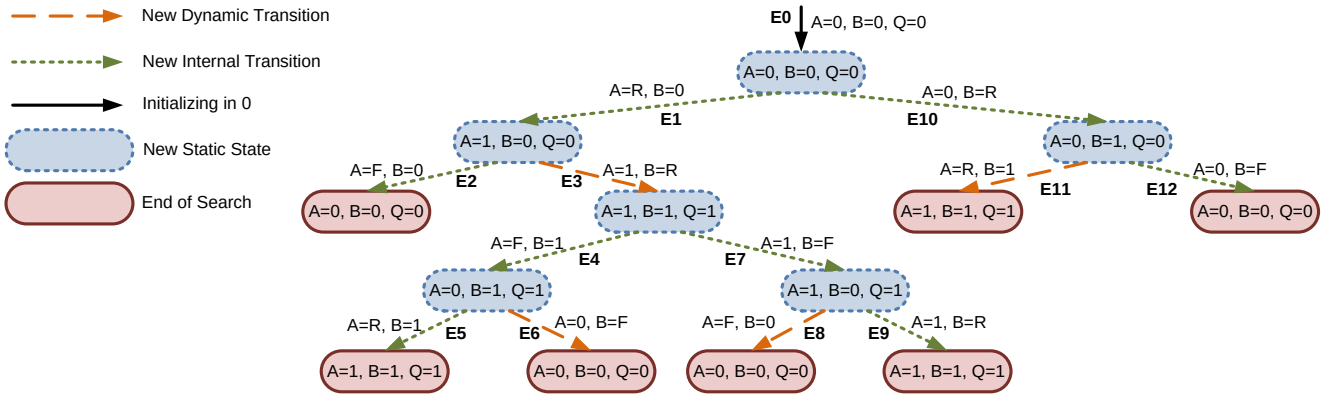


Figure 5 – Computation of transition arcs and static states of a two input C-element. "R" stands for low-to-high transitions (rise) and "F" for high-to-low transitions (fall). Graph edges are labeled numerically by E_N according to the order that they occur during the transition arcs search.

Next, it switches the logical value of one input at a time and evaluates the next output value through a recursive function. Each new input/output value after a transition is accounted as a new SS. To avoid exponential computation time, as soon as LiChEn detects that a previously computed SS was already computed previously, it stops the recursion. During switching of the inputs logical values, LiChEn evaluates in which cases the output value changes when an input switches. These cases are called Dynamic Transition Arcs (DTAs). The cases where input switching does not cause an output switching are called Internal Transition Arcs (ITAs).

Figure 5 presents the computation of transition arcs and SSs for the logic function of the 2-input C-element described by Equation (1). In the Figure, graph edges are numerically labeled according to the order in which they are computed. Initially, inputs A and B and the output Q are set to logical 0 (E0). The resulting value is Q = 0. This characterizes an SS (A=0, B=0, Q=0). Next, through a recursive function, input A switches to logical 1 (E1). The result is still Q = 0. However, a new SS (A=1, B=0, Q=0) and an ITA (A=R, B=0, Q=0) are found. Then, the value of A is switched once more (E2), resulting in the static state (A=0, B=0, Q=0). Because this state was already computed previously, this branch of the search graph is pruned. The function returns one node (A=1, B=0, Q=0), and switches the other input B logical value (E3). However, this time, this input switching event caused the output to switch, generating a new SS (A=1, B=1, Q=1) and a new DTA (A=1, B=R, Q=R). From this node, the next step is to switch the logical value of input A (E4), generating a new branch in the search graph as E5 and E6 take place. Once this branch is pruned, input B has its logical value switched again (E7). This search continues until all branches are killed after the last event (E12).

TABLE II shows the complete set of SSs, DTAs and ITAs, in the order they were computed, for the logic function of the 2-input C-element. This simple algorithm is capable of detecting all SSs, DTAs and ITAs, even those that depend on feedback loops. In this way, it is possible to efficiently characterize the electrical behavior of asynchronous standard cells. Yet, currently, the tool supports only single output standard cell, and cannot characterize typical synchronous constraints for sequential cells, such as setup and hold times. Thus, other than asynchronous cells, LiChEn can be used to characterize typical combinational logic cells, such as ANDs and ORs. However, it is not recommended its use to characterize synchronous sequential standard cells like flip-flops or latches.

TABLE II RESULTING SSS, DTAS AND ITAS AFTER THE COMPLETE SEARCH FOR THE LOGIC FUNCTION OF A 2-INPUT C-ELEMENT. LOGICAL VALUES ASSUME THE ORDER A, B, Q FOR INPUTS/OUTPUTS.

SSs	DTAs	ITAs
0,0,0	1,R,R	R,0,0
1,0,0	0,F,F	F,0,0
1,1,1	F,0,F	F,1,1
0,1,1	R,1,R	R,1,1
1,0,1		1,F,1
0,1,0		1,R,1
		0,R,0
		0,F,0

C. Timing/Power Figures Generation

LiChEn may start the characterization process once all SSs, DTAs and ITAs have been computed. Based on these results, the tool generates SPICE files that implement each transition arc and each SS. Measurements are conducted during SPICE simulation, based on the configuration given in the first stage of the characterization flow, discussed in Section 0.A. During characterization, LiChEn measures input gate capacitance for low-to-high and high-to-low transitions, static and dynamic power and timing figures.

Timing figures are measured as pin-to-pin propagation delays and output transition delays. The static power is measured for each SS. Dynamic power, in turn, is divided in two parts: switching and internal power, measured as the power consumed in DTAs and ITAs, respectively.

The input gate capacitance is computed as the average current for low-to-high and high-to-low input transitions and exported as rising and falling input capacitances, respectively. Static power, on the other hand, is measured for each SS according to the average current drawn from the power source. In addition, the static power consumption of each SS is measured for each power source independently.

LiChEn models dynamic power consumption using non-linear tables to achieve comprehensive coverage of the electrical behavior of standard cells combinations. In this way, a vector of input slew rates and output capacitance loads is required by the simulation environment, as described in Section 0.A. The tool measures switching and internal power consumption [2] for each combination of input slew and output load, generating a two dimensional non-linear table. The switching power is measured as the power consumed in DTAs, and the internal power as the power consumed in ITAs, without considering static power consumption. The internal power consumption is calculated as "P_I", according to Equations (2) and (3) [1]:

$$P_I = V_{(vdd)} * \int_0^T I_{(t)} dt - Static_{tot}, \quad (2)$$

$$Static_{tot} = V_{(vdd)} * \left(I_{before} * t_{init} + I_{after} * (t - t_{init}) \right) \quad (3)$$

Here, $\boldsymbol{Static_{tot}}$ stands for the total measured static power consumption. Computation starts by calculating the total charge drawn from the power source while in the SS that occurred before the transition arc (Ibefore*tinit). This charge is added to the charge drawn from the power source for the SS that results from the transition arc in the remaining simulation time. This sum is then multiplied by the supply voltage, generating $\boldsymbol{Static_{tot}}$. Simulation time definitions are given prior to characterization, as explained in Section 0.A. In this way, internal power consumption considers only the charge drawn from the power source that was caused by an input switch. Switching power consumption is calculated in a similar way. However, the current required to charge the output capacitance is also subtracted. In this way, the switching power consumption is measured as "Ps", according to Equation (4):

$$P_S = V_{(vdd)} * \int_0^T I_{(t)} dt - Static_{tot} - \left(\frac{1}{2} * C_{out} * V_{(vdd)}^{\ 2} \right) \quad (4)$$

As for timing characteristics, LiChEn models propagation and transition delays [2]. Propagation delay is measured for DTAs as the total time that it takes for a switching at the input to cause a switching at the output. This depends on high-to-low and low-to-high transition thresholds, as explained in Section 0.A.

Figure 6 displays an instance, the propagation delays D_{Plth} and D_{Phtl} for a 2-input C-element. These are based on DTAs (A=1, B=R, Q=R) and (A=F, B=0, Q=F), respectively. First, both inputs and the output are at logical 0. Next, input A switches to logical 1, without modifying the value of the output. However as soon as input B reaches the low-to-high transition threshold, when switching its logical value, the time it takes for Q to reach the high-to-low transition threshold starts to be measured. This is modeled as a low-to-high (rising) propagation delay from pin B to Q. Similarly, a high-to-low transition in input B followed by a high-to-low transition in input A is modeled as high-to-low (falling) propagation delay from pin A to Q. In this manner, the tool models all possible propagation delays of a given standard cell from all inputs to the output. Transition delays are modeled as the time it takes for an output to switch from a logical value to another. These values are based on minimum logical 1 and maximum logical 0 voltages, as discussed in Section 0.A.

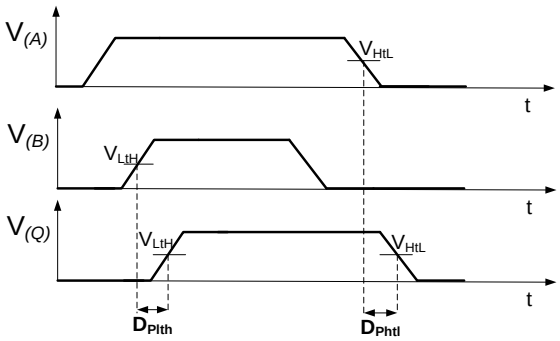


Figure 6 – Example of propagation delays for a 2-input C-element.

Figure 7 shows an example of the transition delays for an output pin Q. The low-to-high (rising) transition delay (D_{Tlth}) is measured as the time an output takes to achieve the minimum logical 1 voltage after it reached the maximum logical 0 voltage. Similarly, the high-to-low (falling) transition delay (D_{Thtl}) is the opposite.

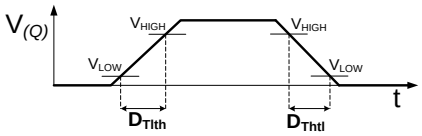


Figure 7 – Transition delays example for an output pin Q.

Currently, the tool does not characterize setup time and minimum pulse width. This is due to the fact that this is not necessary for most of asynchronous templates, because circuit monotony is guaranteed. Also, all simulations are currently performed through Cadence Spectre SPICE simulator. However, LiChEn is easily adaptable to support other simulators. After modeling all the electrical figures of the library standard cells, LiChEn exports the generated results to a text based file in the Open Source Liberty Format [3], which is compatible with most EDA tools.

V. LiChEn Performance

Different experiments were performed during LiChEn development, in order to assure its precision when characterizing standard cells. Some of these are presented herein. All experiments employ the STMicroelectronics 65nm CMOS technology.

The first experiment uses the manufacturer´s standard cell library. LiChEn characterized the extracted physical views of the circuits for seven logic gates: a buffer, 2- and 3-input AND and OR gates, and 2-by-2 ANDOR and ORAND gates. The obtained results were compared to those

[1] Note that strictly speaking, the values called here *power* correspond indeed to energy values, but this is the approach we borrowed from the EDA vendors' terminology and the Liberty, with which we have to keep compatibility.

provided by the chip manufacturer models, also represented in the Open Liberty Format. Results are depicted in TABLE III.

A design containing a single cell was implemented for each of the seven cells. These designs fed the validation environment, where a 10 ps slew rate, equivalent to the transition delay of an average sized inverter, was applied in all inputs and a 5fF load connected to the output, equivalent to the input capacitance of four equivalent inverters. Timing figures for each design were annotated for falling and rising delay of each input-to-output pins, which account for the sum of propagation and transition delays. A switching activity of one million transitions in all pins was provided for power analysis. Through this scenario, results were generated when using the original characterization models, supplied by the standard cell library provider (called Corelib), and the ones generated by LiChEn. All results are based in a typical process operating at 25°C and 1V supply.

TABLE III COMPARISON OF ANOTATED STATISTICAL TIMING, POWER AND INPUT CAPACITANCE VALUES WHEN USING THE CORELIB ORIGINAL LIBERTY FILE AND THE ONE GENERATED THROUGH LICHEN WITH CADENCE RTL COMPILER.

Standard Cell		Delay (ps)		Power (nW)		Input
		Rise	Fall	Dynamic	Static	Cap. (pF)
BUFF	*Corelib*	18.00	21.00	48,978	37.5	1.253
	LiChEn	17.00	21.00	48,542	36.4	1.229
AND2	*Corelib*	22.50	20.00	68,030	49.6	1.576
	LiChEn	21.50	19.00	71,267	48.0	1.530
AND3	*Corelib*	30.60	23.30	84,033	45.8	1.544
	LiChEn	29.60	22.30	85,896	44.1	1.504
OR2	*Corelib*	18.00	29.50	65,142	43.7	1.459
	LiChEn	17.50	29.00	66,993	42.1	1.421
OR3	*Corelib*	25.30	45.60	84,614	37.3	1.427
	LiChEn	24.30	44.60	87,825	34.7	1.409
AO22	*Corelib*	26.50	38.25	97,850	55.8	1.530
	LiChEn	25.25	36.50	102,867	54.6	1.499
OA22	*Corelib*	27.50	38.00	98,585	56.8	1.530
	LiChEn	26.25	37.75	102,382	55.8	1.493

The delay for rise and fall transitions is measured as the average delay for each input pin. Power consumption was obtained according to the power analysis scenario previously described (in Section 0.C) and the input capacitance was annotated directly from the Open Liberty Format files. Results show that LiChEn provides a rather precise characterization. In the worst case, the difference between the results obtained using the original characterization models and the ones generated by LiChEn was within 5%. By the authors' own experience this is not very significant, as similar variations may occur even when using industrial tools from two different EDA vendors and the same timing and power model files.

As a second experiment, LiChEn was employed in the characterization of all C-elements of the ASCEnD library [10], a library for supporting generic standard cell based asynchronous designs. The library was implemented in the same 65nm CMOS technology, using general purpose standard threshold transistors. ASCEnD comprises over five hundred different C-elements [11], with varying driving strengths, number of inputs, control signals and functionalities. The generated characterization models are fully compatible with typical models provided by standard cell library vendors.

The time for characterizing 18 different types of C-elements of the ASCEnD library using LiChEn was measured to evaluate the program performance. Results were obtained using an Intel Xeon W3540 2.93GHz workstation, using one core only, due to restrictions of the tool itself. TABLE IV shows the total wall clock time to characterize 6 types of C-elements for each of the three topologies referred in Section II. The characterized C-elements included: 2- and 3-input cells, 2-input resettable (**rst**) and settable (**set**) and 2-input asymmetric cells, with one input that only interferes in high-to-low transitions of the output (**1d**) and one input that only interferes in low-to-high transitions of the output (**1u**). All results account for the total time required by all characterization steps, including environment configuration, search for SSs, DTAs and ITAs, generation of the simulation environment, electrical simulation and generation of the Open Liberty Format file.

TABLE IV TIME FOR CHARACTERIZING C-ELEMENTS WITH LICHEN.

C-element	Martin	Sutherland	van Berkel
2-input	120 s	130 s	104 s
2input 1d	88 s	79 s	87 s
2-input 1u	95 s	90 s	103 s
2-input rst	244 s	256 s	260 s
2-input set	285 s	244 s	278 s
3-input	419 s	397 s	415 s

As TABLE IV shows, the fastest characterization was for the 2-input **1d** Sutherland C-element, less than 80 seconds. More complex schematics such as the 3-input implementations required up to 419 seconds to be characterized. From the measured times, over 95% of these were typically required by electrical simulation, which demonstrates the efficiency of the LiChEn algorithms. Also, licensing elapsed time for the Spectre simulator employed for these simulations accounted for almost 35% of total time, in the worst case. TABLE V presents the total time required to characterize the C-elements showed in TABLE IV without taking into consideration licensing elapsed time. As the Table shows, the efficiency of LiChEn could be substantially improved if multiple simulator calls were avoided or parallel simulations were performed. This is under development for the next version of the tool.

TABLE V TOTAL TIME FOR CHARACTERIZING C-ELEMENTS WITH LICHEN WITHOUT CONSIDERING SPECTRE LICENSING ELAPSING TIME.

C-element	Martin	Sutherland	van Berkel
2-input	84 s	89 s	98 s
2-input 1d	57 s	68 s	66 s
2-input 1u	64 s	69 s	72 s
2-input rst	201 s	209 s	198 s
2-input set	218 s	216 s	210 s
3-input	322 s	330 s	348 s

Considering the worst case performance presented in this work, 419 seconds to characterize a single C-element, a library of 100 standard-cells would take roughly 11,5 hours

to be characterized. The authors consider this a good result, considering that LiChEn can be configured using a text-based script file, which helps automating the process of characterizing large libraries of standard cells. Also, having a fully automated flow for characterizing asynchronous standard cells, avoids the possibility of errors that can occur when employing tools designed for synchronous systems, given the manual labor required by them.

VI. Conclusions and Future Work

This work proposed an open source EDA tool for automatically characterizing asynchronous standard cell libraries. Results show that the tool is quite precise and can also be employed to characterize typical single-output combinational logic cells. The tool is capable of characterizing input pins capacitance, static, internal and switching power and propagation and transition delays. Also, it was efficiently employed in the electrical characterization of a standard cell library composed by over five hundred standard cells. Performance figures demonstrate that the tool can efficiently generate timing and power models for big libraries in some hours. This is accounted in terms of the total wall clock time required by LiChEn to characterize typical asynchronous standard cells. As the interest in asynchronous circuits continuously grows in the research community, LiChEn can be very useful to enable semi-custom approaches to this design paradigm.

Currently, LiChEn is based on Cadence Spectre SPICE simulator. Future work includes allowing the use of other vendors' simulators. Also, LiChEn does not yet support distributed computing when simulating all the generated scenarios. However there is work in progress in coupling the tool with the Sun Grid Engine, which is also open source and can be used to accelerate the characterization process. Another future work is enabling the tool to characterize multi-output standard cells, which is very useful for specific classes of asynchronous circuits. Finally, another important thing about LiChEn is that it searches all SSs and transition arcs of a logic function, generating SPICE simulation scenarios, which can be very useful for academic purposes, such as in the context of microelectronics and VLSI design courses.

Acknowledgments

This work is partially supported by the CAPES-PROSUP and FAPERGS under grant 11/1445-0, and by the CNPq under grants 310864/2011-9, 556779/2009-6. Authors also acknowledge the support granted by the CNPq to the INCT-SEC (National Institute of Science and Technology – Critical Embedded Systems – Brazil), process no. 573963/2008-8.

References

[1] H. Eriksson, P. Larsson-Edefors, T. Henriksson, C. Svensson. "Full-Custom vs. Standard-Cell Design Flow – an Adder Case Study". In: 8th Asia and South Pacific Design Automation Conference (ASP-DAC), 2003, pp. 507-510.

[2] J. M. Rabaey, A. Chandrakasan, B. Nikolic. "Digital Integrated Circuits a Design Perspective". Upper Saddle River: Pearson Education, 2003, 761p.

[3] Liberty. "Open Source Liberty". Available at http://www.opensourceliberty.org, 2012.

[4] International Technology Roadmap for Semiconductors. "Design", 2011 Edition. Available at http://www.itrs.net, 2012.

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

[6] P. Beerel, R. Ozdag and M. Ferretti. "A Designer's Guide to Asynchronous VLSI". Cambridge University Press, 2010, 337 p.

[7] D. Chapiro. "Globally Asynchronous Locally Synchronous Systems". PhD Thesis, Stanford University, 1984, 134p.

[8] M. Laurence. "Introduction to Octasic Asynchronous Processor Technology". In: International Symposium on Asynchronous Circuits and Systems (ASYNC'12), 2012, pp. 113-117.

[9] S. Ramaswamy, L. Rockett, D. Patel, S. Danziger, R. Manohar, C. W. Kelly, IV, J. L. Holt, V. Ekanayake, D. Elftmann. "A Radiation Hardened Reconfigurable FPGA". In: 2009 IEEE Aerospace Conference, 2009, 10p.

[10] M. T. Moreira et al.. "A 65nm Standard Cell Set and Flow Dedicated to Automated Asynchronous Circuits Design". In: International SoC Conference (SOCC'11), 2011, 6p.

[11] M. T. Moreira, B. S. Oliveira, J. J. H. Pontes, F. G. Moraes, N. L. V. Calazans, "Adapting a C-element Design Flow for Low Power," In: International Conference on Electronics, Circuits and Systems (ICECS'11), 2011, pp. 45-48.

[12] I. Sutherland. "Micropipelines". Communications of the ACM, 32, 1992, pp. 720-738.

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

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

[15] M. T. Moreira, B. S. Oliveira, F. G. Moraes, and N. L. V. Calazans. "Impact of C-elements in asynchronous circuits". In: International Symposium on Quality Electronics Design (ISQED'12), 2012, pp. 438-444.

[16] Y. Thonnart, E. Beigne and P. Vivet, "A Pseudo-Synchronous Implementation Flow for WCHB QDI Asynchronous Circuits," in: International Symposium on Asynchronous Circuits and Systems (ASYNC'12), 2012, pp. 73-80.