Robust and Energy-Efficient Hardware: The Case for Asynchronous Design

Ney L. V. Calazans, Taciano A. Rodolfo, Marcos L. L. Sartori
Pontifícia Universidade Católica do Rio Grande do Sul - Porto Alegre, Brazil
ney.calazans@pucrs.br, taciano.rodolfo@acad.pucrs.br, marcos.sartori@acad.pucrs.br

***Abstract*—Current technologies behind the design of semiconductor integrated circuits allow embedding components in the order of billions in a singe die. This enables the construction of very complex circuits in a tiny space, dissipating little energy and producing huge amounts of useful computational work. However, the present levels of integration for electronic components in silicon and similar materials are not easily managed as parameter variations grow steadily, making the design tasks increasingly challenging. Synchronous techniques have dominated the digital system design landscape for many decades, but their costs are hard to cope with. Asynchronous design and particularly quasi-delay insensitive design promises to deal with the same challenges more gracefully in current advanced nodes, and possibly irrevocably in future technology nodes. This article proposes a review of the state of the art in using asynchronous circuit design techniques to achieve energy-efficient and robust digital circuit and system design. In particular, the definition of a robust digital circuit comprises addressing several aspects to which a digital system design is expected to be robust to, including: (1) voltage variations; (2) process variations; (3) temperature variations; (4) circuit aging. Besides addressing energy-efficiency and all the mentioned robustness aspects, this work also approaches some of the state-of-the-art tools available to deal with asynchronous design, and points to desirable research development to be conducted in these subjects in the future.**

I. Introduction: Robust and Energy-Efficient Hardware

A relentless evolution characterizes semiconductor technologies in the last several decades. From a $10\mu m$ feature size in the beginning of the 1970's, down to sub-micron transistors in the middle 1980's, followed the deep sub-micron era in the early years of the 21st century. Feature sizes of less than $100nm$ could then be fabricated. The last 20 years have seen the rise of $10nm$, $7nm$ and $5nm$ manufacturable technologies, and a few industries are already working toward $3nm$ and $2nm$ nodes. Some basic research breakthroughs already indicate $1nm$ nodes are possible [1]. While the technology developments based in the latest nodes are astonishing, the problems to employ these are definitely challenging at many levels. Designing latest node chips is very hard and very costly; electronic design automation (EDA) lags behind in multiple aspects, and manufacturing is limited to one or to very few places on Earth. Also, in these advanced nodes variability of designs, good dies yield, manufacturing faults and the longevity of circuits are hard to control and predict. Meanwhile, the range of commercial technologies only widens, since multiple much older technology nodes still occupy well defined market niches and are yet economically successful. For example, it is not uncommon for commercial integrated circuits (ICs) to be designed today using a $180nm$ or even a $250nm$ technology, with feature sizes two orders of magnitude or more larger than that in state-of-the-art nodes. In-between there are evidently many technology choices.

Given the wide range of choices, the IC design problems and features also vary widely in importance. For example, the transition from $45nm$ and $32nm$ to smaller feature sizes mandated a change in the way transistors are designed, forcing the abandonment of CMOS bulk in favor of the more advanced FDSOI and/or FinFET technologies. As another example of design choice, there is the need or intent to use transistors with multiple threshold voltages, to better control device characteristics and trade speed and power. Although this has been applied to some early technology nodes such as $500nm$ [2], multi-V_{th} transistors are only easily available in commercial technologies with $120nm$ or $130nm$ feature sizes or smaller [3].

Within the context of the technological scenario just discussed, the adoption of the synchronous digital circuit design paradigm by IC designers is one of the reasons behind the rapid development of the VLSI industry. This is mainly due to the capability of this paradigm to reduce design complexity by using a global control signal called *clock*, that dictates all sequencing of events. A synchronous designer can ignore wire and gate delays, as long as the logic path between each pair of storage elements always takes less time than the clock period. Unfortunately, with the exponential growth of integration capabilities, distributing a clock signal across a complex IC is challenging. Albeit there are different techniques and EDA support to automatically generate clock distributions, the required circuitry may take something from 30% to 50% of the total power in synchronous circuits [4]. This is further complicated by the inevitable delay uncertainties caused by data dependency and process, voltage and temperature (PVT) variations. To cope with these, synchronous designs rely on the addition of delay margins to the clock signal, which translates to performance losses, and can require tuning the operating voltage, further adding power and area overheads [5], [6]. Asynchronous design, on the other hand, does not rely on global timing assumptions and treats time as a continuous variable, such that synchronization and sequencing of events take place locally, between communicating entities [7].

Some of the most relevant design issues faced in any available technology node are design robustness and energy efficiency. While energy efficiency is relatively easy to define and quantify, design robustness requires more careful consideration. First, a dictionary *robustness* definition states it as "the quality or condition of being strong and in good condition" or "the ability to withstand or overcome adverse conditions or rigorous testing." The second definition is more adequate to the goal of this article, since a robust circuit is expected to stand *adverse conditions* and yet operate according to its original specification.

It is interesting to start by investigating formal notions of robustness. Doyen et al. provide a precise definition for the robustness of sequential circuits concept in [8]. In their approach,

Digital Object Identifier 10.29292/jics.v16i2.518

sequential circuits are synonyms of Mealy machines, a formal model with sufficient capacity to express any digital system. The authors start by exploring the definition of robustness of continuous (non-discrete) systems, denoted as a form of *uniform continuity*: "A system is uniformly continuous if for every positive real ϵ, there exists a positive real δ, such that any change smaller than δ in the input results in a change smaller than ϵ in the output." [8]. Simply said, this definition implies that a (continuous) system is robust if bounded variations at the system inputs correspond to bounded output variations. The authors also mention that in control theory the system inputs are often divided into *control* and *disturbance* variables, and robustness is studied considering only the latter. Based on these initial affirmatives, the work develops a notion of robustness for discrete systems in the form of finite state transducers. Mealy machines are modeled as transducers that take letters of an input symbol alphabet and generate letters of an output symbol alphabet. The discrete robustness definition relies on the formal definition of a metric, called *common suffix distance*, identified as the last position in which two (possibly infinite) sequences of letters from the output alphabet differ. Authors then define the notion of *finite disturbance horizon*, and provide a theorem that states that a sequential circuit is robust under its disturbance alphabet if and only if it has a finite disturbance horizon. In more informal terms this means that a robust sequential system always *forgets* about its inputs older than some bounded amount of previous steps.

While instrumental for several theoretical analyses, the formal robustness definition for discrete systems is purely behavioral, and can be hard to employ directly in digital circuit design. In practice, robustness must be considered towards adversity in one of more environmental or operational conditions. It is thus useful to break the concept in terms of robustness to measurable quantities, which amounts to classify it into several categories. The proposal is to explore robustness to: (1) voltage variations; (2) process variations; (3) temperature variations; and (4) aging. For most of these categories and across a wide range of target technologies achieving robust designs for a digital system specification is a demanding task. Conventional synchronous design can lead to sub-optimal and less robust design under several adverse situations.

This survey proposes a review of works that demonstrate asynchronous circuit design techniques can provide a path to obtain robust circuits easier to follow than through the use of synchronous design techniques. It investigates the evolution of methods and tools to deal with the construction of robust and energy-efficient digital systems based on asynchronous circuits. The main, although not sole emphasis, is placed on the employment of quasi-delay insensitive (QDI) design techniques, a concept presented and discussed in Section II. This Section covers some basics on asynchronous design, contrasting these with those of the conventional synchronous circuit design paradigm. The next Sections review three classes of approaches that suggest methods and tools to achieve robust design of asynchronous digital circuits. Section III covers robustness to voltage variations, followed by Section IV, which briefly discusses techniques to achieve robustness to process and temperature variations. Even if robustness to circuit aging is often cited as a potential field where asynchronous design can be successfully employed [9], [10], the authors could not find any technique described in the current literature that takes advantage of asynchronous circuits to explicitly combat aging. Section V accordingly explores the potential of asynchronous design to deal with digital systems aging, together with a discussion about the limitations of QDI design techniques in guaranteeing delay insensitivity to circuits. Follows Section VI that briefly presents Pulsar, an open-source asynchronous QDI circuit design flow. This flow is an original proposition of the authors' research group. The same Section explores some preliminary results on using Pulsar to design energy efficient circuits. Finally, Section VII brings a set of conclusions.

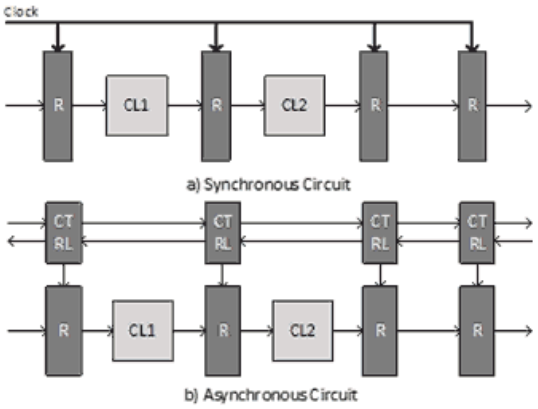

Fig. 1. Simplified linear pipeline circuit structure using (a) synchronous and (b) asynchronous design. Blocks CL_i represent combinational logic, R represent registers, and $CTRL$ indicates control logic. Adapted from [11].

II. A Primer on Asynchronous Circuits

Most synchronous circuits rely on the assumption that the value on the inputs of all its registers will only be sampled at the rising (or/and falling) edge of the clock signal. Refer to Figure 1(a) to notice that in a classic linear pipeline this enables to define timing constraints for the maximum delay in combinational logic paths, which must be typically smaller than the clock period. Using synchronous design techniques allows ignoring gate and wire delays, as long as clock timing constraints are respected. In other words, combinational logic is allowed to switch as it computes data during, say, the interval between two consecutive rising clock edges, but the logic outputs must be stable and correct at each such edge. Having this simple model for circuit design is possible only because the clock is a global and periodic signal, *i.e.* its edges only occur at specific and known points in time, and occur simultaneously at every point of the circuit where it is required. Hence, in synchronous circuits, events only take place at specific moments; time can thus be treated as a discrete variable.

However, a look at Figure 1(b) shows that in asynchronous circuits there is no such thing as a single clock to simultaneously signal data validity on the inputs of all registers. Here, events can happen at any moment, and time must, quite often, be regarded as a continuous variable. Asynchronous designers rely on local handshake protocols for communication and synchronization, and on different design templates to build circuits, each with its own specific assumptions about gate and wire delays [7].

Asynchronous design templates can be broadly classified in two main families: bundled-data (BD) [12] and quasi-delay insensitive (QDI) [13]. Refer to Figure 2 to note that the design of a BD circuit is similar to a synchronous one; the difference is that BD relies on carefully matching the delay of data path combinational logic blocks and controlling registers to the delays in the control block that locally generates a local clock, rather than to employ a single, global clock signal.

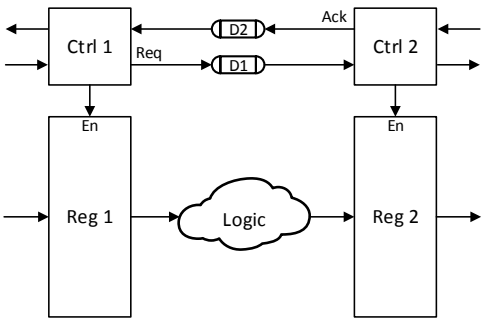

Fig. 2. Example of a typical BD asynchronous pipeline fragment, with delay elements explicitly represented as $D1$ and $D2$ on request (Req) and acknowledge (Ack) paths on the control part of the circuit; blocks $Ctrli$ represent the local stage controllers. Blocks $Regi$ are data path registers and the $Logic$ cloud represents combinational data processing between pipeline temporal barriers (the registers).

Communication and synchronization in BD circuits are accomplished through some handshake scheme, the more common choices being 4-phase, return to zero (RTZ) protocols [7]. Data representation in BD circuits follows the same Boolean encoding used in synchronous circuits[1]. Also, unlike what happens in synchronous circuits, controllers are local and usually comprise just a few logic gates. An illustrative extreme example is the very efficient MOUSETRAP pipeline stage controller, which includes only an $XNOR$ logic gate and a 1-bit latch [14].

Clearly, a major hurdle in BD circuit design is how to guarantee that the control and data paths are always precisely delay-matched, since the data and control flows typically run parallel to each other. This is the reason why in Figure 2 delay elements (DEs) are explicitly shown in the request and acknowledge paths. Much research exists to further the design of DEs to achieve working BD circuits. As an illustration of such research efforts, Heck [15] developed a PhD Thesis where the focus was obtaining a single programmable delay element to support the design of asynchronous BD circuits resilient to timing errors. This was in fact the culmination of a joint research between a research group at the University of Southern California in USA and the authors' research group, which had previously generated research results on several aspects of DE design for BD circuits [16]–[19]. Specifically addressed design aspects in the cited publications are analysis and optimization of programmable DEs, studies on how fine-grained and coarse-grained delay adjustments perform in practice, and to control the effect of voltage variations over the delay-matching characteristic of DEs.

The required delay-matching design effort is one of the main issues to design robust circuits using the BD family of templates. BD circuit implementations can be as small as an equivalent synchronous implementation, or even smaller [20]. However, BD techniques share some of the disadvantages that plague synchronous design techniques, including a potential reduction in circuit robustness to variations, mostly due to the decoupling of control and data parts of the circuit.

A. Quasi-Delay Insensitive Design

A fundamental difference between BD and QDI design is that the latter relies on data encoding schemes that allow data to carry their own validity information, which enables receivers to compute the presence or absence of data at inputs/outputs, and renders possible the local exchange of information in a mostly delay insensitive way, matching control and data information processing more easily. Because of this characteristic, QDI circuits can adapt more gracefully to wire and gate delay variations, and are thus one of the best choices to achieve robust circuits. On the negative side, QDI furthers robustness often at the expense of larger area and/or power overheads.

[1]This is not the case for QDI circuits, as Section II-A details.

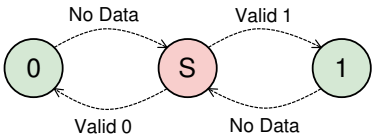

Fig. 3. The basic state transition diagram for transmitting binary data in a DI channel. Here, S stands for the spacer or bit separation symbol.

QDI design relies on the use of the so-called delay-insensitive (DI) codes [21]. DI codes use only part of the Boolean encoding spectrum possible over n bits. An n-bit code length allows representing 2^n distinct codewords. A DI code pledges the use of only a subset of these to achieve delay insensitivity. Verhoeff [21] explores the basic details of the theory behind DI codes. These include some codewords to represent valid data, and at least one special (invalid) codeword to represent the absence of data. This codeword is usually called a *spacer*. Since valid codewords and spacer codeword(s) do not comprise all 2^n different codewords possible with n bits, it is clear that some codewords are wasted as invalid and the matter of *code efficiency* arises. This is treated in the work of Verhoeff [21], which defines the *rate* R of a code. Given a code with M valid codewords and a length of n bits its rate is $R = (log_2 M)/n$. Of course, $0 \leq R \leq 1$ always holds, and Verhoeff proves that the Sperner codes, those where every codeword has as structure $(n \,\mathbf{div}\, 2)$-out-of-n, are DI codes, and such codes provide the highest possible encoding efficiency. For example, if the code length is $n = 20$ bits, all codewords with 10 bits at 1 and 10 bits at 0 are valid Sperner codewords, and there are a total of $184,756$ distinct codewords in this code. Even though this is much less than the $1,048,576$ codewords of a non-DI, 20-bit ordinary Boolean code, this Sperner code is much more efficient than the commonly used dual-rail DI code with length 20 bits that contains only $1,024$ valid codewords. As it can be verified, although as $n \to \infty$ the rate of Sperner codes tends to 1, practical n-bit Sperner codes (and all DI codes) have $R << 1$, while a non-DI code such as the n-bit Boolean code has $R = 1$ for any n.

To understand how DI codes achieve delay independence, Figure 3 shows a basic state transition diagram for transmitting data on a 1-bit DI channel, establishing a protocol. Assume transmission always starts with a spacer (S). A transition from the spacer codeword to 1 (or to 0) characterizes the transmission of a valid 1 (resp. 0) and a transition from 1 (resp. 0) to S characterizes the removal of data. In other words, DI communication protocols assume there is a spacer between any pair of consecutive data values.

This in fact depicts just a specific family of communication protocols, often associated to DI codes, that can be called *return to spacer* (RTS) protocols. Since the spacer is frequently a code with all bits in 0, a more commonly used term is *return to zero* (RTZ) protocols, although other spacer codewords are sometimes used.

Referring back to Figure 1(b) and its relation to the protocol that Figure 3 depicts, QDI circuit pipelines can be implemented using one of two approaches: (i) half-buffer,

where data and spacers alternate occupying successive pipeline stages; (ii) full-buffer, where all stages can contain data at every moment. Although at first counter-intuitive, since half-buffer schemes seem wasteful, these are more frequently used, because they are faster and simpler to built. Also it is easier to achieve robust circuits using half-buffer schemes.

In circuit design, often used examples of DI codes are the k-of-n codes, where n is the number of wires used to represent data (or its absence) and k is the number of wires that must be at a given logic value for the codeword to represent valid data. Albeit different codes are available in the contemporary literature (see e.g. [21]), according to Martin and Nyström [13] the most practical class of DI codes is the 1-of-n (or one-hot), and more specifically the 1-of-2 code. The latter is the basis to form codes to represent any n-bit information using two wires to denote each of the n bits, producing the so-called *dual-rail* code. Furthermore, Martin and Nyström argue that DI codes can be coupled to either 2-phase or 4-phase handshake protocols, but 2-phase protocols often lead to more complex circuits. Thus, 4-phase is frequently chosen by QDI circuit designers. In fact, the majority of QDI designs available in the state-of-the-art, from networks-on-chip [22], [23], to general purpose processors [24], and network switches [25], primarily rely on 4-phase protocols and dual-rail or 1-of-4-based codes[2]. The 1-of-4 code is equivalent to two 1-of-2 codes considered together, but these codes are different. In fact, switching a 1-of-4 codeword (say 0100, corresponding to decimal 2) to a 0000 spacer implies switching just 1 bit, while the same value encoded in dual-rail, 1001 (equivalent to 10 in binary or decimal 2), requires two bits to switch to reach the same spacer. Thus, 1-of-4 codes present roughly a 50% switching power advantage over dual-rail encoding.

Figure 4(a) depicts a 4-phase dual-rail DI channel *D*, where a single bit datum is represented using two wires, *D.0* and *D.1* that together carry the datum value, and one signal *ack* to control data flow [3]. A spacer is encoded here as a codeword with all wires at 0. Valid data are encoded using exactly one wire at 1, *D.1*=1 for a logic 1 and *D.0*=1 for a logic 0. In this case, both wires at 1 is a codeword that does not correspond to any valid datum and is not used. Figure 4(b) shows an example of data transmission using this convention to demonstrate the control flow allowed by the *ack* wire combined to codewords represented in wires *D.1* and *D.0*. In this example, a sender provides dual-rail data in *D.1* and *D.0* to a receiver that acknowledges received data through *ack*. Communication starts with a spacer, all signals at 0. Note that the *ack* wire also starts at 0, signaling the receiving side is ready to get new data. Next, the sender puts a valid 0 bit in the channel, by raising the logic value of *D.0*, which is acknowledged by the receiver raising the *ack* wire. After the sender receives *ack*, it produces a spacer to end communication, bringing all data signals in the channel back to 0. The receiver then lowers its *ack* signal, after which another communication can take place. Due to its nature, which requires all signals to go to 0 before each new data transmission starts, this justifies the return-to-zero (RTZ) denomination for this protocol.

Another protocol for dual-rail QDI design is the return-to-one (RTO) protocol [26]. RTO is similar to RTZ, but its

[2]Note that 1-of-2 (resp. dual-rail) and 1-of-4 codes have the same rate, $R = 1/2 = 0.5$, while a 1-of-8 code has a rate of just $R = 3/8 = 0.375$ and is accordingly never used.

[3]Some works prefer the use of *true* and *false* suffixes instead of 0 and 1 to distinguish between the two wires of a 1-of-2 or dual-rail code, which would lead e.g. to the terminology D_f and D_t in place of *D.0* and *D.1*, respectively.

Wire	Spacer	Value 0	Value 1
D.0	0	1	0
D.1	0	0	1

(a)

(b)

Fig. 4. The RTZ, dual-rail channel operation: (a) data encoding; (b) example of data transmission waveform.

Wire	Spacer	Value 0	Value 1
D.0	1	0	1
D.1	1	1	0

(a)

(b)

Fig. 5. The RTO, dual-rail channel operation: (a) data encoding; (b) example of data transmission waveform.

data values are inverted compared to the latter. As Figure 5(a) shows, a spacer here is the codeword with all wires at 1 and valid data is represented by one wire at 0, *D.1*=0 for a logic 1 and *D.0*=0 for a logic 0. Figure 5(b) depicts an example RTO data transmission, which starts with all wires at 1 in the data channel. As soon as the sender puts valid data in the channel, the receiver may acknowledge it by lowering *ack*. Next, all data wires must return to 1 to denote a spacer, ending the transmission. When the spacer is detected by the receiver, it raises the *ack* signal and new data can follow. The idea behind the RTO protocol is simple but powerful and allows a better design space exploration for QDI circuits. It also permits optimizations in power [27] and robustness [28]. Furthermore, as demonstrated in [29], RTZ and RTO can be mixed in a same QDI design and the conversion of values between them requires only an inverter per wire. According to Martin and Nyström, in [13], such conversion is DI and does not compromise the robust functionality of a QDI circuit. This article refers to signals operating under the RTZ (RTO) protocol as RTZ (RTO) signals.

The reader may have noticed that the acronyms DI and QDI are both used in this Section. There is then a question that may arise on how these differ. In fact, every time a reference above to the DI acronym appears, it refers to a type of code, while QDI (from *quasi-delay insensitive*) always refers to a class of circuit design techniques. There is indeed a class of DI circuit design techniques. This stands for the most robust of all ways to design circuits, where any delay of any wire or any gate (or other logic components) is irrelevant to define the overall circuit functionality, meaning the functionality of the design is fully insensitive to delays in wires or gates. Unfortunately, this *ideal* class of designs was proven to be limited, too limited to be of practical use [30]. On the bright side however, a small compromise can produce a set of design techniques that are expressive enough to be used in the construction of any digital circuit and be *mostly* delay-insensitive. This compromise consists in constraining some selected wire forks in a design to be *isochronic*. Assume a wire fork has a source terminal and two sink terminals. Saying this fork is *isochronic* basically means the propagation delays from the source terminal to both sink terminals only differ by a negligible amount. The combination of DI data codes and the isochronic fork assumptions creates the QDI design paradigm. An early theoretical result showed that QDI design is Turing-complete, unlike the DI design paradigm [31], which opened the door to use QDI as a serious class of design techniques.

Of course, the simply stated isochronic fork concept can

be hard to control, specially in large circuits. Much research has been dedicated in the last decades to define, refine and employ QDI design to build circuits. A major problem with such designs is the lack of EDA support for the elaboration and verification of circuits. Section VI of this article briefly reviews solutions proposed by the authors to solve this problem.

III. Robustness to Voltage Variations

Currently, the nominal supply voltage for an IC core is around 1.0V, and can even go to 0.65V or less in more recent nodes [32]. Under certain circumstances, this supply can vary due to noise at the output of the voltage regulator which can cause overshoots or undershoots (around more or less 10%). Another phenomenon leading to supply voltage variations is IR drop, caused by the current flow over the parasitic resistance of the power supply intra-chip grid.

Among the constraints imposed by contemporary designs, power consumption has attracted growing attention. With the increasing investment on portable devices and, more recently, the deployment of the Internet of Things (IoT), together with emerging applications such as distributed sensor networks and wearable devices there is a perception that low energy consumption is the key factor during circuit design [33]–[35].

In addition, with decreasing feature size, transistors have become increasingly leaky, augmenting static power dissipation, and challenging designers to meet power constraints [36]. This has motivated the research for new design techniques to minimize power as much as possible.

These efforts usually focus on high performance strong inversion in the super-threshold region of operation of transistors, and are implemented at the architectural level, where designers can reduce the computational workload or improve the architecture to achieve better power optimizations [35]. At the circuit level, a compelling approach to lower power consumption is reducing the voltage supply, called voltage scaling (VS). As the supply voltage is quadratically related to the dynamic power, VS is a very effective low power design technique [37]. Taking this design option to the extreme, some low power systems operate in the subthreshold region of transistor operation [38]. This allows achieving drastic power reductions, although with heavy performance penalties.

A. Subthreshold Regime Operation

The subthreshold effect is present in MOS transistors when the gate to source V_{GS} voltage is equal to or lower than the threshold voltage V_{th}. When $V_{GS} \leq V_{th}$, ideally, the transistor should be off and no current would flow through the transistor drain. In practice, the transistor still leaks a small current, usually denominated *subthreshold* current. In most digital applications, the subthreshold current is caused by parasitic leakage currents and is, accordingly, undesirable. This is because leakage is a deviation from the ideal switch-like behavior of the MOS transistor [39] [40]. If a circuit is powered by a voltage source where $V_{dd} \leq V_{th}$, then the circuit is on subthreshold operation – also said it is operating in the subthreshold region.

Subthreshold operation is a well known technique to reduce both static and dynamic power consumption [33], [38], [39], [41]. The downscaling of the supply voltage leads to quadratic savings in dynamic power and linear savings in static power. Due to these savings, subthreshold operation fits well in digital applications that must rely on energy harvesting or on limited power supplies such as small batteries. Nonetheless, this operation regime brings a significant performance degradation and higher sensibility to process, voltage and temperature (PVT) variations, which narrows the set of applications that can benefit from it [39], [42].

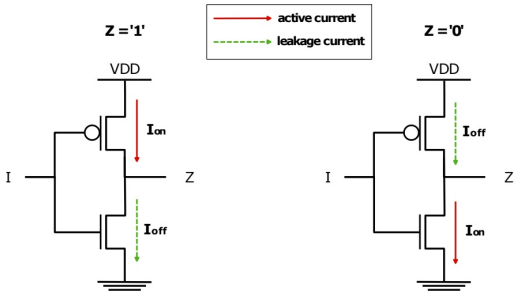

Fig. 6. Active (Ion) and leakage (Ioff) current representation in a basic CMOS inverter.

In a CMOS standard cell, the output fall and rise transitions rely on the Ion/Ioff ratio, where Ion is the active current and Ioff is the leakage current [42]. Figure 6 illustrates these currents for an inverter, depending on the output logic level. When Z = 1, the pull-up network (PUN) is enabled and is responsible to generate Ion, whereas the pull-down network (PDN) is disabled but stills generates Ioff . When Z = 0, the roles change: the PUN is disabled (Ioff) and the PDN is enabled (Ion). If the relation Ion/Ioff is too small, the PDN or PUN may not have enough strength to drive the logic level of the output, making the circuit fail. Due to the use of reduced supply voltages and lower active currents, subthreshold operation implies in smaller Ion/Ioff ratios. This interferes in the performance of subthreshold circuits, increasing delays by up to several orders of magnitude. Moreover, PVT variability, transistor sizing and channel doping (among other effects) are responsible for variations of the threshold voltage, which directly affects transistor currents and the Ion/Ioff ratio [42].

Most QDI asynchronous circuits rely on the use of sequential logic gates. The simplest such gate is the 2-input C-element that outputs 0 when both inputs are 0, outputs 1 when both inputs are 1 and keeps the output unchanged for any other input combinations. There are several ways to implement C-elements and other sequential gates composing libraries for QDI asynchronous design, for this see e.g. references [43] and [44]. However, three basic transistor topologies stand as the most accepted and employed in practical circuits: Martin's [45] Sutherland's [46] and van Berkel's [47] C-elements. Moreira & Calazans [48] approach these three C-element topologies to evaluate voltage scaling effects over them. Figure 7 shows the associated symbol for a C-element and the transistor level schematic for two of the three mentioned implementations: Martin´s and Sutherland´s[4].

These two implementations are available with different driving strengths (the capability of charging/discharging output loads) in a standard-cell library called ASCEnD-ST65. This library employs a commercial 65nm bulk CMOS technology, and uses a specially developed cell library design flow [49], [50]. The C-elements of the ASCEnD-ST65 library are all designed to the layout level and count with timing, power and functional models to support automated design and analysis of asynchronous ICs. For instance, Figure 8 and Figure 9 show the layout of a small drive (X2) version of the Martin and Sutherland C-elements. Also, RC extracted views are available

[4]The van Berkel topology is less generic and is accordingly ignored here, but data about such C-elements appear in [48].

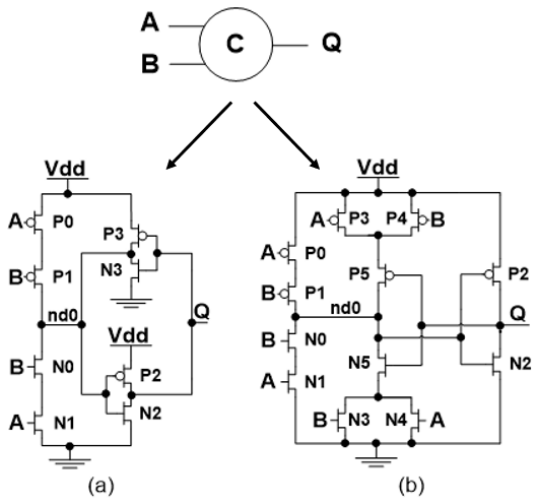

Fig. 7. Two alternative CMOS transistor topologies for a 2-input C-element: (a) Martin´s and (b) Sutherland´s.

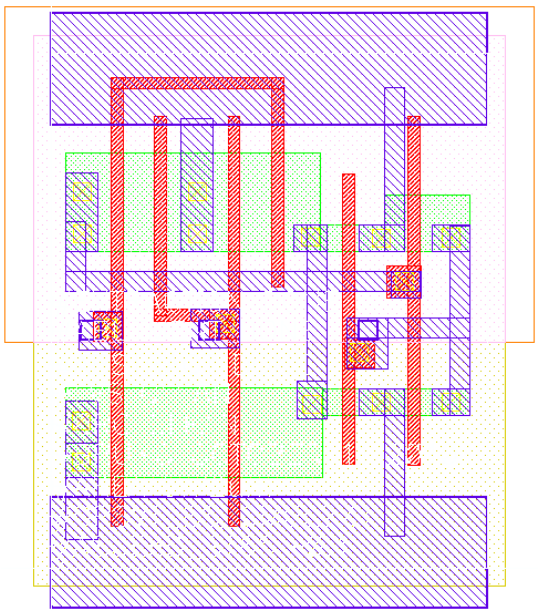

Fig. 8. Martin´s X2 C-element layout in the ASCEnD-ST65 library.

for different fabrication processes, based on layout parasitics. All the experiments reported in [48] are based on RC extracted views for a typical fabrication process.

B. Minimum Operating Voltage

One of the experiments that Moreira & Calazans [48] report detects the minimum voltages that can be applied to each C-element without interfering in its correct behavior. The experiment investigates scenarios for varying temperatures and a fixed fan-out of four (FO4) output load. Minimum voltages were obtained by simulating all transition arcs of each C-element for each temperature/voltage scenario. When at least one arc does not generate the correct output or a static state is not able to maintain correct functionality, the scenario is defined as not functional. Also, generated signals must have voltages in well defined regions, for logic 1 or for logic 0. If a signal presents a voltage level in the undefined region, the scenario is also defined as not functional. In summary, the minimum operating voltage is defined as the lowest voltage

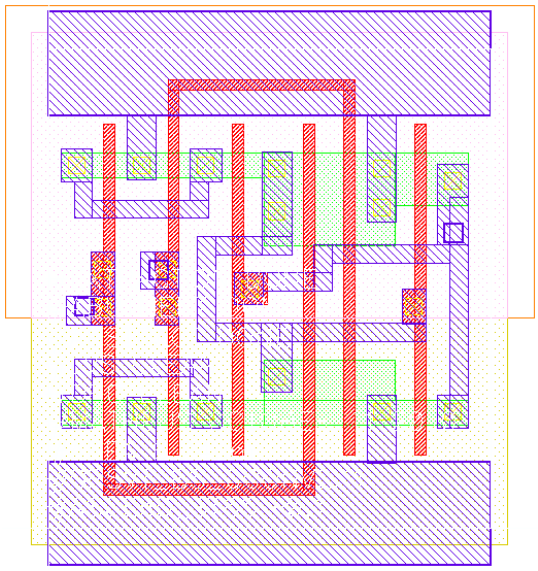

Fig. 9. Sutherland´s X2 C-element layout in the ASCEnD-ST65 library.

Drive ↓ / Temp.→	125ºC	100ºC	75ºC	50ºC	25ºC	0ºC	-25ºC	-50ºC
X2	0.15	0.15	0.15	0.2	0.5	0.6	0.65	0.65
X4	0.15	0.15	0.15	0.2	0.5	0.6	0.65	0.7
X7	0.15	0.15	0.15	0.2	0.45	0.55	0.6	0.65
X9	0.15	0.15	0.15	0.2	0.2	0.35	0.5	0.6
X13	0.15	0.15	0.15	0.2	0.2	0.25	0.45	0.5

(a)

Drive ↓ / Temp.→	125ºC	100ºC	75ºC	50ºC	25ºC	0ºC	-25ºC	-50ºC
X2	0.15	0.15	0.15	0.15	0.2	0.2	0.25	0.25
X4	0.15	0.15	0.15	0.15	0.2	0.2	0.25	0.25
X7	0.15	0.15	0.15	0.15	0.2	0.2	0.25	0.25
X9	0.15	0.15	0.15	0.15	0.2	0.2	0.25	0.25
X13	0.15	0.15	0.15	0.15	0.2	0.2	0.25	0.25

(b)

Fig. 10. Minimum voltage for maintaining correct functionality of the two C-elements: (a) Martin's and (b) Sutherland's.

at which the C-element can function without jeopardizing its correct logical/electrical behavior. Results are summarized in Figure 10, where six distinct cell drives are analyzed.

It is easy to note here the influence of cell topology, drive and temperature combinations on the robustness of the library cells. As expected, Martin's topology are less robust to variations than Sutherland's topology.

IV. Robustness to Process and Temperature Variations

Current technology trends lead to an increasing impact of manufacturing process variations on the performance of digital circuits. Such variations are responsible for deviations in transistor attributes during its manufacture, such as impurity concentration, oxide thickness and diffusion depth. Non-uniform conditions such as small variations in temperature, pressure, dopant concentrations in the manufacturing process and the inherent limitation of the photo-lithographic process (which cause variations in the width W and length L), change the electrical properties of transistors, introducing variations in the threshold voltage from one transistor to another, for example. This makes the propagation delay vary along different parts of an IC.

As already proposed in the arena of synchronous design, deterministic methods have made room to probabilistic techniques to address process variations. In this sense, finite state machine models for synchronous design change to probabilistic models such as Markov chains. Also, static timing analysis (STA) is often substituted by statistical static timing analysis (SSTA) [51], [52]. In the asynchronous design domain the

problems faced by the advent of process variations has to be addressed differently, since the absence of well-defined temporal barriers makes it harder to partition an asynchronous design. However, adaptations of asynchronous timing models already started to deal with process variations, as proposed e.g. by Raji et al. [53]. The basic idea of the cited work is to change the often used deterministic Timed Petri Nets (TPN) model to describe asynchronous concurrent behavior by probabilistic models such as the *Variant-Timed Petri Net* (VPTN) model [53].

Temperature variation is also a factor that directly affects the performance of a circuit and its occurrence is inevitable during IC operation. The transistor junction temperature impacts the transistor current flow. Switching a large number of transistors at the same time within an IC creates a local temperature variation on the chip due to power dissipation. As the temperature increases, the random movement of carriers in the channel increases due to the collision effect, degrading the mobility of these carriers. Therefore, the lower the temperature, the more random movements decrease and, due to better mobility, current increases and delay decreases.

Asynchronous QDI design is potentially beneficial to spatially and temporally distribute heat inside an IC, given its use of local handshakes. To illustrate how this can be useful, refer to the work of Hollosi et al. [54]. This work compares synchronous and QDI equivalent circuits regarding thermal distribution in 3D ICs. Measurements show that the QDI circuit exhibits smaller minimum, average and maximum temperatures in all layers of a 3D circuit, when compared to the synchronous version.

V. Perspectives on Aging and Limitations of QDI Design

The aging process causes delays of a circuit to change in a non-uniform manner along the circuit lifetime. Although QDI circuits offer resilience against delay variations, uneven delay variations may cause local violations of the isochronic fork assumption [55]. This assumption states that all branches in some sensitive wire[5] forks must complete transitioning at roughly the same time before a transition in any of the branches is acknowledged in a handshake operation. Isochronic forks are necessary whenever a transition on a wire fork origin does not cause a transition on every gate output connected to the fork destinations (the fork fan-out gates).

Figure 11 illustrates this problem on the DIMS expansion of a 2-input NAND gate (i.e. a dual-rail NAND gate); here the bold lines indicate the path that causes the output $y.t$ to activate when the inputs $a.f$ and $b.f$ are enabled. The dashed lines represent the hidden paths, i.e. the branches of the fork that are activated but do not cause a transition in the output of gates with this input configuration. As already explained in the discussion around Figure 3, this circuit alternates between the propagation of valid data and spacers in a QDI circuit. If the hidden paths are slow enough, they may linger high whilst the bold activation path goes low to propagate the spacer. This lingering value is not visible to the environment, as far as the spacer on the input is acknowledged; hence, it can now insert new valid data. If $a.t$ and $b.t$ are activated whilst the hidden paths from $a.f$ and $b.f$ still lingers high, both outputs $y.t$ and $y.f$ will be activated; this is an invalid codeword, constituting an invalid state for the circuit.

[5]Wires here are entities with a single input and possibly multiple outputs; this includes not only physical wires, but also delay elements, inverting and non-inverting buffers and/or buffer networks.

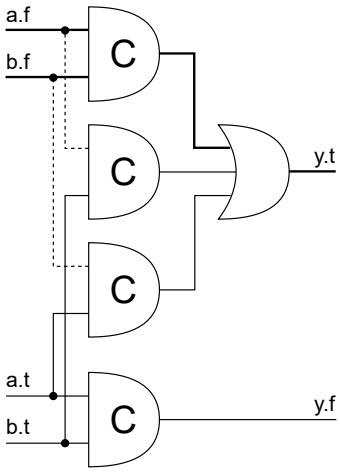

Fig. 11. A 2-input DIMS NAND gate. Notice the branches of an isochronic fork; the bold lines are the active acknowledged paths, the thin straight lines are the inactive paths and the dashed lines are the active hidden paths.

The original isochronic fork assumption imposes a too strict restriction; it assumes that some wire delays are negligible compared to gate delays. However, on modern technologies this does not hold true anymore; wire delays have often become dominant. Thus, a more modern restriction is required to cope with modern technology nodes. Under this light, Martin et al. [56] revise the isochronic fork assumption to lessen its requirements. These authors propose that the delay of the slowest branch in a sensitive fork must be less than the combined delay of the fastest *adversary path*, including gate and wire delays. In Figure 11, an adversary path to the highlighted hidden paths would be any path that causes $a.t$ or $b.t$ to rise once $y.t$ has fallen. The example circuit works correctly if the hidden paths are faster than the adversary path, i.e. if the hidden paths in $a.f$ and $b.f$ fall before $a.t$ or $b.t$ rise at the respective gate inputs.

Keller et al. [57] later formalized and proved the adversary paths timing assumption sufficient to guarantee the correct operation of QDI circuits. Raji et al. [58] designed an analysis framework to identify the critical sensitive forks that must obey timing restrictions.

This timing assumption is the limiting factor to QDI circuit's tolerance to delay variation. If aging, process or any other sources of delay variation are to change the relative delay of paths beyond the tolerated boundaries, the affected circuit can operate incorrectly. Dealing with isochronic forks is the main challenge in building robust asynchronous QDI circuits. Design methods and tools that can cope with the mentioned variations for asynchronous QDI design to deal with the problem just cited are still missing, and this is an open track for innovative research work. The focal point of such a work is to control the identification and treatment of the relevant circuit isochronic forks in a systematic way. The next Section approaches an ongoing work by the authors exploring the mentioned research track.

VI. Pulsar - A Method to Design QDI Circuits

Pulsar [59], [60] is an open-source [61] complete flow for synthesizing a constrained QDI circuit from an RTL-like input description. It employs commercial EDA tools to synthesize and optimize QDI circuits under cycle time constraints, and allows to trade-off performance and power targets. This is achieved by encompassing three key de-

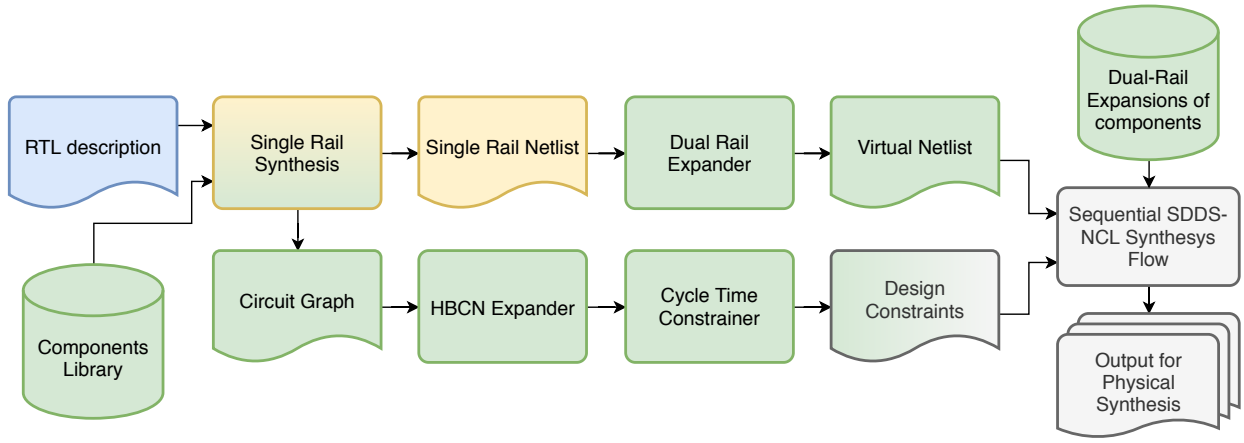

Fig. 12. The Pulsar flow. In blue is the user input, in an RTL-like format, in (System)Verilog or VHDL; yellow items are either third party (commercial) tools or conventional output by such tools; green items comprise the Pulsar frontend, covered in [59]; while grey items are initial Pulsar contributions, covered in [60].

sign and modeling techniques: (i) a QDI asynchronous design template called Spatially Distributed Dual-Spacer Null Convention Logic (SDDS-NCL) [62], [63]; (ii) the pseudo-synchronous Weak-Conditioned Half-Buffer (WCHB) temporal barrier model [60], [64]; and (iii) the Half-Buffer Channel Network (HBCN) timing model [60]. Technique (i) leverages conventional EDA tools to synthesize QDI circuits from Boolean virtual functions (V-functions); Model (ii) allows using standard STA tools to analyze the propagation paths in asynchronous pipelines (and additionally to accurately simulate synthesized circuits); The timing model (iii) enables the performance analysis of complex non-linear QDI circuits.

The overall Pulsar Synthesis Flow is depicted in Figure 12. Here, the flow begins with a single-rail synthesis, which produces a single-rail netlist. This netlist is processed and transformed into a virtual netlist comprising V-functions and pseudo-flops. The virtual netlist is the input to the Pseudo-Synchronous SDDS-NCL synthesis flow. The Pulsar flow also constructs the HBCN and automatically creates cycle time constraints for synthesis.

The design capture methodology shares similarities with the Null Convention Logic (NCL) Uncle synthesis tool [65], as it uses especially crafted RTL descriptions and traditional EDA tools to synthesize a single-rail netlist. However, Uncle relies on its own specialized tool for technology mapping and optimization of the NCL netlist, whereas the Pulsar Flow relies on SystemVerilog constructs and the in-house Sequential SDDS-NCL Synthesis Flow. The RTL netlist is synthesized using a traditional EDA tool (currently it supports the Cadence *genus* tool) to produce a single-rail netlist of *components*. These components are defined in the *components library* using the Synopsys Liberty format. This library contains the combinational and sequential components that can be used by the commercial EDA tool during the synthesis of the *single-rail netlist*. Each component has an equivalent SystemVerilog module defining its dual-rail expansion.

The dual-rail expansion takes advantage of SystemVerilog interfaces [66] to represent dual-rail four-phase RTZ channels. These channels interconnect modules implementing the dual-rail expansion of components. SystemVerilog interfaces are also used for constructing the acknowledgment network for channels. A simple tool replaces every wire in the single-rail netlist with a channel to create the *virtual netlist*.

Concurrent to the creation of the virtual netlist, cycle time constraints are computed. The scripts used for the single-rail synthesis produce a *structural graph* describing the pipeline topology. This circuit graph is used to model the HBCN of the expanded circuit.

A. Some Results on Using Pulsar

The results in this Section were extracted from [67]. To build asynchronous QDI circuits up to the layout level, Pulsar relies on a series of standard cell libraries containing SDDS-NCL cells, all developed by the authors' research group for several technology nodes. Results reported here refer to the ASCEnD-TSMC180 library for a 180nm bulk CMOS node. Also available as open-source there is the ASCEnD-FreePDK45 library [68]. Experiments demonstrate the potential of Pulsar to design circuits for aggressive voltage scaling. First, a 5-stage, 16-bit multiply-accumulate (MAC) unit was synthesized using the worst corner of the selected technology at $1.68\,\mathrm{V}$ and $125\,^{\circ}\mathrm{C}$, with slow transistors.

This circuit was subject to aggressive voltage scaling in analog simulation; it operates correctly under all tested supply voltages without any modification, from the nominal $1.8\,\mathrm{V}$ supply down to $500\,\mathrm{mV}$. Figure 13 depicts the results of average throughput and power for each voltage level. These show three-order magnitude reduction in power, whilst displaying only a two-order magnitude reduction in performance. Clearly, the design shows enhanced energy efficiency and robustness to voltage variations.

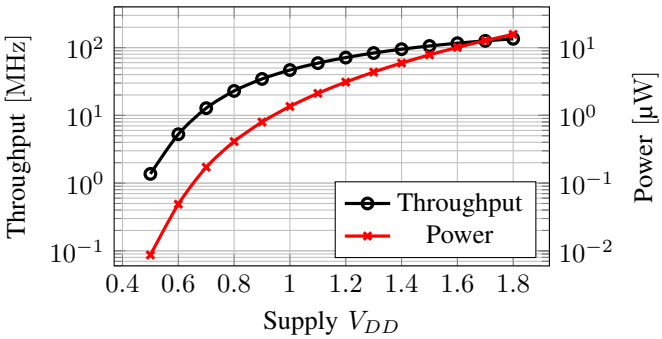

Fig. 13. MAC performance and power under voltage scaling.

Monte Carlo analog simulation was used to estimate the effects of process variation. Figure 14 shows that process variation causes greater timing variation under aggressive

voltage scaling. Nonetheless, the QDI circuit can tolerate it well, albeit being designed and synthesized to operate at nominal conditions only.

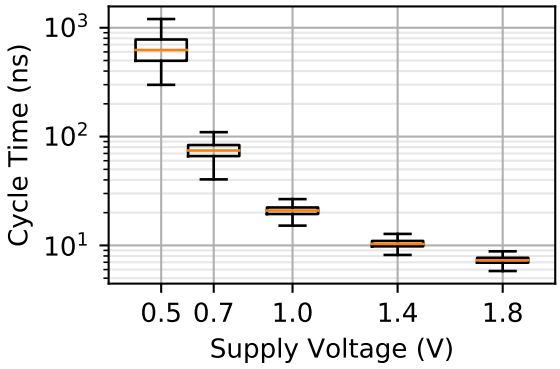

Fig. 14. MAC cycle times under process and supply variations.

An additional work on the use of Pulsar included the investigation of a spectrum of asynchronous first-in-first-out (FIFO) structures [69]. The idea of this work is to use Pulsar to automatically synthesize eight different classes of asynchronous QDI FIFOs, and compare these to trade area, throughput and power efficiency across combinations of data encoding, communication protocol and handshake control style.

A more recent work comprised the design and implementation of a 32-bit RISC-V RV32i processor organization [70]. This complete QDI implementation was very challenging and contributed to enhance the functionality of the tool to deal with complex asynchronous design issues such as choices, the use of mutual exclusion components, etc.

VII. Conclusions & Further Research

This article covers part of the state of the art in the design of robust and energy-efficient hardware with the help of asynchronous design techniques.

The evolution of technologies to manufacture digital systems is still evolving at an impressive path. Nano-electronics are a possible target of this evolution. In the words of Martin and Prakash [71] however, at the nanoscale level the technology "... does not allow to grow long wires reliably and wire delay is quadratic in the length. Consequently, it will be impossible to build a useful global clocking network with those technologies." What to expect then? Asynchronous design may be the way to go.

Beyond the research this paper reviews, there is a large body of work to undertake to support the smooth design of asynchronous circuits and systems. For example, the absence of investigations in the state of the art mentioning the use of asynchronous for combating IC aging indicates a clear research opportunity for those interested in asynchronous design breakthroughs and analysis.

There is a very relevant aspect of robustness omitted in the article for mere lack of space, which is the point of *robustness to faults and its relationship with asynchronous design*. QDI logic behavior under faults greatly diverges from synchronous circuits. Under stuck-at (SA) faults QDI circuits almost always deadlock; this outcome can be easily detected as the circuit simply stops functioning. This is a preferred behavior to produce erroneous values since it can be easily detected. However, transient faults, e.g. single event errors (SEEs), can lead to different predicaments. The wok of LaFrieda and Manohar [72] is an example. It explores the effects of transient faults in QDI circuits. These authors propose to model three possible failure types: (i) deadlock; (ii) token removal; and (iii) token insertion. Failures of type (i) are trivial and do not need special handling. Type (ii) failures do not produce a result and eventually can lead to circuit deadlock. Given enough time, faults of type (iii) generate an invalid token[6]; this invalid token may propagate causing havoc downstream. However, it is possible to employ techniques to detect and perhaps correct the propagation of invalid tokens. This is just an example of work done and still to be done on the subject of asynchronous design robustness to faults.

The class of quasi-delay-insensitive (QDI) asynchronous circuit templates is often considered one of the most practical approaches, because of its looser timing assumptions [13]. The flexible timing of these templates allows circuits to gracefully adapt to delay variations, making them attractive. Some QDI templates employ the Null Convention Logic (NCL), first proposed by Fant and Brandt in [73]. NCL is interesting due to its semi-custom nature, its reported low power and high robustness capabilities [74], and its validation in industrial designs by Theseus Logic [75]. Different works in the state of the art explore the usage of NCL circuits in applications such as low power design [76], [77]. However, a common pitfall for NCL designs is the reduced support by automated synthesis tools, technology mapping and optimization. Consequently, designs end up relying on conservative approaches. This usually occurs through the use of template-based methods, which restrict the employment of automated optimization and result in sub-optimal circuits.

Previous research proposed automated flows to design and optimize QDI circuits based on NCL, e.g. [65], [74], [75], [78]–[80]. Most of these perform logic optimization before technology mapping. However, the latter is precisely the first step in the synthesis process that enables improvements with realistic cost parameters from the target technology. Also, they rely on template-based approaches for technology mapping, i.e. synthesize the circuit as a single-rail version, and then replace logic gates by corresponding NCL templates (combinations of NCL gates). This often prevents exploring optimizations enabled by logic sharing and those specific to NCL. The main reason why these flows rely on such an approach is the limitation of the set of logic functions available in traditional NCL libraries. Such libraries only allow the usage of positive unate functions, reducing the gains achievable with EDA tools.

A few works propose optimizations either at the technology mapping level or at the logic synthesis level. Jeong and Nowick [74] suggest a method to allow post-mapping optimizations, which remaps predefined combinations of sets of gates in the netlist in a less conservative manner than previous works. The method first synthesizes a synchronous description, generating a clocked Boolean encoded netlist. The resulting netlist is then post-processed and has each of its gates expanded and directly mapped into macros of a corresponding dual-rail QDI cell library composed of NCL gates. Next, local handshake logic and wiring replaces the clock distribution. During the mapping of gates from the synchronous netlist to the corresponding sets of NCL gates, the authors propose the use of sophisticated algorithms that interpret the complete netlist as an unmapped logic network itself. This enables logic

[6]An invalid token in a dual-rail RTZ encoding occurs when the $D.0$ and $D.1$ wires in a channel are both high.

optimizations such as substituting some NCL gates by simpler Boolean gates. However, optimization is basic, again based on predefined logic templates, which impairs it from fully benefiting from consolidated logic mapping and optimizing algorithms implemented by commercial EDA tools.

Another work by the same authors [78] describes an optimization method for NCL circuits based on the relaxation of input completeness constraints of gates. The method allows substituting intermediate input complete gates by simpler gates without compromising circuit robustness. Results demonstrate gains in area and in delay. Later, local relaxation was also used in the Uncle tool [65]. Besides the cited Pulsar and Uncle tools, other design flows and systems are under proposition of have been proposed and used in the past. Proteus [81] was originally designed in the University of Southern California by the research group led by Prof. Peter Beerel, and was later licensed by Intel Inc., enhanced internally and used in the Intel Network Equipment division to produce several asynchronous IC designs based on the Pre-Charged Half-Buffer (PCHB) QDI asynchronous template [25]. Another similar effort lies in the Yale University, where a complete open-source EDA system for the design of asynchronous circuits is under development on the research group of Prof. Rajit Manohar [82]. This system includes a vertical set of tools, which include from low-level layout generation resources such as placers and routers, up to design capture at a high level of abstraction, such as the recently proposed Fluid tool, to be presented at this year's IEEE ASYNC Conference.

ACKNOWLEDGMENTS

This research was partially funded by CAPES and CNPq (grant 312917/2018-0), Brazilian funding organizations.

REFERENCES

[1] P.-C. Shen, C. Su, Y. Lin, A.-S. Chou, C.-C. Cheng, J.-H. Park, M.-H. Chiu, A.-Y. Lu, H.-L. Tang, M. M. Tavakoli, G. Pitner, X. Ji, Z. Cai, N. Mao, J. Wang, V. Tung, J. Li, J. Bokor, A. Zettl, C.-I. Wu, T. Palacios, L.-J. Li, and J. Kong, "Ultralow contact resistance between semimetal and monolayer semiconductors," *Nature*, vol. 593, no. 7858, pp. 211–217, 13 May 2021.

[2] S. Shigematsu, S. Mutoh, Y. Matsuya, Y. Tanabe, and J. Yamada, "A 1-V High-Speed MTCMOS Circuit Scheme for Power-Down Application Circuits," *IEEE Journal of Solid-State Circuits*, vol. 32, no. 6, pp. 861–869, 1997.

[3] N. Rohrer, C. Akrout, M. Canada, D. Cawthron, B. Davari, R. Floyd, S. Geissler, R. Goldblatt, R. Houle, P. Kartschoke, D. Kramer, P. McCormick, S. G, R. Schulz, L. Su, and L. Whitney, "A 480MHz RISC Microprocessor in a 0.12μm L_{eff}, CMOS Technology with Copper Interconnects," in *IEEE International Solid-State Circuits Conference (ISSCC)*, 1998, pp. 240–241.

[4] J. Lu, W.-K. Chow, and C.-W. Sham, "Fast Power- and Slew-Aware Gated Clock Tree Synthesis," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 20, no. 11, pp. 2094–2103, Nov 2012.

[5] S. Kim, I. Kwon, D. Fick, M. Kim, Y.-P. Chen, and D. Sylvester, "Razor-lite: A side-channel error-detection register for timing-margin recovery in 45nm SOI CMOS," in *IEEE International Solid-State Circuits Conference (ISSCC)*, Feb 2013, pp. 264–265.

[6] K. Bowman, J. Tschanz, N. S. Kim, J. Lee, C. Wilkerson, S. Lu, T. Karnik, and V. De, "Energy-Efficient and Metastability-Immune Resilient Circuits for Dynamic Variation Tolerance," *IEEE Journal of Solid-State Circuits*, vol. 44, no. 1, pp. 49–63, Jan 2009.

[7] P. Beerel, R. O. Ozdag, and M. Ferretti, *A Designer's Guide to Asynchronous VLSI*. Cambridge University Press, 2010.

[8] L. Doyen, T. A. Hentzinger, A. Legay, and D. Ničković, "Robustness of Sequential Circuits," in *International Conference on Application of Concurrency to System Design (ACSD)*, Jun. 2010, pp. 77–84.

[9] J. Beaumont, A. Mokhov, D. Sokolov, and A. Yakovlev, "High-Level Asynchronous Concepts at the Interface Between Analog and Digital Worlds," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 61–74, 2018.

[10] S. M. Nowick and M. Singh, "Asynchronous Design—Part 1: Overview and Recent Advances," *IEEE Design and Test of Computers*, vol. 32, no. 3, pp. 5–18, 2015.

[11] J. Sparsø, *Introduction to Asynchronous Circuit Design*. Independently published, 2020. [Online]. Available: https://orbit.dtu.dk/en/publications/introduction-to-asynchronous-circuit-design.

[12] I. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, 1989.

[13] A. Martin and M. Nyström, "Asynchronous Techniques for System-on-Chip Design," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1089–1120, 2006.

[14] M. Singh and S. M. Nowick, "MOUSETRAP: High-Speed Transition-Signaling Asynchronous Pipelines," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 6, pp. 684–698, Jun. 2007.

[15] G. Heck, "The Impact of Voltage Scaling over Delay Elements with Focus on Post-Silicon Tests," Ph.D. dissertation, PPGCC - FACIN - Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS), Mar. 2018.

[16] A. Singhvi, M. T. Moreira, R. Tadros, N. L. V. Calazans, and P. A. Beerel, "A Fine-Grained, Uniform, Energy-Efficient Delay Element for 2-Phase Bundled-Data Circuits," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 13, no. 2, pp. 1–23, Jan. 2017.

[17] R. Tadros, W. Hua, M. Gibiluka, M. T. Moreira, N. L. V. Calazans, and P. A. Beerel, "Analysis and Design of Delay Lines for Dynamic Voltage Scaling Applications," in *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, May 2016, pp. 11–18.

[18] A. Singhvi, M. T. Moreira, R. Tadros, N. L. V. Calazans, and P. A. Beerel, "A Fine-Grained, Uniform, Energy-Efficient Delay Element for FD-SOI Technologies," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2015, pp. 27–32.

[19] G. Heck, L. Heck, A. Singhvi, M. T. Moreira, P. Beerel, and N. L. V. Calazans, "Analysis and Optimization of Programmable Delay Elements for 2-Phase Bundled-Data Circuits," in *International Conference on VLSI Design (VLSID)*, Jan. 2015, pp. 321–326.

[20] J. Teifel, "Asynchronous Cryptographic Hardware Design," in *Annual IEEE International Carnahan Conference on Security Technology (ICCST)*, Oct. 2006, pp. 221–227.

[21] T. Verhoeff, "Delay-insensitive Codes - An Overview," *Distributed Computing*, vol. 3, no. 1, pp. 1–8, 1988.

[22] E. Beigné, F. Clermidy, P. Vivet, A. Clouard, and M. Renaudin, "An Asynchronous NOC Architecture Providing Low Latency Service and its Multi-Level Design Framework," in *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, 2005, pp. 54–63.

[23] J. Pontes, M. Moreira, F. Moraes, and N. Calazans, "Hermes-AA: A 65nm Asynchronous NoC Router with Adaptive Routing," in *IEEE International System on Chip Conference (SoCC)*, Sep. 2010, pp. 493–498.

[24] A. Martin, M. Nyström, and C. Wong, "Three Generations of Asynchronous Microprocessors," *IEEE Design and Test of Computers*, vol. 20, no. 6, pp. 9–17, 2003.

[25] M. Davies, A. Lines, J. Dama, A. Gravel, R. Southworth, G. Dimou, and P. Beerel, "A 72-Port 10G Ethernet Switch/Router Using Quasi-Delay-Insensitive Asynchronous Design," in *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, 2014, pp. 103–104.

[26] M. Moreira, R. Guazzelli, and N. Calazans, "Return-to-One Protocol for Reducing Static Power in QDI Circuits Employing m-of-n Codes," in *Symposium on Integrated Circuits and Systems Design (SBCCI)*, 2012.

[27] M. T. Moreira, R. A. Guazzelli, and N. L. V. Calazans, "Return-to-One DIMS Logic on 4-phase m-of-n Asynchronous Circuits," in *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2012, pp. 669–672.

[28] M. Moreira, R. Guazzelli, G. Heck, and N. Calazans, "Hardening QDI Circuits Against Transient Faults Using Delay-insensitive Maxterm Synthesis," in *ACM Great Lakes Symposium on VLSI (GLSVLSI)*, 2014, pp. 3–8.

[29] M. Moreira, J. Pontes, and N. Calazans, "Tradeoffs between RTO and RTZ in WCHB QDI Asynchronous Design," in *International Symposium on Quality Electronic Design (ISQED)*, March 2014, pp. 692–699.

[30] A. J. Martin, "The limitations to delay-insensitivity in asynchronous circuits," in *6th MIT Conference on Advanced Research in VLSI (AUSCRYPT)*, 1990, pp. 263–278.

[31] R. Manohar and A. J. Martin, "Quasi-Delay-Insensitive Circuits Are Turing-Complete," California Institute of Technology - CalTech, USA, Tech. Rep., 1995, rVM33.

[32] E. Sicard, "Introducing 7-nm FinFET technology in Microwind," INSA-DGEI, Toulouse, France, Tech. Rep. HAL-01558775 App Note, Jul. 2017. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01558775/document

[33] B. H. Calhoun, A. Wang, and A. Chandrakasan, "Modeling and sizing for minimum energy operation in subthreshold circuits," *IEEE Journal of Solid-State Circuits*, vol. 40, no. 9, pp. 1778–1786, 2005.

[34] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of things (iot): A vision, architectural elements, and future directions," *Future generation computer systems*, vol. 29, no. 7, pp. 1645–1660, 2013.

[35] C.-I. Ieong, M. Li, M.-K. Law, P.-I. Mak, M.-I. Vai, P.-U. Mak, F. Wan, and R. P. Martins, "Standard cell library design with voltage scaling and transistor sizing for ultra-low-power biomedical applications," in *IEEE International Conference of Electron Devices and Solid-State Circuits*. IEEE, 2013, pp. 1–2.

[36] P. Beerel, R. O. Ozdag, and M. Ferretti, *A Designer's Guide to Asynchronous VLSI*. Cambridge University Press, 2010.

[37] I. J. Chang, S. P. Park, and K. Roy, "Exploring asynchronous design techniques for process-tolerant and energy-efficient subthreshold operation," *IEEE Journal of Solid-State Circuits*, vol. 45, no. 2, pp. 401–410, 2010.

[38] N. Lotze and Y. Manoli, "A 62 mV 0.13µm CMOS Standard-Cell-Based Design Technique Using Schmitt-Trigger Logic," *IEEE Journal of Solid-State Circuits*, vol. 47, no. 1, pp. 47–60, 2011.

[39] B. Liu, M. Ashouei, J. Huisken, and J. P. De Gyvez, "Standard Cell Sizing for Subthreshold Operation," in *Design Automation Conference (DAC)*, 2012, pp. 962–967.

[40] J. M. Rabaey, A. P. Chandrakasan, and B. Nikolić, *Digital Integrated Circuits: A Design Perspective*. Pearson education Upper Saddle River, NJ, 2003, vol. 7.

[41] S. R. Vangal, S. Jain, and V. De, "A solar-powered 280mv-to-1.2 v wide-operating-range ia-32 processor," in *IEEE International Conference on Integrated Circuit Design and Technology (ICICDT)*, 2014, pp. 1–4.

[42] M. Pons, J.-L. Nagel, D. Séverac, M. Morgan, D. Sigg, P.-F. Rüedi, and C. Piguet, "Ultra Low-Power Standard Cell Design using Planar Bulk CMOS in Subthreshold Operation," in *International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*, 2013, pp. 9–15.

[43] M. Moreira, B. Oliveira, J. Pontes, and N. Calazans, "A 65nm Standard Cell Set and Flow dedicated to Automated Asynchronous Circuits Design," in *IEEE International System on Chip Conference (SoCC)*, Sep. 2011, pp. 99–104.

[44] H. K. O. Berge, A. Hasanbegović, and S. Aunet, "Muller C-elements ased on Minority-3 Functions for Ultra Low Voltage Supplies," in *IEEE Design and Diagnostics of Electronic Circuits and Systems (DDECS)*. IEEE, 2011, pp. 195–200.

[45] A. J. Martin, "Formal Program Transformations for VLSI Circuit Synthesis," in *Formal Development Programs and Proofs*, 1989, pp. 59–80.

[46] I. Sutherland, "Micropipelines," *Communications of the ACM*, vol. 32, no. 6, pp. 720–738, 1989.

[47] K. van Berkel, "Beware the Isochronic Fork," *Integration the VLSI Journal*, vol. 13, no. 2, pp. 103–128, 1992.

[48] M. T. Moreira and N. L. V. Calazans, "Voltage Scaling on C-elements: A Speed, Power and Energy Efficiency Analysis," in *IEEE International Conference on Computer Design (ICCD)*. IEEE, 2013, pp. 329–334.

[49] ——, "Design of standard-cell libraries for asynchronous circuits with the ASCEnD flow," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Aug. 2013, pp. 217–218.

[50] M. t. Moreira, B. S. Oliveira, J. J. H. Pontes, F. G. Moraes, and N. L. V. Calazans, "Adapting a C-element Design Flow for Low Power," in *IEEE International Conference on Electronics, Circuits and Systems (ICECS)*. IEEE, 2011, pp. 45–48.

[51] C. Visweswariah, K. Ravindran, K. Kalafala, S. Walker, S. Narayan, D. Beece, J. Piaget, N. Venkateswaran, and J. Hemmett, "First-order Incremental Block-based Statistical Timing Analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 25, no. 1, pp. 2170—-2180, 2004.

[52] W. Wang, V. Kreinovich, and M. Orshansky, "Statistical Timing Based on Incomplete Probabilistic Descriptions of Parameter Uncertainty," in *Design Automation Conference (DAC)*, Sep. 2006, pp. 161–166.

[53] M. Raji, B. Ghavami, H. Pedram, and H. R. Zarandi, "Process variation-aware performance analysis of asynchronous circuits," *Microelectronics Journal*, vol. 41, pp. 99–108, 2010.

[54] B. Hollosi, T. Zhang, R. S. P. Nair, Y. Xie, J. Di, and S. Smith, "Investigation and Comparison of Thermal Distribution in Synchronous and Asynchronous 3D ICs," in *IEEE International Conference on 3D System Integration (3DIC)*, 2008, pp. 58–68.

[55] A. J. Martin, "The Limitations to Delay-Insensitivity in Asynchronous Circuits," in *6th MIT Conference in Advanced Research in VLSI (AUSCRYPT)*, Mar. 1990, pp. 263–278.

[56] A. J. Martin and P. Prakash, "Asynchronous nano-electronics: preliminary investigation," in *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, 2008, pp. 58–68.

[57] S. Keller, M. Katelman, and A. J. Martin, "A necessary and sufficient timing assumption for speed-independent circuits," in *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, 2009, pp. 65–76.

[58] M. Raji and B. Ghavami, "Redressing fork constraints in nanoscale quasi-delay-insensitive asynchronous pipelines," *The Journal of Supercomputing*, vol. 74, no. 8, pp. 3820–3840, 2018.

[59] M. L. L. Sartori, M. T. Moreira, and N. L. V. Calazans, "A Frontend using Traditional EDA Tools for the Pulsar QDI Design Flow," in *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, 2020, pp. 114–123.

[60] M. L. L. Sartori, R. N. Wuerdig, M. T. Moreira, and N. L. V. Calazans, "Pulsar: Constraining QDI Circuits Cycle Time Using Traditional EDA Tools," in *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, 2019, pp. 114–123.

[61] M. L. L. Sartori, M. T. Moreira, and N. L. V. Calazans, "Pulsar - A Flow to Support the Design of QDI Asynchronous Circuits," Jun. 2020. [Online]. Available: https://github.com/marlls1989/pulsar

[62] M. T. Moreira, G. Trojan, F. G. Moraes, and N. L. V. Calazans, "Spatially Distributed Dual-Spacer Null Convention Logic Design," *Journal of Low Power Electronics*, vol. 10, no. 3, pp. 313–320, 2014.

[63] M. T. Moreira, P. A. Beerel, M. L. L. Sartori, and N. L. V. Calazans, "NCL Synthesis With Conventional EDA Tools: Technology Mapping and Optimization," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 65, no. 6, pp. 1981–1993, 2018.

[64] Y. Thonnart, E. Beigné, and P. Vivet, "A pseudo-synchronous Implementation Flow for WCHB QDI Asynchronous Circuits," in *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, 2012, pp. 73–80.

[65] R. B. Reese, S. C. Smith, and M. A. Thornton, "Uncle - An RTL Approach to Asynchronous Design," in *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, 2012, pp. 65–72.

[66] S. Sutherland, S. Davidmann, and P. Flake, *SystemVerilog for Design: A Guide to Using SystemVerilog for Hardware Design and Modeling*, 2nd ed. Springer Science & Business Media, 2006.

[67] M. L. L. Sartori, R. N. Wuerdig, M. T. Moreira, S. Bampi, and N. L. V. Calazans, "Leveraging QDI Robustness to Simplify the Design of IoT Circuits," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Oct. 2020, pp. 1–4.

[68] M. L. L. Sartori, M. T. Moreira, and N. L. V. Calazans, "ASCEnD-FreePDK45 - A Free Standard Cell Library for SDDS-NCL Circuits," Jun. 2020. [Online]. Available: https://github.com/marlls1989/ascend-freepdk45

[69] T. A. Rodolfo, M. L. L. Sartori, M. T. Moreira, and N. L. V. Calazans, "Quasi Delay Insensitive FIFOs: Design Choices Exploration and Comparison," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2021, pp. 1–5.

[70] W. A. Nunes, M. L. L. Sartori, and N. L. V. Calazans, "Enhancing an Asynchronous Circuit Design Flow to Support a RISC-V Processor Design," in *Submitted to ICCD'21*, 2021.

[71] A. J. Martin and P. Prakash, "Asynchronous Nano-electronics: Preliminary Investigation," in *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, 2021.

[72] C. LaFrieda and R. Manohar, "Fault detection and isolation techniques for quasi delay-insensitive circuits," in *International Conference on Dependable Systems and Networks (DSN)*. IEEE, 2004, pp. 41–50.

[73] K. Fant and S. Brandt, "NULL Convention Logic™: a complete and consistent logic for asynchronous digital circuit synthesis," in *International Conference on Application Specific Systems, Architectures and Processors (ASAP)*, 1996, pp. 261–273.

[74] C. Jeong and S. Nowick, "Technology Mapping and Cell Merger for Asynchronous Threshold Networks," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 27, no. 4, pp. 659–672, 2008.

[75] M. Ligthart, K. Fant, R. Smith, A. Taubin, and A. Kondratyev, "Asynchronous Design using Commercial HDL Synthesis Tools," in *International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, 2000, pp. 114–125.

[76] M. Seok, G. Chen, S. Hanson, M. Wieckowski, D. Blaauw, and D. Sylvester, "Mitigating Variability in Near-Threshold Computing," *IEEE Transactions on Emerging and Selected Topics in Circuits and Systems*, vol. 1, no. 1, pp. 42–49, Mar. 2011.

[77] R. Jorgenson, L. Sorensen, D. Leet, M. Hagedorn, D. Lamb, T. Friddell, and W. Snapp, "Ultralow-Power Operation in Subthreshold Regimes Applying Clockless Logic," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 299–314, Feb. 2010.

[78] C. Jeong and S. Nowick, "Block-Level Relaxation for Timing-Robust Asynchronous Circuits Based on Eager Evaluation," in *IEEE International Symposium on Advanced Research in Asynchronous Circuits and Systems (ASYNC)*, 2008, pp. 95–104.

[79] A. Kondratyev and K. Lwin, "Design of Asynchronous Circuits using Synchronous CAD Tools," *IEEE Design and Test of Computers*, vol. 19, no. 4, pp. 107–117, Jul. 2002.

[80] F. Parsan, W. Al-Assadi, and S. Smith, "Gate Mapping Automation for Asynchronous NULL Convention Logic Circuits," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 1, pp. 99–112, 2013.

[81] P. Beerel, G. Dimou, and A. Lines, "Proteus: An ASIC Flow for GHz Asynchronous Designs," *IEEE Design and Test of Computers*, vol. 28, no. 5, pp. 36–51, 2011.

[82] R. Manohar, "An Open-Source Design Flow for Asynchronous Circuits," in *Government Microcircuit Applications and Critical Technology Conference (GOMACTech)*, Mar. 2019.