IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 12 June 2022, accepted 16 July 2022, date of publication 26 July 2022, date of current version 1 August 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3194028

SURVEY

Evolution of NULL Convention Logic Based Asynchronous Paradigm: An Overview and Outlook

DANYLO KHODOSEVYCH AND ASHIQ A. SAKIB, (Member, IEEE)
Department of Electrical and Computer Engineering, Florida Polytechnic University, Lakeland, FL 33805, USA
Corresponding author: Ashiq A. Sakib (asakib@floridapoly.edu)

ABSTRACT The synchronous design paradigm dominates today's semiconductor industry. However, this clocked approach is facing major challenges with today's high-speed, low-power design expectations, using processes with ever-increasing physical level variability. Several clock related issues surface in designs operating at higher frequencies, which make clock management increasingly difficult. Quasi-delay insensitive (QDI) asynchronous (clockless) designs have proved to be effective in circumventing the major limiting factors associated with the clocked designs. NULL Convention Logic (NCL) is one such QDI asynchronous design paradigm, which presents itself as a promising alternative to conventional synchronous circuits and has already found numerous commercial applications due to its low power, robust architecture, and ease of design reuse. This paper presents the evolution of NCL based asynchronous paradigm over the past two decades, primarily focusing on existing fundamental research in NCL design automation, spanning over NCL synthesis, optimization, testing, and verification. The methods are systematically analyzed to determine their limitations and future research directions.

INDEX TERMS Asynchronous circuits, design automation, logic optimization, null convention logic, testing, verification.

I. INTRODUCTION

Within the field of digital VLSI, the demand for low-power, high-speed, and miniaturized Integrated Circuits (ICs) is ever increasing. Recent advancements in the conventional synchronous (clocked) domain allow designs to operate at Gigahertz (GHz) level frequency range while requiring lesser area. However, nowadays most devices based on synchronous digital designs are becoming extremely power-hungry, where the clock accounts for a significantly large portion of the power consumed in these designs. Several clock-related issues, such as clock skew, clock jitter, complex timing analysis, etc., make clock management extremely challenging. Additional driver circuitry required for clock distribution further adds to the energy utilization and area overhead. Moreover, decreasing feature size and higher integration density result in significant power dissipation per unit area as well as more timing inconsistencies (e.g., stretching of timing margins) due to increased process variations at scaled technology nodes.

Asynchronous designs [1] present themselves as a promising alternative to conventional synchronous circuits, which are inherently robust against process, voltage, and temperature (PVT) variations and have been effective in circumventing the major challenges associated with the clocked designs [2]. This has resulted in the domain's growing popularity over the past few decades. The most recent 2013 international technology roadmap for semiconductors (ITRS) predicts asynchronous logic to account for more than 50% of IC global signaling in the multibillion-dollar semiconductor industry by 2027 [3], and the more recent 2018 IEEE International Roadmap for Devices and Systems (IRDS) lists asynchronous computing as a potential solution to reduce power consumption [4]. Quasi-delay insensitive (QDI) is one of the widely utilized implementation models of asynchronous circuits with distributed switching (i.e., switching is not

The associate editor coordinating the review of this manuscript and approving it for publication was Yang Liu.

TABLE 1. Summarization of gate level delay insensitive methods.

Methods	*Delay Analysis/ Matching*	*Completion Detection*	*Full Minterm Synthesis*	*State Holding Elements*	*Performance*	*Optimization Possibilities*
[10]	Not required	Required	Yes	C-elements	Average case	Not possible
[11]	Not required	Required	Yes	C-elements	Average case	Not possible
[12]	Not required	Required	Yes	C-elements	Average case	Not possible
[13]	Not required	Required	No	C-elements	Average case	Limited
[14]	Not required	Required	No	C-elements	Average case	Limited
NCL	Not required	Required	No	Threshold gates	Average case	Significant

triggered simultaneously at the clock edge), which is achieved through a well-defined control mechanism to preserve error-free functionality and synchronization over the operation period [2]. The unique architecture prevents any data from being overwritten in the absence of external synchronizing signals. QDI model utilizes a phased *request-acknowledge* control mechanism along with certain timing assumptions to ensure that the functionality is not compromised. It operates under the assumption that any wire or gate delay is unbounded, i.e., unlike synchronous circuits, minimal timing analysis needs to be performed as the worst-case scenario is not assumed. However, the model requires the isochronic fork assumption, which dictates that the wire delays are less than the logic element delays within the components [2], [5], [6]. NULL Convention Logic (NCL) [7] is one of the major QDI design paradigms, which has found numerous commercial applications due to their inherent advantages, such as excellent power performance, less electromagnetic interferences (EMI), less noise, robust architecture, and ease of design reuse [2], [8], [9]. Although NCL circuits have managed to establish a growing industrial interest, the widescale adoption has been primarily hindered due to 1) the lack of matured computer aided design (CAD) tools to support automated synthesis, optimization, testing, and verification, 2) area overhead due to architectural constraints, and 3) lack of human resources with related expertise. Over the last two decades, several notable research works have focused on addressing these limitations, whereas several funding agencies (like NSF) have supported projects that aimed at educating and preparing the next generation workforce in the domain, as well as facilitating cooperation between industry and academia. This paper presents a comprehensive review and comparative analysis of the existing fundamental research in NCL design automation, spanning over NCL synthesis, architecture and circuit level optimization, testing, and formal verification. The objective of this work is to critically analyze the existing works to determine their limitations and realize future research opportunities to aid current researchers in their efforts towards developing industry standard support tools, which can greatly reduce the productivity gap, thus reducing the time-to-market, and facilitate further industrial adoption.

The article is divided into six main sections. A brief overview of NCL circuits and their fundamentals are presented in Section II. The evolution of automated NCL design flows is illustrated in Section III. Section IV details the different transistor-level implementations of NCL threshold gates proposed over the years, compares their performances, and systematically analyzes the cost metrics associated with each implementation. Different NCL testing and verification schemes, their unique attributes, limitations, and overall challenges associated with NCL design validation is summarized in Section V, followed by conclusions and directions for future work in Section VI.

II. NCL BACKGROUND

Prior to NCL, there were several other gate-level delay insensitive (DI) methods [10]–[14]. Delay insensitivity was attained in those methods by either generating a comprehensive set of all minterms [10]–[12], constructing and combining smaller self-timed components [13], or utilizing multiple subnets for self-timed logic construction [14]. The common attribute in all these methods is the utilization of C-elements as state-holding components along with Boolean gates to implement delay-insensitive (DI) functionality [15]. NCL circuits, on the other hand, utilize a library of threshold gates to achieve delay-insensitivity and do not require the generation of all minterms. Table 1 lists the attributes of all the above-mentioned DI methods.

NCL utilizes multi-rail logic, most commonly the dual-rail logic, to eliminate timing references, and a 4-phased handshaking-protocol for synchronization and control. A 2-phase protocol could also serve as an alternative handshaking scheme [16]. Unlike Boolean logic, dual-rail encoding requires two wires per variable to represent one bit of information (i.e., logic '0' or '1'), thus simultaneously representing both literals of the variable. A dual-rail encoded variable, X, comprising of two wires/rails, X^0 and X^1, can assume any of the three values from the set {NULL, DATA0, DATA1}; where X^0 (and X^1)ϵ {0,1}. DATA0 ($X^0 = 1$ and $X^1 = 0$) and DATA1 ($X^0 = 0$ and $X^1 = 1$) are equivalent to Boolean logic '0' and '1', respectively. Both rails being equal to '0' corresponds to the NULL state, which serves as a spacer value between two different data fronts. $X^0 = X^1 = 1$ is an

illegal state as it violates the mutual exclusive principle of the dual-rail protocol, which dictates that both rails cannot be asserted simultaneously.

The NCL framework resembles the conventional synchronous framework arranged in a micro-pipeline fashion [17]. The framework consists of QDI NCL registers, combinational logic unit, and completion detection unit as depicted in Fig. 1. For combinational circuits, the NCL paradigm requires at least two sets of QDI registers and one completion detection unit per stage to generate the handshaking control signals, which together enable the architecture to maintain an alternating NULL and DATA sequence to distinguish between two different DATA wavefronts in absence of the clock [2]. The completion detection unit, comprising of a combination of NCL threshold gates with 2/3/4-input C-element like functionality, uses the next stage registers' acknowledge signals (*Kos*) to detect complete DATA/NULL sets, and outputs a single-bit signal, which is fed into the *request* input *(Ki)* of all previous stage registers to request for the next NULL/DATA sets. The NCL logic, including registers and completion detection, is comprised of 27 fundamental gates with hysteresis, i.e., state holding capability. The threshold gates together can implement any function of maximum four non-inverted variables. The 27 threshold gates could be categorized into two groups: weighted and non-weighted. A non-weighted gate is represented as *THmn*, where m ([1, n]) and n are the threshold and number of inputs, respectively. A weighted gate can be represented as $THmnWw_1, w_2 \ldots, w_r$, where w_r $(1 < w_r \leq m)$ is the weight corresponding to input r. A non-weighted gate with $m = n$ behaves like an *n-input* C-element and can be represented as *THnn*. Fig. 2a depicts an NCL TH13 gate with 3 inputs (*A, B, C*) and a threshold value of 1, which indicates that the gate will assert its output only when at least one of the three inputs is asserted. Therefore, the set equation of TH13 gate becomes $A + B + C$. A weighted TH24w22 gate is shown as an example in Fig. 2b. Two of the four gate inputs to the gate (*A, B*) have a weightage of 2, whereas the remaining inputs (*C, D*) have a weightage of 1. Hence, if one of *A* or *B* gets asserted, it will be sufficient to meet the threshold and assert the output. However, C and D both must be asserted to assert the output. Therefore, the set equation of TH24W22 becomes $A + B + CD$. NCL gates are primarily implemented using static CMOS architecture [18], which is a subject of its own optimization at the transistor level.

NCL circuits must adhere to two requirements to remain delay insensitive: *input-completeness* and *observability* [2]. An NCL circuit is said to be *input-complete* if all its outputs transition from NULL-to-DATA/ DATA-to-NULL only after all its inputs have transitioned from NULL-to-DATA/ DATA-to-NULL. There can be some exceptions to the input-completeness requirement. For instance, some paths in the NCL datapath can be *relaxed* by implementing some of the gates as Boolean functions (without hysteresis). Under a relaxed scenario, it is acceptable for some of the outputs to transition without having a complete input set present, as long as all the circuit outputs cannot transition before all inputs transition. Input-completeness is the subject of various research works as it introduces a lot of overhead and restrictions to NCL circuits. This indicates that working around the concept of input-completeness can yield potential optimizations in NCL circuits, which will be discussed in future sections. Observability requires that each gate in a combinational circuit that transitions is required to make at least one output transition, i.e., every gate transition needs to be observable at the output. An unobservable circuit may introduce *orphans* during operation [19]. An *orphan* is a transition on a wire or a gate that is not acknowledged by a transition on the primary output, which may result in erroneous functionality under some timing scenario (e.g., if the transition is too slow).

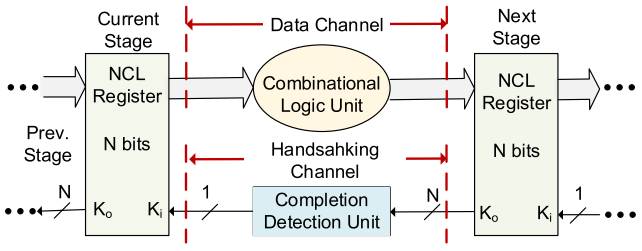

FIGURE 1. NCL framework.

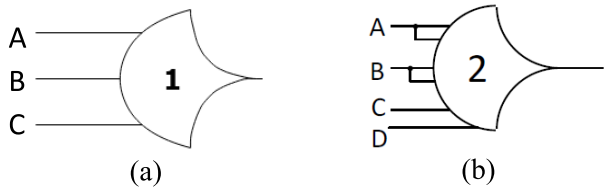

FIGURE 2. (a) TH13 gate, and (b) TH24W22 gate.

III. EVOLUTION OF NCL DESIGN FLOWS: LOGIC SYNTHESIS, OPTIMIZATION, AND TECHNOLOGY MAPPING

A. EARLY DESIGN FLOWS AND RELATED OPTIMIZATIONS

Synchronous designs have been dominating the industry for decades. As a result, the development of electronic design automation (EDA) tools over the years mostly focused on the clocked domain to cater the need of the industry. The NCL framework resembles the conventional synchronous framework, which enables designers to automate the synthesis of NCL circuits following similar processes as existing synchronous design automation. This promotes easier incorporation of NCL design flows into the semiconductor industry. As a result, NCL has become a more prominent asynchronous framework, especially since the inception of the first commercial design flow called NCL_D [20] in early 2000. The NCL_D design flow can perform logic synthesis using conventional CAD tools and primarily focuses on area

optimization. The method implements two synthesis steps starting from the circuits' Register Transfer Level (RTL) specifications. The Hardware Description Language (HDL) implementation (VHDL) of the specification circuit first goes through an initial RTL synthesis process, which results in a 3NCL gate level netlist. 3NCL netlist, consisting of 2-input Boolean gates, is a single-rail representation of the circuit that can accept one of the three values from the set {0, 1, N}. *N* is equivalent to a NULL signal in NCL framework and is treated as a *don't care* value by the compiler (Synopsys or Cadence). Therefore, the resulted 3NCL circuit is like a Boolean circuit, where the only difference is that Boolean circuits do not deal with three values in a single wire. In the second synthesis step, each wire gets converted to dual-rails and each 3NCL gate is expanded into a fully dual-rail 2NCL implementation using delay insensitive minterm synthesis (DIMS) technique. The 2NCL netlist mostly comprises of 2-input C-elements (i.e., TH22 gates) and OR gates, which guarantees input-completeness and observability by construction. Following the conversion to a 2NCL DIMS [21] circuit, the compiler prepares the circuit for further optimization, such as multi-level minimization of the Boolean network, targeting an NCL library.

Despite a significant step taken by the NCL_D design flow towards making NCL a more viable alternative to synchronous circuits, it suffered from large area overhead. This is mostly due to the optimization scopes being limited by the imposed delay-insensitivity constraints. NCL_X [22] emerged as a successor to NCL_D, building up on the parent design flow. NCL_X stands for NCL with *explicit completeness*, which modifies the design flow to handle the functionality and delay insensitivity separately. This partition allows for less restrictive, simplified, and independent optimization of logic and completion components. The NCL_X design flow also starts with logic synthesis from the RTL specification using synchronous CAD tools. After synthesis, the generated logic network (Boolean) gets converted to a unate network by replacing the direct value of a signal, *x,* by x^1 and inverse of x by x^0, followed by the dual-rail expansion. The unate network evaluates rail^1 of the circuit. For each gate in the rail^1 network, a dual gate is introduced in the rail^0 network as a part of the dual-rail expansion procedure. The two output rails of each dual gate pair in the combinational circuit are ORed together and then fed into a C-element network to generate the acknowledgement signals for previous register stages. The OR gates are termed as '*local completion detectors*. Unlike conventional NCL framework that contains a single completion detection unit per DI stage, NCL_X contains separate completion detectors for registers and logic unit per stage, as depicted in Fig. 3. The resulting combinational NCL circuit is delay-insensitive at each internal gate stage, thus allowing more flexible optimizations. Furthermore, in case some internal gates are acknowledged through the functional parts, their local completion detectors (OR gates) remain no longer required and can be removed for further area saving. The NCL_X flow permits more aggressive optimization of the combinational circuit itself through explicit completeness and significantly reduces the area overheard as compared to NCL_D.

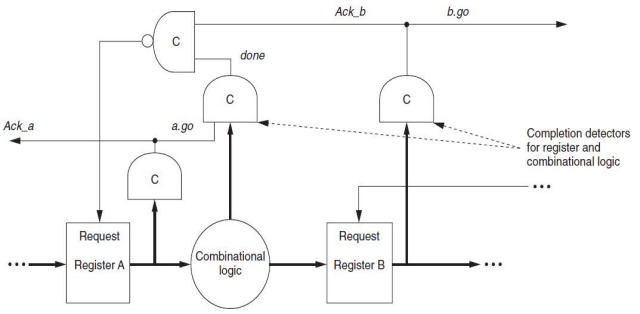

FIGURE 3. NCL_X framework [22].

B. INPUT COMPLETENESS RELAXATION

[23] proposed two design flows targeting further area reduction of NCL circuits based on the concept of *partial acknowledgment*. Any dual-rail signal within a circuit can be *partially acknowledged* by the output of at least one of the input-complete gates it fans into in both rising and falling transitions (where rising refers to a NULL-to-DATA transition and falling refers to a DATA-to-NULL transition). In NCL_D, all gate functions are input-complete; hence, a gate-input variable is acknowledged by *all* the outputs of the gates that it fans into. In contrast to NCL_D, [23] indicates that having a variable partially acknowledged in one of the fanout paths is sufficient. This opens possibilities for the gates in other fanout paths of the partially acknowledged variable to be considered for *relaxation*. Having more relaxed gates can significantly reduce the transistor count. Based on this concept, the first design flow targets area optimization, while also considering the induced overhead required for the verification of the circuit. The latter part of the work proposes another closely related design flow, which allows circuit design utilizing custom cells. The synthesized function modules can partially acknowledge signals in both rising and falling transitions. Results demonstrate that both the design flows can achieve a moderate reduction in area as compared to NCL_D and NCL_X. The second design flow performs better than the first in terms of area optimization but demands more aggressive verification. The partial acknowledgment concept is further extended in an automated synthesis flow in [24].

Chelcea et at. developed a method targeting area optimization of NCL circuits utilizing relative-timing analysis [25]. The method first observes the isochronic fork assumption and shifts to the notion that wire and gate delays are bounded (unlike QDI assumption). A minimal timing analysis is performed with the end goal of reducing the layout area through minimizing the number of completion detection units. Based on the timing interval calculations involving the comparison of the gate and wire delays with global propagation delays, several heuristic and optimal area optimization algorithms

were developed by the authors. The most advanced algorithm finds the optimal number of ***strict*** gates (i.e., input-complete gates) within the network. This indirectly improves the area utilization by finding the right balance between the minimum amount of completion detection units and input-complete gates. [25] reports a 2.4x improvement in area-utilization (on an average) based on their optimal algorithm, as compared to NCL_X [22]. However, due to the bounded-delay assumption, the synthesized circuits no longer remain QDI.

In many design flows, gate optimization by adding local completion detection units was preferred over an implementation with all input-complete gates, as the latter is associated with higher area cost. Jeong and Nowick [26] argue that a better optimization in terms of area utilization can be achieved by their method of *local relaxation* (termed as *Nowick's relaxation*), which does not require local completion detection, and where the number of *strict or input-complete* gates remains optimal. The method proposed in [26] has a similar objective to that of the works presented in [23] and [25]. In fact, the concept of *local relaxation* is very similar to *partial acknowledgment*. However, in contrast to those methods, which only focused on minimizing area, the optimizing algorithms in [26] target three cost parameters: number of input-complete gates, area, and critical path delay.

The algorithm minimizes the number of *strict* gates by ensuring that each primary input and intermediate gate output of the circuit is acknowledged only by a single fan-out point. For instance, Fig. 4 shows a 2NCL implementation of a 2-input XOR function, which is translated from a 3NCL representation. The 2NCL implementation starts as *strict* (Fig. 4a), where dual-rail input signals a and b are each acknowledged on two distinct paths, through input complete blocks X and Y. Fig. 4b depicts a less-restrictive implementation of the 2-input XOR function, where the input-complete block Y is *locally relaxed* by removing the C-elements, as acknowledging same inputs in multiple paths is redundant. Additionally, the method illustrates the possibilities of having multiple different choices for relaxation, which require consideration for evaluating the overall cost-function. The cost of expansion of each unique gate is also considered by the algorithm to further improve the area utilization. Moreover, the latency of the circuit can be optimized by identifying the longest path(s) and relaxing as many gates as possible in the path(s) to allow early evaluation (and reset) without compromising the delay-insensitivity.

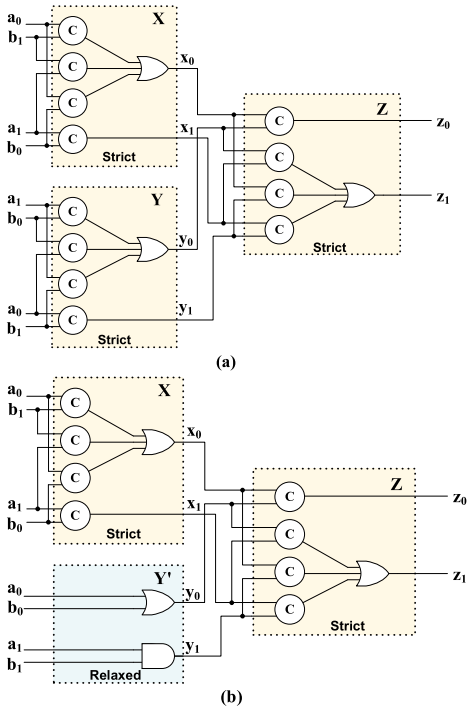

FIGURE 4. (a) Strict and (b) Relaxed implementations [26].

The concept of input completeness relaxation is extended to hierarchical designs in [27]. Unlike [26], the relaxation is not directly performed at the gate level, rather the technique aims to relax a *block* (comprised of multiple gates) within a circuit. The use of block-level relaxation can relax coarser-granularity nodes with more than one output, and the relaxation of a particular block inherently induces relaxation on its gate level implementation. The significant difference between block-level relaxation and Nowick's relaxation (of gates) is in the perspective from which the relaxation is performed. In addition to identifying the most suitable *block(s)* for relaxation within a given circuit, the algorithm in [27] finds the best *'partially eager'* implementation of each non-relaxed block. *Partially eager* indicates that within a multi-output input-complete block, some of the outputs can be *eager* (i.e., they can be evaluated early without compromising the delay-insensitivity of the overall block). Three *partially eager* approaches are used by the developed method to find the optimal relaxation of each block. The first approach is analogous to the Nowick's gate-level relaxation method [26]. The second approach is based on the idea that if a multi-output block has at least one path where all its inputs are acknowledged, then the remaining outputs in other paths can perform *eager* (i.e., early) evaluation. Finally, the third technique is a distributive approach towards implementing a multi-output input-complete block. In this approach, all outputs jointly ensure the input completeness for the block's inputs, where each output only acknowledges a subset of inputs (and remain *eager* for other inputs). Like [26], the algorithm decides how to implement each input complete block based on the cost functions and trade-off analysis.

In [28], Toms and Edwards utilize Nowick's gate-level and block-level relaxation methods and further extend those by incorporating them into an automated synthesis flow, which allows for a more efficient relaxation of a circuit network. The synthesis flow is initiated by forming clusters of gates (blocks) within the network utilizing a clustering algorithm to employ the block-level relaxation procedure as an intermediate step. The aim is to maximize the number of gates in a cluster while not going over the input limit of gates. After the circuit has been partitioned into clusters

of gates, the block-level relaxation synthesis starts off by utilizing the prime indicant synthesis algorithm [29]. The algorithm finds the minimal non-overlapping implementation of input-complete gates that acknowledge the signals within each cluster. The algorithm also determines the transitions acknowledged by different functional blocks (i.e., the number of variables one block must acknowledge), followed by a procedure to reduce indicants. As compared to [27], the fully automated procedure can apply block-level relaxation at much finer granularity, which results in further improvement in terms of speed of operation as well as area and energy utilization.

C. MORE RECENT NCL DESIGN FLOWS AND ADVNANCED TECHNOLOGY MAPPING

UNCLE [30] (Unified NCL Environment) is a relatively recently developed, custom, end-to-end toolset that facilitates the automated synthesis of NCL circuits. The design flow is very similar to NCL_D, in that the flow initiates from the RTL description of the circuit, which is synthesized to a netlist of Boolean gates using commercially available synthesis tools (supports Synopsys and Cadence Encounter RTL compiler). In case of NCL_D, the responsibility lies with the designer to indicate the location of register components, which are replaced with NCL registers during synthesis. However, UNCLE flow does not require the designers to explicitly locate the register components in the RTL description, which makes it more flexible. The Boolean gate netlist, i.e., the 3NCL netlist, then undergoes dual-rail expansion, where gates and registers are mapped to their equivalent dual-rail NCL implementations. The handshaking signals (acknowledge and request) are generated and connected as per the NCL architecture and four-phased handshaking protocol, followed by a series of optimization procedures (e.g., gate and block level relaxation [26], [27], cell merging, etc.). UNCLE flow is based on a *data-driven* approach, where there is no separate control network, except one dedicated *acknowledge* network. However, the flow could be extended to support *control-driven* design style, like BALSA [31, 32], as demonstrated in [30]. In contrast to the *data-driven* approach, the *control-driven* approach has a dedicated control network, which is separate from the datapath. The UNCLE design flow also supports the synthesis of Multi-threshold NULL Convention Logic (MTNCL), commonly referred to as SLEEP Convention Logic (SCL). SCL is a variant of NCL that targets low power applications, where the concept of multi-threshold CMOS is implemented to further reduce the leakage power dissipation. SCL logic units, registers, and completion units utilize an additional *sleep* signal that can immediately reset a particular stage to NULL, i.e., unlike NCL, a separate NULL wavefront is not required to be propagated through the pipeline after each DATA propagation. [2] and [33] explain the SCL architecture and its operation in detail.

Bhaskaran [34] developed another automated NCL design flow, which is quite different from NCL_D and UNCLE. [34] initiates from the synchronous RTL description of the circuit. Like NCL_D (and unlike UNCLE), the method requires the register locations to be explicitly specified. However, in contrast to existing design flows, this flow requires the combinational blocks to be partitioned into sub-modules by imposing constraints, such that synthesized circuits remains cost-effective in terms of area and delay. After partitioning, the combinational modules are synthesized to Boolean sum-of-product (SOP) representations, instead of generating the Boolean gate netlist, which is the major distinguishing feature of this methodology. The single-rail SOP representation is then expanded to dual-rail, ensuring input-completeness and observability, followed by further custom optimization procedures. A custom mapping algorithm is developed to map the fully optimized SOP expressions to NCL gates. Fig. 5a and 5b highlight the distinctions between [30] and [34].

In the abovementioned design flows, all methods ([20], [30], and [34]) implement their own grouping, mapping, and merging algorithms during synthesis to further optimize the logic network. Few works exist in literature that specifically focus on improving the gate-mapping and cell-merging procedures. For example, Jeong and Nowick proposed a technology mapping and cell-merging algorithm in [35], which supports post-mapping optimizations. The method successfully demonstrated its significance by further improving the performance metrics (area, delay, and power) when incorporated into the NCL_D design flow. Parsan *et al.* [36] proposed a gate mapping automation technique, which they applied on the last stage of synthesis of [34], by replacing the original grouping and mapping algorithm. The proposed gate mapping could achieve up to 10% area improvement and up to 39% improvement over delay, as compared to the original optimization algorithm used in [34]. [36] made a case that the method could be integrated or used as a stand-alone tool in any custom NCL design flow.

All the NCL synthesis procedures mentioned before use custom design flows. This is because NCL threshold gates do not directly map to the conventional standard cell libraries as provided by commercial design automation frameworks. An emerging semi-custom design flow is proposed and discussed in detail in [37] to bridge this gap in technology mapping. The design flow has its own approach to technology mapping of a logic network, where a network, specified in VHDL/Verilog, is mapped to a limited set of NCL gates using the concept of Boolean Virtual Functions (BVFs). Each NCL gate is defined based on its equivalent Boolean function in its on-set as well as off-set, which masks the sequential behavior (i.e., state holding capability) to tactfully employ existing CAD tools. The method that is used to map a logic network to a particular set of gates is based on an existing work proposed in [38]. An NCL Virtual Library is then generated from the Boolean netlist. It is at this point where the design flow in [37] starts to differ significantly from the other works from literature, in that it can use conventional CAD tools to adjust the mapped network on the circuit level

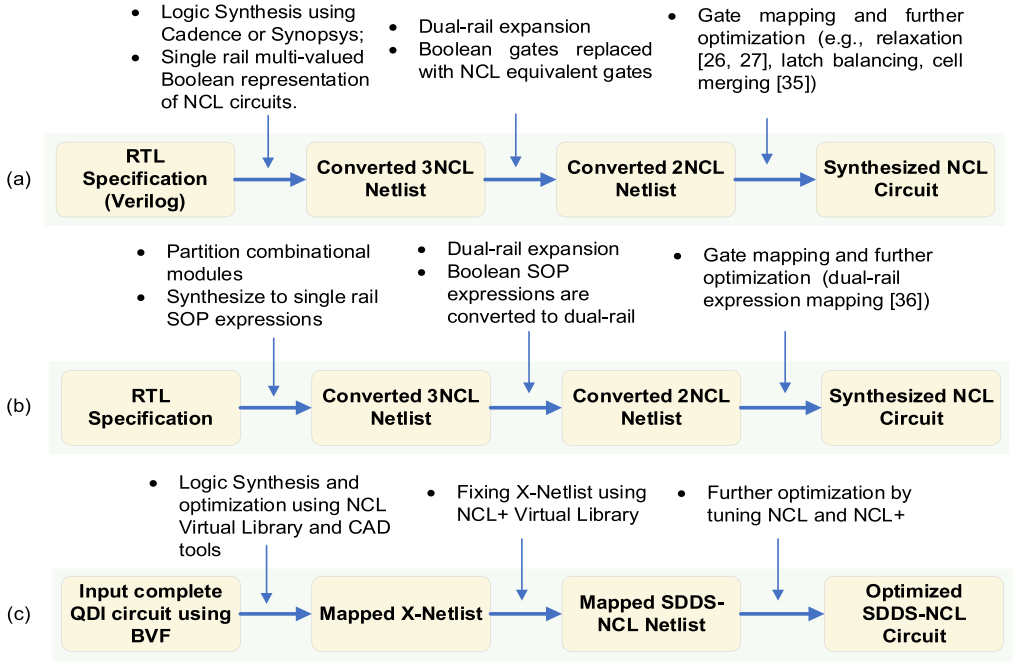

FIGURE 5. Automated NCL Flows: a) UNCLE [30], b) Bhaskaran *et al.* [34], and c) SDDS-NCL[41].

to account for timing and power constraints. However, previous works (e.g., [23], [26], and [27]) extensively discussed and warned about the possibility of generating gate *orphans* during technology mapping and logic optimization using commercial CAD tools, which could potentially generate a corrupted netlist. [37] addresses and tackles the issue of corrupted netlist generation using a library of NCL+ gates, which is a variant of NCL gates, and is based on the authors' previous work [39]. Unlike NCL gates that use Return-to-Zero (RTZ) handshake protocol, NCL+ gates are designed to work with Return-to-One (RTO) handshake protocol [40]. RTO and RTZ protocols are conceptually similar. The only difference is that the rail values are inverted. [37] establishes the possibility of a mixed implementation of NCL and NCL+ gates within the same network and demonstrates that the replacement of NCL gates with NCL+ gates within the logic network does not compromise the observability, since the on-sets of NCL+ (NCL) gates cover all the inputs coming from an RTO (RTZ) domain. The work is further extended in [41], where both NCL and NCL+ are employed to form a template called *Spatially Distributed Dual Spacer* NCL (SDDS-NCL). A synthesis flow is further developed based on the SDDS-NCL template.

In [41], a distinction is made between NCL and NCL+ gates to be able to differentiate them better based on the concepts of unate functions [42]. As per [41], "*for each positive unate NCL (NCL+) gate, a negative unate gate can be defined, where the latter has its OFF-SET defined as the ON-SET of the former*". A negative unate NCL (NCL+) gate is termed as inverted-NCL or *INCL* (*INCL+*). The NCL Virtual Library, consisting of *NCL* and *INCL* gates, is used to synthesize an NCL circuit. In case the synthesized circuit exhibits incorrect functionality, it is then fixed by the NCL+ Virtual Library, consisting of NCL+ and INCL+ gates.

The corruption of the netlist often stems from designs that contain inverting elements. The inverting elements may invert the internal signals, thus creating certain signal domains that are more suitable for RTO protocol. Moreover, the change in the internal signal domain causes the traditional NCL gates with hysteresis to not transition in their primary outputs as their off-sets did not account for those transitions. For instance, let us assume that the half-adder circuit in Fig. 6a is in NULL state, i.e., all its input rails (*A.0*, *A.1*, *B.0*, and *B.1*) and output rails (*S.0*, *S.1*, *C.0*, *C.1*) are '0'. If *A* and *B* both transition to DATA1 (i.e., $A.1{=}B.1{=}1$ and $A.0{=}B.0{=}0$), then the NAND gate (G3) will compute '0', which will make $C.1{=}1$ due to the NCL inverter. However, the gate G8 that receives the output of G3 as input, will not transition to '1' due to the hysteresis of 2W11 gate (equivalent to an inverted TH22 (TH22n) gate). That indicates that the netlist has been corrupted as *S.0* will not become '1'.

As per the logic synthesis flow of SDDS-NCL, NCL+ Virtual Library is utilized to 'fix' the corrupted netlists by replacing each NCL (INCL) gate, which contains an odd number of inverting elements on the path to one of its inputs, with an NCL+ (INCL+) gate. As a result, G2, G6, G7, and G8 gates from the NCL Virtual Library in the corrupted netlist (Fig. 6a) get replaced by their NCL+ counterparts (as shown in Fig. 6b). For the same combination of inputs, i.e., for *A* and *B* both being DATA1, the *fixed* half-adder (Fig. 6b) now computes correctly by enabling *S.0* to be '1'. Finally, the flow takes advantage of being compatible with commercial CAD

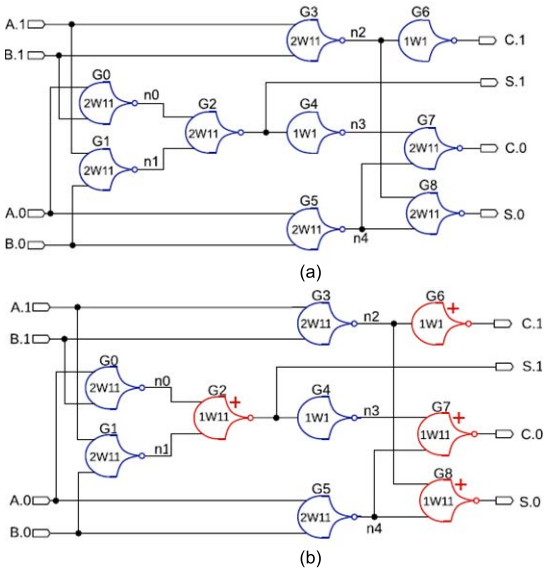

FIGURE 6. NCL mapped netlist: a) Corrupted netlist (X_netlist), and b) Fixed netlist [41].

tools to perform sophisticated timing and power optimizations as opposed to many other flows, which are limited in their circuit level optimization post mapping. The overview of the SDDS-NCL logic synthesis flow is depicted in Fig. 5c. A physical synthesis flow that synthesizes NCL circuits down to layout is also presented in [41]. However, the flow can only support the automated design of combinational NCL circuits.

Fig. 7 depicts a timeline to show the advancements in automated design of NCL circuits over the last two decades and summarizes the salient features, contributions, and limitations of these methods.

IV. EVOLUTION OF NCL BUILDING BLOCKS: OVERVIEW OF DIFFERENT NCL GATE IMPLEMENTATIONS

NCL threshold gates are the building blocks of NCL circuits. The combinational logic, storage elements, and completion detection units comprise of NCL threshold gates. Therefore, optimizations at the gate level can significantly reduce the overall area and energy utilization, as well as improve the speed of operation of NCL circuits. Various implementations of NCL gates have been proposed over the past two decades, such as static, semi-static, differential, etc. Additionally, several variants of these implementations also exist in the literature, which are discussed, compared, and analyzed herein.

A. STATIC NCL IMPLEMENTATIONS (S-NCL)

The static CMOS implementation [18] of NCL gates is most widely utilized as it offers robustness and better tradeoffs in terms of crucial performance parameters (area, power, latency, etc.) [43]. A static NCL threshold gate is comprised of four blocks: one *SET* block, one *RESET* block, and two *HOLD* blocks (*Hold0* and *Hold1*) [2], [18], [44], [45]. The output threshold function of a gate is determined by its *SET* block, which asserts the output based on the Boolean equation. The *RESET* block de-asserts the output when all inputs are de-asserted. *Hold0* and *Hold1* blocks are additional pull-up and pull-down networks, respectively, which account for hysteresis when neither *SET* nor *RESET* are true [2]. Fig. 8a shows a non-minimal CMOS implementation of TH23 gate. The *SET* function of TH23, $F_{SET} = AB + AC + BC$, is implemented by the *SET* block. The *RESET* block implements a function, where all inputs are complemented and ANDed together. *Hold0* and *Hold1* functions are the complements of *SET* and *RESET* functions, respectively.

The static NCL gates can operate correctly at very low supply voltages [43], [44], [45], and are efficient in terms of power and delay. However, these advantages come at a cost of area overhead, which is due to the additional *HOLD* blocks for state-holding. [45] demonstrates that transistors in *SET* and *Hold1* blocks (pull-down network) as well as *RESET* and *Hold0* blocks (pull-up network) can be shared to reduce the area utilization. Fig. 8b shows the minimal implementation of TH23 gate with shared transistors. [45] further identifies that in the actual static NCL implementation only the transistors in *SET* and *RESET* blocks contribute to switching (termed as *switcher* transistors). *HOLD* block transistors, termed as *keepers*, do not contribute to output switching. [45] introduces a new static implementation (we have termed it as new static NCL (New S-NCL) herein), which integrates pull-up (*RESET* and *Hold0*) and pull-down (*SET* and *Hold1*) network transistors to increase the number of switchers for faster operation. Fig. 8c. shows the speed-optimized static implementation of TH23 gate (keepers are in boldface), which enables higher speed of operation due to lower equivalent resistance in *SET* and *RESET* paths of the gate. Although this implementation improves the speed of operation and has symmetric rise and fall times, it generally has more transistors (on average ∼2.3 more transistors/gate) than the minimal (transistor-shared) implementation. However, the authors indicate that the resulting structure of the gates can allow better transistor sizing compared to the original implementation, thus reducing the area overhead imposed by the extra transistors.

B. SEMI-STATIC NCL IMPLEMENTATIONS (SS-NCL)

Fig. 8d shows a generic implementation of NCL gate in semi-static configuration. Instead of the *HOLD* blocks (*Hold0* and *Hold1*), the semi-static implementation utilizes a weak-inverter arrangement to maintain the current state of output when both *SET* and *RESET* are inactive. The elimination of the *HOLD* blocks significantly reduces the transistor count [2], [18], [43], [44]. However, the implementation requires complicated transistor sizing to ensure correct functionality. For instance, the weak inverters need to be strong enough to tolerate the noise in internal nodes, but not too strong to restrict the pull-up network from overpowering the feedback inverters. Therefore, determining the correct sizing of transistors in weak-inverters, *SET*, and *RESET* arrangements is non-trivial and requires extensive simulation. The PMOS transistors are required to be sized up considerably,

	Year	Proposed work	Salient Features, Contributions, and Limitations
Recent Advancements Towards Commercial NCL Synthesis	2018	SDDS-NCL Design Flow Moreira et al. [41]	• Semi-custom design flow for logic and physical synthesis. • Supports technology mapping of logic network using SDDS-NCL template. • Uses NCL virtual library for logic synthesis; tackles mapped netlist corruption using NCL+ virtual library. • Supports post mapping timing and power optimizations using CAD tools. • Limitation: supports automated synthesis of combinational NCL blocks only.
	2017	Zhou et al. [24]	• Further builds on the Partial Acknowledgment concept in [23]. • Develops a systematic synthesis and optimization flow based on Partial Acknowledgment mapping. • Targets area and delay optimization.
	2012	UNCLE Reese et al. [30]	• Initiates from an RTL specification, followed by a logic synthesis to generate a Boolean netlist. • Dual-rail expansion; Boolean gates are replaced with equivalent NCL gates. • Based on data-driven approach; however, can support control-driven style. • Integrates several optimization schemes (e.g., gate merging, relaxation, latch balancing, etc.).
Following Efforts Towards Cost-Aware Synthesis	2010	Toms and Edwards [28]	• Extends gate & block level relaxation; allows automated synthesis using Prime Indicant Synthesis algorithm. • Focuses on maximizing the shared acknowledgment; applies block-level relaxation at a much finer granularity.
	2008	Jeong and Nowick [27]	• Extends input completeness relaxation at block-level, comprising of multiple gates. • Introduces *Partial Eagerness*, allowing some of the outputs of an input-complete block to early evaluate. • Better area optimization as compared to local relaxation.
	2007	Bhaskaran et al. [34]	• Combinational blocks are partitioned into sub-modules and synthesized to Boolean SOP expressions. • Boolean SOP expression is expanded to dual-rail; input completeness and observability is ensured. • Dual-rail SOP expression is further optimized and mapped to NCL gates. • Targets area and delay optimization.
	2007	Jeong and Nowick [26]	• Formulates a novel method of *Local Relaxation*, termed as Nowick relaxation. • Very similar to the concept of Partial Acknowledgment. • Maintains an optimal number of strict gates without the requirement of local completion detection units. • Offers greater flexibility in terms of multi-parameter optimization (area as well as critical path delay).
	2007	Chelcea et al. [25]	• Area-aware optimization using *relative timing analysis*. • Heuristics based optimization algorithms: finds the minimum number of strict gates and completion units. • 2.4x smaller implementation, on an average, as compared to NCL_X. • Assumes gate and wire delays are bounded; hence, synthesized circuits are non-QDI.
	2006	Zhou et al. [23]	• Targets area optimization utilizing a newly introduced concept of Partial Acknowledgment. • Allows more gates in a circuit to be *relaxed*, which results in significant area reduction. • Limitation: increased verification demand.
Initial Efforts In NCL Synthesis	2002	NCL_X: Kondratyev and Lwin [22]	• Implements functionality and delay insensitivity separately within the design flow. • Allows optimization of individual gates by adding local completion detectors. • Separate completion detectors for combinational logic unit and registration unit in each stage. • Flexible for optimization; ~30% area reduction on average as compared to NCL_D.
	2000	NCL_D: Ligthart et al.[20]	• Custom synthesis of NCL circuits from RTL specification; relies on DIMS. • Simple translation scheme. • Less flexible for optimization; sharing signal acknowledgment is not allowed. • Significant area overhead.

FIGURE 7. Evolution of NCL design flows.

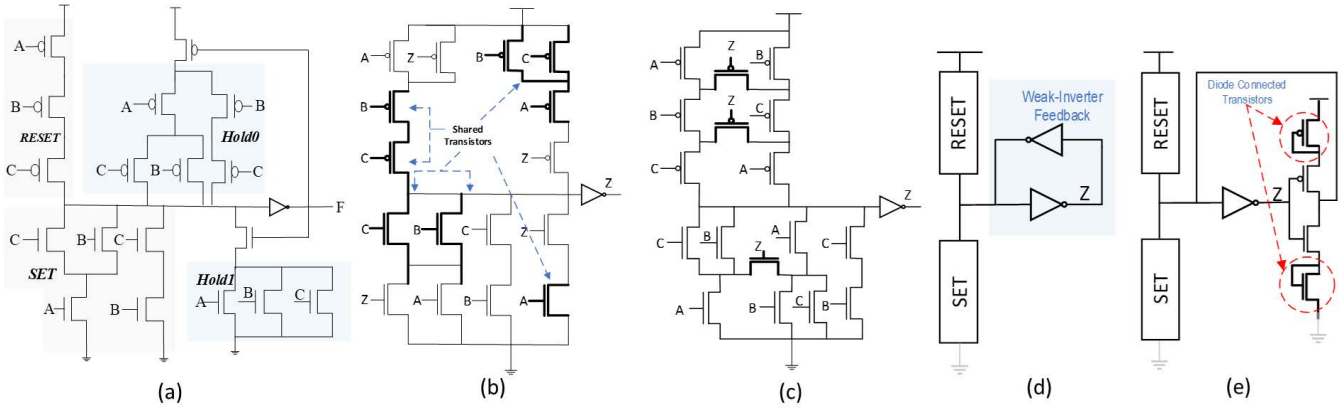

FIGURE 8. a) Non-optimized S-NCL TH23, b) Optimized TH23 with shared transistors, c) New S-NCL implementation of TH23 [45], d) SS-NCL [44], and e) SSDC-NCL [44].

which can significantly increase the overall silicon area, despite the lesser transistor count in SS-NCL implementation. Semi-static NCL implementation with diode-connected transistors (SSDC-NCL) is a variant of SS-NCL [44], where two additional transistors are introduced in the weak-inverter arrangement (Fig. 8e). These transistors behave like resistors,

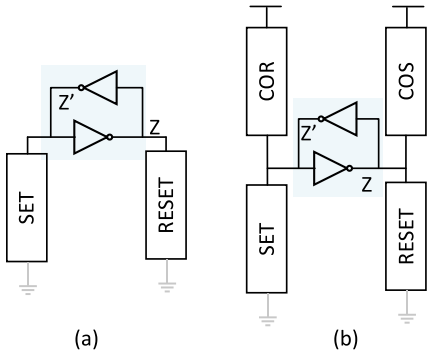

FIGURE 9. a) SS-DNCL, and b) S-DNCL.

which limit the current to weaken the inverters. SSDC-NCL implementation significantly improves the gate delay and energy consumption per operation, as compared to SS-NCL. However, SSDC-NCL gates are highly sensitive to noise. Several other weakening methods, such as resistive method and supply feedback method, have also been discussed and analyzed in [44]. SSDC-NCL implementation outperforms those methods in terms of power, delay, and energy consumption. Therefore, they have not been discussed herein.

C. DIFFERENTIAL NCL IMPLEMENTATIONS (D-NCL)

[46]–[48] explore a differential implementation of NCL gates (DNCL) based on differential cascode voltage switch logic (DCVSL) family of circuits [49]. There are primarily two distinguishing features that separate DCVSL from conventional logic: 1) there are two pull-down (NMOS) networks, where one implements the function, and another implements the complemented function; and 2) the pull-up network contains only two cross-coupled PMOS transistors. Both inverted and non-inverted inputs are required for functioning, and inputs are accepted only on the pull-down network. The differential NCL design implements *SET* and *RESET* blocks as the two pull-down networks [46]. Additionally, two NMOS transistors are combined with the pull-up PMOS transistors to form a cross-coupled inverter arrangement, as shown in Fig. 9a. Like SS-NCL, the cross-coupled inverters account for hysteresis; therefore, the implementation presented in [46] is termed as semi-static differential NCL (SS-DNCL). SS-DNCL design can operate at supply voltages comparable to the static design, requires less transistors, and maintains excellent performance and noise resistance [43]. However, the power dissipation remains high due to the large contention current between the cross-coupled inverters and the pull-down networks while setting or resetting the gates.

Lee and Kim proposed a modification over [46], where the two NMOS transistors forming the inverters were removed yielding a more conventional DCVSL implementation [47]. Although the design reduces the power dissipation, it fails to maintain hysteresis throughout the operation. A more recent work proposes a novel static DNCL design (S-DNCL), as depicted in Fig. 9b, which addresses the contention issues during switching in the SS-DNCL implementation [48]. The S-DNCL design adds two additional PMOS pull-up networks to the circuit, which aid the pull-down networks to change the gate output by disabling the inverters during switching. Like S-NCL, the S-DNCL structure also has four blocks, *SET*, *RESET*, *COS* (complement of SET), and *COR* (complement of RESET), where *COS* and *COR* assert the complement of output and de-assert the output, respectively. Due to the additional blocks, the S-DNCL implementation has higher transistor count as compared to SS-DNCL. However, S-DNCL manages to deliver lower delay and better power performance.

D. BEYOND CMOS NCL IMPLEMENTATIONS

All the different NCL implementations discussed before were based on MOSFET Technology. MOSFET technology has delivered improved performance with every level of scaling and integration, until recently when more aggressive scaling introduced major challenges in the deep-submicron region due to device level limitations. Multi-gate designs, like FinFETs, have the potential to tackle issues beyond conventional planar-bulk CMOS technology [50]. Sakib *et al.* explores the possibility of FinFET based NCL implementations to utilize FinFET's superior gate-controllability, low-voltage operation, and improved power performance at scaled technology nodes [51]. [51] utilizes double-gate FinFETs, which can function in one of three modes: 1) shorted gate (SG) mode, where both the gates are shorted; 2) independent gate (IG) mode, where both the gates can be used independently as if they are connected in parallel; and 3) low-power (LP) mode, where one gate's threshold can be modulated by the other. These three configurations of FinFET present interesting optimization opportunities. [51] explores these opportunities, analyzes different combinations of FinFET based static NCL threshold gates in different modes of FinFET operations, and formulates a generic design rule based on the simulation results. The design guideline suggests that the SG mode is the more suitable choice for switchers (*SET*/*RESET* block transistors) as it provides the lowest delay values due to high drain current. The high-power dissipation of SG mode could be compensated by configuring non-parallel keepers (*HOLD* block transistors) in LP mode. As the keepers do not contribute to switching, the lesser drive strength of LP configuration does not affect the overall speed much. The parallel keepers could be merged using IG mode to further reduce area overhead. [51] demonstrates that FINFET-NCL implementation improves the propagation delay, energy consumption, and require $\sim$2 fewer transistors per gate on an average. Fig. 10a depicts the optimized FinFET implementation of static TH23 gate.

[52] proposed a carbon nanotube field effect transistor (CNTFET) based implementation of static NCL gates (CNTFET-NCL). CNTFETs have several unique advantages over CMOS, which make them a promising alternative for digital designs, especially at scaled technology nodes. For instance, CNTFETs provide higher drain currents and

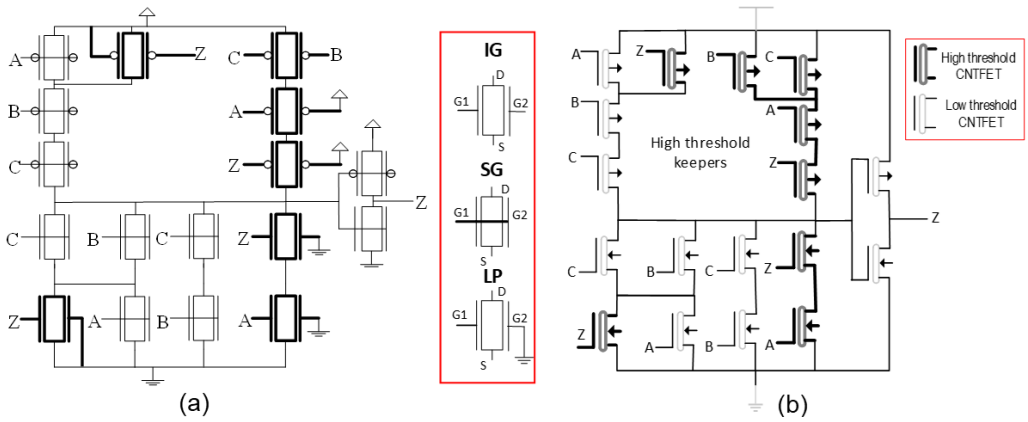

FIGURE 10. a) FinFET S-NCL implementation of TH23 [51], and b) CNTFET S-NCL implementation of TH23 [52].

demonstrate significantly better leakage power performance at deep-submicron level. Additionally, CNTFETs allow for easier threshold voltage adjustment by altering the diameter or the *chirality* vector of the nanotubes. The easier modulation of threshold voltage provides some design advantages, which are explored in [52]. In [52], a multi-threshold design approach was proposed, where the *switchers* were implemented with low threshold voltage CNTFETs for improving speed of operation, and *keepers* were implemented with high threshold voltage CNTFETs for limiting the *off* current for further leakage power reduction. The CNTFET implementation of static TH23 gate is shown in Fig. 10b. [52] validates that static CNTFET-NCL outperforms static CMOS-NCL in terms of speed, energy utilization, and leakage power dissipation. Both CNTFET and CMOS implementations have the same transistor count.

Bai *et al.* [53] proposed an NCL implementation, named spin torque enabled NCL (STENCL), based on emerging spintronic technology. The implementation is based on magnetic domain wall (DW) logic [54], which demonstrates fast switching and minimal off current. In STENCL, the authors exploit the inherent hysteresis property of DW devices to remove additional transistors in HOLD blocks in CMOS static implementation. Fig. 11 depicts a STENCL implementation of TH23 gate, where the DW shifts laterally from d_1 to d_3 (or d_3 to d_1) during a SET (or RESET) operation due to a current generated by the architecture. Details about the device physics can be found in [53]–[55]. Although STENCL implementation significantly improves the area utilization and energy consumption, the speed of operation is much slower as compared to static CMOS implementations. The design attributes and performance tradeoffs of the above mentioned NCL threshold gate implementations have been summarized and listed in Table 2.

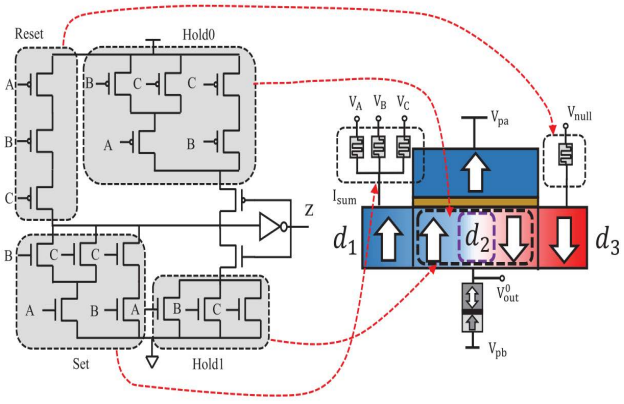

FIGURE 11. STENCL implementation of TH23 gate [53].

V. EVOLUTION OF TESTING AND VERIFICATION METHODOLOGIES FOR NCL BASED ASYNCHRONOUS DESIGNS

Testing and verification is an integral component of any ASIC design flow. As discussed in Section III, automated synthesis, optimization, and ease of simulation using CAD tools are crucial for widescale incorporation of NCL based asynchronous designs. However, that alone cannot ensure commercial adoption if the synthesized circuits do not provide efficient means for design validation. While numerous testing frameworks exist for the synchronous domain, testing of asynchronous circuits remains an area less explored. Moreover, conventional synchronous testing frameworks are not readily applicable to the asynchronous domain due to their unique architecture. Developing testing methodologies for NCL based asynchronous circuits has been historically challenging due to the following reasons:

- NCL circuits contain sequential components (e.g., gates with hysteresis, registration, etc.). As a result, NCL circuits, even the combinational ones, behave like synchronous pipelines, which creates a complicated testing environment.
- NCL based asynchronous circuits are nondeterministic in nature due to the absence of clock, which makes test timing management extremely challenging.
- The handshaking control paths within the NCL circuits contain feedback loops for synchronization, which deteriorate the test controllability and observability, and further complicate the design for testing.

TABLE 2. Different NCL threshold gate implementations.

Design	*Tech.*	*Attributes*	*Design Tradeoffs**			*Remark*
			Transistor Count	*Speed*	*Energy Consumption*	
Static NCL	CMOS	• Four blocks: *SET/RESET* implements gate assertion/de-assertion; *Hold0 and Hold1* maintain hysteresis.	**High**	**Fast**	**Moderate**	Robust; can operate at lower supply voltages
New S-NCL	CMOS	• Pull-up (RESET-Hold0) and pull-down (SET-Hold1) networks are integrated to have more *switchers.*	**Slightly Higher** than S-NCL	**Faster** than S-NCL	**Moderate**	--
SS-NCL	CMOS	• No *HOLD* blocks. • Weak inverter arrangement provides for state-holding.	**Much Less** than S-NCL. (Sized up PMOS→ increases area)	**Significantly Slower** than S-NCL	**Significantly Higher** than S-NCL	Transistor's sizing is crucial; requires extensive simulation
SSDC-NCL	CMOS	• Requires 2 additional transistors in the weak-inverter arrangement of SS-NCL to limit current; larger PMOS are not required.	**Less** than S-NCL	**Fast** (Comparable to S-NCL)	**Lower** than S-NCL	Highly susceptible to noise
SS-DNCL	CMOS DCVSL	• Both *SET* and *RESET* are in the pull-down network. • Hysteresis is provided by the weak inverter arrangement.	**Less** than S-NCL. (Sized up transistors→ increases area)	**Slower** rise time, but much **faster** fall time than S-NCL	**Higher** than S-NCL	Transistor sizing is crucial. Unbalanced rise and fall time
S-DNCL	CMOS DCVSL	• 2 additional pull-up networks are added to the SS-DNCL implementation. • Complicated transistor sizing is no longer required.	**Higher** than S-NCL	**Slower** rise time, but much **faster** fall time than S-NCL	**Higher** than S-NCL	Unbalanced rise and fall time
FinFET-NCL	Double Gate FinFET	• Different FinFET modes of operation are explored to improve speed, area, and energy parameters.	**Less** than CMOS S-NCL	Comparison data was not available	Comparison data was not available	Improved performance at scaled nodes.
CNTFET-NCL	CNTFET	• Easier modulation of CNTFET threshold voltage (V_{th}) is explored for a multi-threshold implementation. • *Switchers* and *keepers* are implemented with low and high V_{th} CNTFETs, respectively.	**Same as** CMOS S-NCL	**Much Faster** than CMOS S-NCL	**Much Lower** than CMOS S-NCL	Significantly reduced leakage power & improved performance
STE-NCL	Spintronic	• Based on emerging spintronic technology: magnetic DW logic.	**Significantly less** area	**Much Slower**	Significantly Lower than CMOS S-NCL	--

S-NCL = Static NCL; SS-NCL= Semi-static NCL; SSDC-NCL= Semi-static Diode Connected NCL, SS-DNCL= Semis-static Differential NCL; S-DNCL= Static Differential NCL; STE-NCL= Spin-Torque Enabled NCL.

* Design tradeoffs are reported based on the comparison of each listed implementation with the CMOS S-NCL implementation.

- Several stuck at faults in NCL gates' internal feedback paths do not necessarily result in incorrect outputs or deadlock, which may cause some of the faults to go undetected. Enhancing the controllability and observability to detect such faults is a non-trivial task.

Several researchers have developed testing methodologies addressing these limitations to some extent. Kondratyev *et al.* proposed one of the earlier testing methodologies to detect stuck-at faults using conventional CAD tool for test vector generation [56]. The work addressed the testing of both acyclic and cyclic NCL pipelines. The testing methodology for acyclic NCL pipelines was straightforward and utilized conventional fault analysis methods, such as fault grading and fault collapsing. Conventional automated test pattern generation (ATPG) tool was used to generate efficient test vectors. For test pattern generation, the acyclic NCL pipeline was converted to a single combinational network by removing the registration and completion units. The threshold gates were replaced with their equivalent Boolean gates. A set of test vectors were derived for the equivalent Boolean network to detect stuck-at-faults. The generated test vectors were applied to the original NCL pipeline following a sequence, where each DATA test vector was interleaved by a NULL to check stuck-at-0 and stuck-at-1 faults in SET and RESET phases, respectively. Dominance based fault collapsing was implemented separately to eliminate the faults in registration stage [57]. The authors demonstrated that stuck-at faults in the completion detection units are easy to detect; hence, the completion circuitry was not considered for testing.

The methodology to test cyclic NCL pipelines was more complicated due to the presence of feedback loops within the

datapath. The authors utilized a partial scan-based approach to break the cycles and insert test vectors. The design of an NCL scan register was presented in [56], where each scan register comprised of one D-latch, one C-element, and an AND gate. Each scan register had two inputs for mode configuration (serial, parallel, load, and normal), one scan input for testing, one data input for normal operation, one enables signal coming from the acknowledge output of the next registration stage, 2 non-overlapping clocks (resembling level sensitive scan design (LSSD) type clocking), one data, and one scan output. Some regular NCL registers within the test datapath were selectively replaced with scan registers for testing. Determination of insertion points was not an automated process and was required to be done manually by the test engineers. The proposed methodology was tested on various NCL circuits, and results confirmed almost 100% stuck-at fault coverage for all the test circuits with moderate area overhead.

The methodology in [56] only considered the global feedback loops within an NCL circuit and did not address the possibility of faults in the internal feedbacks of NCL gates (local feedback), which is a major limitation from testing viewpoint. In [58], Satagopan *et al.* proposed an automatic design-for-test (DFT) insertion flow (ADIF) methodology, which addressed the abovementioned limitation. To break the global feedback loops, XOR gates, controlled by an external test input, were inserted in the feedback paths, which provided additional test points to enhance controllability. A balanced XOR tree structure was used to enhance the observability of unobservable faults at the output. However, the XOR tree structure can exponentially grow for circuits with larger set of unobservable faults, eventually adding to the design cost in terms of area. [58] proposed an alternative scan-based approach to enhance observability by inserting scannable observable latches in unobservable fault nets. The scan-based approach does not require the XOR tree structure. To break the local feedback, a clocked scan D-latch was inserted in the internal feedback path of each NCL gate, and a custom NCL gate library was formed for ATPG. The clocked NCL ATPG gate library facilitates the application of synchronous scan-based approach for test pattern generation. Although the methodology yields high test coverage, the design-for-test significantly increases the area overhead, mostly due to the additional hardware in each threshold gate. [59] modifies the ADIF methodology further to address the area overhead issue. The scan D-latch is removed in the internal gate feedback path and an external input is introduced to increase the controllability and observability. The D-latch removal significantly reduces the gate area overhead. Like [58], XOR gates, controlled by the latched test input signal, were introduced to break the global feedback paths. Test points were inserted by grouping the remaining unobservable fault nets, as identified by SCOAP analysis (Sandia Controllability and Observability Program), using a XOR gate tree structure followed by a scannable-observation-latch. While the overall area utilization of the test circuits in [59] was less than that of [58], the overhead was still significant due to the addition of clocked latches and XOR trees.

All the above-mentioned methods ([56], [58], and [59]) use synchronous testing frameworks, which mandate the incorporation of additional circuitry for the synchronous-asynchronous interfacing. This interfacing comes at the cost of significant area overhead. [60] argues that reintroducing clocks in clockless designs defeats the purpose and limits the potential of asynchronous designs. This argument is backed by Cheng and Li's assessment in [61], where they have warned about the possibility of reliability issues and timing violations within an NCL system due to the inclusion of clocked hardware. To overcome these limitations, [60] proposed a completely homogenous design-for-test (DFT) methodology for NCL circuits. Instead of clocked DFT elements, the methodology modified an existing NCL read/write (R/W) circuit as DFT element [19] to insert controllability and observability test points into the previously uncontrollable and unobservable nodes. Global feedback loops were broken by replacing NCL registers in the feedback path with scan registers, where the scan registers were designed by modifying the NCL R/W circuit. The unique contribution of this work was the development of a completely asynchronous testing framework, which was not addressed before. However, the work did not consider the testing of gate internal feedback paths.

The same research group of [60] proposed an asynchronous interleaved scan architecture to facilitate on-line built-in self-test (BIST) of NCL circuits [62]. On-line implies testing the circuit during the idle phases instead of halting the operation of the circuit. The interleaved scan architecture ensures that the alternating NULL/DATA pattern of NCL circuit is maintained during on-line testing. Two scan paths were implemented by establishing a connection between the input (output) registers of each combinational block and the output (input) registers of the next (previous) combinational block. Couple of test pattern generators and output response analyzers were utilized for applying correct DATA/NULL wavefronts. [62] avoids utilizing clock trees and sync-async interfacing circuitries, which significantly reduces the area overhead (on an average) as compared to other schemes. This work later acted as a motivation for the development of a BIST scheme for SCL circuits [63].

[64] proposed the first ever self-timed ATPG for testing NCL circuits along with quiescent current testing to detect stuck-at faults in gate internal feedbacks. As discussed earlier, faults in gate internal feedback may not interfere with the circuit operation. However, it may affect the timing, which is very difficult to detect in NCL circuits as NCL circuits are indeterminate. Therefore, [64] argued in favor of current testing, since a fault in the internal feedback of gates are expected to increase the leakage current of the circuit, which, if traced, can detect a fault. The method did not add additional circuitry in individual gates for testing and minimized area overhead significantly, which was a major concern in prior methods. However, sophisticated sensing and measuring device was

TABLE 3. Different NCL testing methodologies.

Related Work	*Year*	*Testing Framework*	*Testing Considerations*				
			Global Feedback	*Gate Internal Feedback*	*Test Point Insertion*	*Sync-Async Interfacing*	*Area Overhead*
[56]	2002	Scan based synch. DFT	Considered	Not Considered	✗	Required	Moderate
[58]	2007	Non-scan and scan-based synch. DFT	Considered	Considered	✓	Required	Very High
[59]	2008	Scan based synch. DFT	Considered	Considered	✓	Required	High
[60]	2014	Asynch. DFT	Considered	Not Considered	✓	Not Required	High
[62]	2016	Asynch. Interleaved Scan for on-line BIST	Considered	Not Considered	✓	Not Required	Moderate
[64]	2018	Clockless Self-timed ATPG + current testing	Considered	Considered	✓	Not Required	Low

required to measure the current. Additionally, testing was comparatively slower as the current measurement had to wait until the quiescent current settled. A summarization of the abovementioned NCL testing schemes is listed in Table 3.

The testing methodologies discussed herein rely on extensive simulations for detecting faults and ensuring correctness. Although simulation-based testing has been predominantly used in the IC design industry, only simulation may not guarantee complete correctness. There still can be untestable bugs that escape and remain undetected. Formal verification is an alternate approach to establish design correctness, where the correctness properties are formulated as mathematical proofs. Mathematical proofs, covering a larger and comprehensive set of faults, can detect corner-case errors that generally go undetected in simulations. Nowadays, formal verification is considered as an integral component of any commercial design flow. However, not many works, which focus on formal verification of asynchronous designs, can be found in literature. There are some methods that focus on the verification of bounded-delay model of asynchronous paradigm [65], [66]. The verification schemes involve trace theory and timed petri-nets to verify the timing constraints imposed by the bounded-delay model. On the other hand, circuits based on QDI model do not require any timing analysis. Therefore, these timed verification models are not suitable for the validation of QDI circuits.

A deadlock verification scheme for QDI circuits was proposed by Verbeek *et al.* in [67]. In [67], the circuits to be verified were based on click library [68], which is significantly different from NCL. Moreover, the method could only validate the liveness (absence of deadlock) of the circuits, not safety (functional correctness). Wijayasekara *et al.* proposed a verification scheme for NCL circuits based on equivalence checking [69]. [69] modeled the synchronous specification (input to the synthesis tool) as well as the NCL asynchronous implementation (synthesized output) as transition systems (TSs), which were checked for equivalence utilizing the theory of well-founded equivalence bisimulation (WEB) refinement [70]. Both specification and implementation TSs had to satisfy the refinement properties, which was ensured by a decision procedure. The methodology checked for both safety and liveness. However, scalability was a major limitation. This is because, modeling actual QDI circuits as TSs is not ideal. QDI circuits, such as NCL and PCHB [6], are highly nondeterministic in nature, which results in extremely complicated TSs. Moreover, due to the hysteresis of individual gates, the state space increases almost exponentially for larger circuits, resulting in an infeasible verification time. This issue with scalability was also encountered by [71]. In [71], the authors developed a model-checking based approach to verify QDI combinational PCHB circuits, where the circuits were also modeled as TSs.

Sakib *et al.* [72] developed an alternate verification approach for combinational as well as sequential NCL circuits, where the circuits were not required to be modeled as TSs. This was based on the authors' previous verification works applicable for asynchronous PCHB circuits [73], [74]. To tackle the state space explosion, [72] requires the synthesized NCL circuit to undergo a structural transformation/abstraction. The actual NCL circuit is converted into its corresponding Boolean/synchronous netlist as an intermediate process of the verification flow. The converted netlist is then checked against the synchronous specification utilizing the notion of WEB refinement to ensure the safety of the circuit. The abstraction significantly improves the functional verification time. The liveness check is a separate process, which can be conducted parallelly. As a part of the liveness check, the actual (non-converted) NCL circuit is converted

into a graph structure to efficiently trace component-to-component handshaking connections. The method proves to be highly scalable and significantly faster than previous verification methods. A variant of the method was proposed by Hossain *et al.* to verify SCL circuits [75].

In case of NCL circuits, only checking for safety and liveness is not sufficient. Input-completeness and observability are also crucial and must be considered as a part of the complete verification process. While an input-incomplete and/or unobservable NCL circuit may exhibit correct functionality under normal circumstances, the circuit may malfunction under some unexpected scenarios, such as changing operating conditions caused by PVT variations, and/or environmental radiations. Few verification schemes exist with specific focus on the verification of NCL circuits' input-completeness and observability [2], [72], and [76]. [2] entails a manual procedure to check input-completeness, which is not scalable and cannot ensure input-completeness of relaxed circuits. Both [72] and [76] involve similar approach for input-completeness and observability verification. In [76], each gate and input are individually checked to verify the observability and input-completeness; whereas [72] formulates two proof obligations to simultaneously verify the observability and input-completeness of all gates and inputs.

VI. CONCLUSION AND SCOPES OF FUTURE WORK

NCL is one of the major QDI asynchronous design paradigms, which presents itself as a promising alternative to conventional synchronous circuits and has already found numerous commercial applications due to its low power, robust architecture, and ease of design reuse. This paper presents the evolution of NCL based asynchronous domain over the past two decades, primarily focusing on existing fundamental research in NCL design automation, spanning over NCL synthesis, optimization, testing, and verification.

It is evident that there has been a significant advancement in NCL design automation over the past two decades, especially in logic synthesis and optimization methodologies. NCL framework, being very similar to synchronous framework, allows automated NCL circuit design to follow similar design automation steps as synchronous circuits. As a result, many of the existing works focused on leveraging conventional synchronous CAD tools for NCL circuit synthesis. Earlier design flows built on each other and focused mostly on area optimization of the synthesized circuits, while preserving the delay-insensitivity requirements. In contrast, more recent design flows prioritized on utilizing the full potential and capacity of commercial CAD tools for technology mapping and cost-aware synthesis in terms of power, area, and delay. However, physical synthesis, i.e., synthesizing NCL circuits down to layout, is an area that remains underexplored.

Design validation of NCL circuits has also been widely investigated. Although various testing and verification schemes for NCL based asynchronous circuits have been developed over the years, there are certain limitations. Several testing methods rely on synchronous testing frameworks and require the introduction of clocked elements in the clockless design, which defeats the purpose. Moreover, the additional synchronous-asynchronous interfacing circuitry results in significant test overhead. Few promising research works focused on developing fully asynchronous DFT schemes. However, those relied on extensive simulations alone, which might not be sufficient to detect corner-case bugs. Some recent works focused on the formal modeling and verification of NCL circuits based on widely utilized verification techniques. However, scalability was a major concern in many of those methods. Couple of works addressed this issue and successfully developed unified and highly scalable verification schemes for different QDI design paradigms, including NCL. However, the completeness arguments of those verification schemes were not established formally, which may not ensure the comprehensive nature of the considered erroneous scenarios. Finally, different transistor-level implementations of NCL threshold gates are also discussed, analyzed, and compared in this paper, which can guide designers to choose the best suitable implementation depending on the application requirements.

REFERENCES

[1] D. E. Müller, "A theory of asynchronous circuits," Univ. Illinois, Graduate College, Digit. Comput. Lab., Urbana, IL, USA, Tech. Rep., 75, 1955.
[2] S. C. Smith and J. Di, *Designing Asynchronous Circuits Using NULL Convention Logic (NCL)*. San Rafael, CA, USA: Morgan & Claypool, 2009.
[3] *International Technology Roadmap for Semiconductors*. Accessed: May 2022. [Online]. Available: https://www.dropbox.com/sh/6xq737bg6pww9gq/AACQWcdHLffUeVloszVY6Bkla?dl=0&preview=2013_Design-v3.pdf
[4] *IEEE International Roadmap for Devices and Systems, Medical Devices Market Driver*. Accessed: May 2022. [Online]. Available: https://irds.ieee.org/editions/2018/medical-devices-market-driver
[5] J. T. Udding, "Classification and composition of delay-insensitive circuits," Ph.D. dissertation, Dept. Math. Comput. Sci., Eindhoven Univ. Technol., Eindhoven, The Netherlands, 1984.
[6] A. J. Martin and M. Nystrom, "Asynchronous techniques for system-on-chip design," *Proc. IEEE*, vol. 94, no. 6, pp. 1089–1120, Jun. 2006, doi: 10.1109/JPROC.2006.875789.
[7] K. M. Fant and S. A. Brandt, "NULL convention Logic™: A complete and consistent logic for asynchronous digital circuit synthesis," in *Proc. Int. Conf. Appl. Specific Syst., Archit. Processors*, Aug. 1996, pp. 261–273, doi: 10.1109/ASAP.1996.542821.
[8] S. C. Smith, "Gate and throughput optimizations for NULL convention self-timed digital circuits," Ph.D. Dissertation, School Elect. Eng. Comput. Sci., Univ. Central Florida, Orlando, FL, USA, May 2001.
[9] S. C. Smith, R. F. DeMara, J. S. Yuan, D. Ferguson, and D. Lamb, "Optimization of NULL convention self-timed circuits," *Integration*, vol. 37, no. 3, pp. 135–165, 2004.
[10] C. L. Seitz, "System timing," in *Introduction to VLSI Systems*. Reading, MA, USA: Addison-Wesley, 1980, pp. 218–262.
[11] T. S. Anantharaman, "A delay insensitive regular expression recognizer," *IEEE VLSI Tech. Bull.*, vol. 1, no. 2, pp. 3–15, 1986.
[12] J. Sparso, J. Staunstrup, and M. Dantzer-Sorensen, "Design of delay insensitive circuits using multi-ring structures," in *Proc. Eur. Design Automat. Conf.*, 1992, pp. 15–20, doi: 10.1109/EURDAC.1992.246271.
[13] N. P. Singh, "A design methodology for self-timed systems," M.S. thesis, MIT/LCS/TR258, Lab. Comput. Sci., MIT, Cambridge, MA, USA, 1981.
[14] I. David, R. Ginosar, and M. Yoeli, "An efficient implementation of Boolean functions as self-timed circuits," *IEEE Trans. Comput.*, vol. 41, no. 1, pp. 2–11, Jan. 1992, doi: 10.1109/12.123377.
[15] A. A. Sakib, "Formal modeling and verification methodologies for quasi-delay insensitive asynchronous circuits," Ph.D. dissertation, Dept. Elect. Comput. Eng., ProQuest, North Dakota State Univ., Fargo, ND, USA, 2019.

[16] L. D. Tran, T. C. Pham, O. Kavehei, and G. I. Matthews, "Asynchronous 2-phase level-encoded convention logic (LCL)," in *Proc. Int. Symp. Electr. Electron. Eng. (ISEE)*, Oct. 2019, pp. 1–6.

[17] I. E. Sutherland, "Micropipelines," *Commun. ACM*, vol. 32, no. 6, pp. 720–738, Jun. 1989.

[18] G. E. Sobelman and K. Fant, "CMOS circuit design of threshold gates with hysteresis," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, vol. 2, Mar. 1998, pp. 61–64, doi: 10.1109/ISCAS.1998.706841.

[19] K. M. Fant, *Logically Determined Design: Clockless System Design With Null Convention Logic*. Hoboken, NJ, USA: Wiley, 2005.

[20] M. Ligthart, K. Fant, R. Smith, A. Taubin, and A. Kondratyev, "Asynchronous design using commercial HDL synthesis tools," in *Proc. 6th Int. Symp. Adv. Res. Asynchronous Circuits Syst.*, Apr. 2000, pp. 114–125, doi: 10.1109/ASYNC.2000.836983.

[21] I. E. Sutherland and J. K. Lexau, "Designing fast asynchronous circuits," in *Proc. 7th Int. Symp. Asynchronous Circuits Syst. (ASYNC)*, 2001, pp. 184–193, doi: 10.1109/ASYNC.2001.914082.

[22] A. Kondratyev and K. Lwin, "Design of asynchronous circuits using synchronous CAD tools," *IEEE Design Test Comput.*, vol. 19, no. 4, pp. 107–117, Jul. 2002, doi: 10.1109/MDT.2002.1018139.

[23] Y. Zhou, D. Sokolov, and A. Yakovlev, "Cost-aware synthesis of asynchronous circuits based on partial acknowledgement," in *Proc. IEEE/ACM Int. Conf. Comput. Aided Design*, Nov. 2006, pp. 158–163.

[24] Y. Zhou, C. Shi, Z. Deng, and A. Yakovlev, "Synthesis and optimization of asynchronous dual rail encoded circuits based on partial acknowledgement," in *Proc. IEEE 12th Int. Conf. ASIC (ASICON)*, Oct. 2017, pp. 496–503, doi: 10.1109/ASICON.2017.8252522.

[25] T. Chelcea, G. Venkataramani, and S. C. Goldstein, "Area optimizations for dual-rail circuits using relative-timing analysis," in *Proc. 13th IEEE Int. Symp. Asynchronous Circuits Syst. (ASYNC)*, Mar. 2007, pp. 117–128, doi: 10.1109/ASYNC.2007.10.

[26] C. Jeong and S. M. Nowick, "Optimization of robust asynchronous circuits by local input completeness relaxation," in *Proc. Asia South Pacific Design Autom. Conf.*, Jan. 2007, pp. 622–627, doi: 10.1109/ASPDAC.2007.358055.

[27] C. Jeong and S. M. Nowick, "Block-level relaxation for timing-robust asynchronous circuits based on eager evaluation," in *Proc. 14th IEEE Int. Symp. Asynchronous Circuits Syst.*, Apr. 2008, pp. 95–104, doi: 10.1109/ASYNC.2008.25.

[28] W. B. Toms and D. A. Edwards, "A complete synthesis method for block-level relaxation in self-timed datapaths," in *Proc. 10th Int. Conf. Appl. Concurrency Syst. Design*, 2010, pp. 24–34, doi: 10.1109/ACSD.2010.29.

[29] W. Toms and D. Edwards, "Prime indicants: A synthesis method for indicating combinational logic blocks," in *Proc. 15th IEEE Symp. Asynchronous Circuits Syst.*, May 2009, pp. 139–150, doi: 10.1109/ASYNC.2009.24.

[30] R. B. Reese, S. C. Smith, and M. A. Thornton, "Uncle—An RTL approach to asynchronous design," in *Proc. IEEE 18th Int. Symp. Asynchronous Circuits Syst.*, May 2012, pp. 65–72.

[31] A. Bardsley, "Balsa: An asynchronous circuit synthesis system," M.S. thesis, Dept. Comp. Sci., Univ. Manchester, Manchester, U.K., 1998.

[32] D. Edwards, A. Bardsley, L. Janin, L. Plana, and W. Toms, "Balsa: A tutorial guide," M.S. thesis, Dept. Comp. Sci., Univ. Manchester, Manchester, U.K., 2006.

[33] L. Zhou, R. Parameswaran, F. Parsan, S. Smith, and J. Di, "Multi-threshold NULL convention logic (MTNCL): An ultra-low power asynchronous circuit design methodology," *J. Low Power Electron. Appl.*, vol. 5, no. 2, pp. 81–100, May 2015.

[34] B. Bhaskaran, "Automated synthesis and cycle reduction optimization for asynchronous NULL convention circuits using industry-standard CAD tools," Ph.D. dissertation, Dept. Comp. Eng., Univ. Missouri, Columbia, MO, USA, 2007.

[35] C. Jeong and S. M. Nowick, "Technology mapping and cell merger for asynchronous threshold networks," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 27, no. 4, pp. 659–672, Apr. 2008.

[36] F. A. Parsan, W. K. Al-assadi, and S. C. Smith, "Gate mapping automation for asynchronous NULL convention logic circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 22, no. 1, pp. 99–112, Jan. 2014.

[37] M. Moreira, A. Neutzling, M. Martins, A. Reis, R. Ribas, and N. Calazans, "Semi-custom NCL design with commercial EDA frameworks: Is it possible?" in *Proc. 20th IEEE Int. Symp. Asynchronous Circuits Syst.*, May 2014, pp. 53–60, doi: 10.1109/ASYNC.2014.15.

[38] A. Neutzling, M. G. A. Martins, R. P. Ribas, and A. I. Reis, "Synthesis of threshold logic gates to nanoelectronics," in *Proc. 26th Symp. Integr. Circuits Syst. Design (SBCCI)*, Sep. 2013, pp. 1–6, doi: 10.1109/SBCCI.2013.6644871.

[39] M. T. Moreira, C. H. M. Oliveira, R. C. Porto, and N. L. V. Calazans, "NCL+: Return-to-one null convention logic," in *Proc. IEEE 56th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2013, pp. 836–839, doi: 10.1109/MWSCAS.2013.6674779.

[40] M. T. Moreira, R. A. Guazzelli, and N. L. V. Calazans, "Return-to-one protocol for reducing static power in C-elements of QDI circuits employing m-of-n codes," in *Proc. 25th Symp. Integr. Circuits Syst. Design (SBCCI)*, Aug. 2012, pp. 1–6, doi: 10.1109/SBCCI.2012.6344444.

[41] 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 Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 6, pp. 1981–1993, Jun. 2018, doi: 10.1109/TCSI.2017.2772206.

[42] S. L. Hurst, "An introduction to threshold logic: A survey of present theory and practice," *Radio Electron. Engineer*, vol. 37, no. 6, pp. 339–351, 1969.

[43] K. Haulmark, W. Khalil, W. Bouillon, and J. Di, "Comprehensive comparison of NULL convention logic threshold gate implementations," in *Proc. New Gener. CAS (NGCAS)*, Nov. 2018, pp. 37–40, doi: 10.1109/NGCAS.2018.8572223.

[44] F. A. Parsan and S. C. Smith, "CMOS implementation comparison of NCL gates," in *Proc. IEEE 55th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2012, pp. 394–397, doi: 10.1109/MWSCAS.2012.6292040.

[45] F. A. Parsan and S. C. Smith, "CMOS implementation of static threshold gates with hysteresis: A new approach," in *Proc. IEEE/IFIP 20th Int. Conf. VLSI System-Chip (VLSI-SoC)*, Oct. 2012, pp. 41–45.

[46] S. Yancey and S. C. Smith, "A differential design for C-elements and NCL gates," in *Proc. 53rd IEEE Int. Midwest Symp. Circuits Syst.*, Aug. 2010, pp. 632–635, doi: 10.1109/MWSCAS.2010.5548905.

[47] H. J. Lee and Y.-B. Kim, "Low power null convention logic circuit design based on DCVSL," in *Proc. IEEE 56th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2013, pp. 29–32, doi: 10.1109/MWSCAS.2013.6674577.

[48] M. T. Moreira, M. Arendt, F. G. Moraes, and N. L. V. Calazans, "Static differential NCL gates: Toward low power," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 62, no. 6, pp. 563–567, Jun. 2015, doi: 10.1109/TCSII.2015.2407198.

[49] L. Heller, W. Griffin, J. Davis, and N. Thoma, "Cascode voltage switch logic: A differential CMOS logic family," in *IEEE Int. Solid-State Circuits Conf., Dig. Tech. Papers*, Feb. 1984, pp. 16–17, doi: 10.1109/ISSCC.1984.1156629.

[50] R. S. Pal, S. Sharma, and S. Dasgupta, "Recent trend of FinFET devices and its challenges: A review," in *Proc. Conf. Emerg. Devices Smart Syst. (ICEDSS)*, Mar. 2017, pp. 150–154.

[51] A. A. Sakib, A. A. Akib, and S. C. Smith, "Implementation of FinFET based static NCL threshold gates: An analysis of design choice," in *Proc. IEEE 63rd Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2020, pp. 37–40, doi: 10.1109/MWSCAS48704.2020.9184629.

[52] A. A. Sakib and S. C. Smith, "Implementation of static NCL threshold gates using emerging CNTFET technology," in *Proc. 27th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Nov. 2020, pp. 1–4, doi: 10.1109/ICECS49266.2020.9294823.

[53] Y. Bai, R. F. DeMara, J. Di, and M. Lin, "Clockless spintronic logic: A robust and ultra-low power computing paradigm," *IEEE Trans. Comput.*, vol. 67, no. 5, pp. 631–645, May 2018.

[54] M. Feigenson, J. W. Reiner, and L. Klein, "Efficient current induced domain-wall displacement in $SrRuO_3$," *Phys. Rev. Lett.*, vol. 98, Jun. 2007, Art. no. 247204.

[55] D. Fan, Y. Shim, A. Raghunathan, and K. Roy, "STT-SNN: A spin-transfer-torque based soft-limiting non-linear neuron for low-power artificial neural networks," *IEEE Trans. Nanotechnol.*, vol. 14, no. 6, pp. 1013–1023, Nov. 2015.

[56] A. Kondratyev, L. Sorensen, and A. Streich, "Testing of asynchronous designs by 'inappropriate' means. Synchronous approach," in *Proc. 8th Int. Symp. Asynchronous Circuits Syst.*, 2002, pp. 171–180, doi: 10.1109/ASYNC.2002.1000307.

[57] M. Abramovici, M. Breuer, and A. Friedman, *Digital Systems Testing and Testable Design*. New York, NY, USA: Computer Science, 1990.

[58] V. Satagopan, B. Bhaskaran, W. K. Al-Assadi, S. C. Smith, and S. Kakarla, "DFT techniques and automation for asynchronous NULL conventional logic circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 10, pp. 1155–1159, Oct. 2007.

[59] W. K. Al-Assadi and S. Kakarla, "Design for test of asynchronous NULL convention logic (NCL) circuits," in *Proc. IEEE Int. Test Conf.*, Oct. 2008, pp. 1–9.
[60] N. Nemati, M. C. Reed, and M. R. Frater, "Asynchronous test hardware for null convention logic," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Jun. 2014, pp. 1744–1747.
[61] C.-H. Cheng and J. C.-M. Li, "An asynchronous design for testability and implementation in thin-film transistor technology," *J. Electron. Test.*, vol. 27, no. 2, pp. 193–201, Apr. 2011.
[62] N. Nemati, M. C. Reed, K. Fant, and P. Beckett, "Asynchronous interleaved scan architecture for on-line built-in self-test of null convention logic," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 746–749.
[63] B. Sparkman, S. C. Smith, and J. Di, "Built-in self-test for multi-threshold NULL convention logic asynchronous circuits," in *Proc. IEEE 38th VLSI Test Symp. (VTS)*, Apr. 2020, pp. 1–6, doi: 10.1109/VTS48691.2020.9107627.
[64] N. Nemati, P. Beckett, M. C. Reed, and K. Fant, "Clock-less DFT-less test strategy for null convention logic," *IEEE Trans. Emerg. Topics Comput.*, vol. 6, no. 4, pp. 460–473, Oct. 2018.
[65] C. J. Myers, *Asynchronous Circuit Design*. New York, NY, USA: Wiley, 2001.
[66] A. Semenov and A. Yakovlev, "Verification of asynchronous circuits using time Petri net unfolding," in *Proc. 33rd Design Autom. Conf.*, Las Vegas, NV, USA, 1996, pp. 59–62.
[67] F. Verbeek and J. Schmaltz, "Verification of building blocks for asynchronous circuits," in *Proc. Int. Workshop ACL2 Theorem Prover Appl.*, vol. 114, R. Gamboa and J. Davis, Eds., 2013, pp. 70–84.
[68] A. Peeters, F. T. Beest, M. de Wit, and W. Mallon, "Click elements: An implementation style for data-driven compilation," in *Proc. IEEE Symp. Asynchronous Circuits Syst.*, May 2010, pp. 3–14.
[69] V. M. Wijayasekara, S. K. Srinivasan, and S. C. Smith, "Equivalence verification for NULL convention logic (NCL) circuits," in *Proc. IEEE 32nd Int. Conf. Comput. Design (ICCD)*, Oct. 2014, pp. 195–201.
[70] P. Manolios, "Correctness of pipelined machines," in *Formal Methods in Computer-Aided Design* (Lecture Notes in Computer Science), vol. 1954, W. A. Hunt, Jr., and S. D. Johnson, Eds., New York, NY, USA: Springer-Verlag, 2000, pp. 161–178.
[71] A. A. Sakib, S. C. Smith, and S. K. Srinivasan, "Formal modeling and verification for pre-charge half buffer gates and circuits," in *Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2017, pp. 519–522.
[72] A. A. Sakib, S. Le, S. C. Smith, and S. K. Srinivasan, "Formal verification of NCL circuits," in *Asynchronous Circuit Applications*. Edison, NJ, USA: IET, 2019, pp. 309–338.
[73] A. A. Sakib, S. C. Smith, and S. K. Srinivasan, "An equivalence verification methodology for combinational asynchronous PCHB circuits," in *Proc. IEEE 61st Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2018, pp. 767–770.
[74] A. A. Sakib, S. C. Smith, and S. K. Srinivasan, "Formal modeling and verification of PCHB asynchronous circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 27, no. 12, pp. 2911–2924, Dec. 2019.
[75] M. Hossain, A. A. Sakib, S. K. Srinivasan, and S. C. Smith, "An equivalence verification methodology for asynchronous sleep convention logic circuits," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2019, pp. 1–5, doi: 10.1109/ISCAS.2019.8702098.
[76] A. Kondratyev, L. Neukom, O. Roig, A. Taubin, and K. Fant, "Checking delay-insensitivity: 10^4 gates and beyond," in *Proc. 8th Int. Symp. Asynchronous Circuits Syst.*, 2002, pp. 149–157.

DANYLO KHODOSEVYCH received the B.S. degree in computer engineering from Florida Polytechnic University, Lakeland, FL, USA, in 2020, where he is currently pursuing the M.S. degree in computer engineering. His research interests include asynchronous digital designs, NULL convention logic (NCL) design, and optimization. The main objective of his current research work is to develop enhanced optimization methodologies for NCL circuits.

ASHIQ A. SAKIB (Member, IEEE) received the B.Tech. degree in electronics and communication engineering from the West Bengal University of Technology, Kolkata, India, in 2013, and the Ph.D. degree in computer engineering from the North Dakota State University, Fargo, ND, USA, in 2019. He is currently an Assistant Professor with the Department of Electrical and Computer Engineering, Florida Polytechnic University, Lakeland, FL, USA. His current research interests include asynchronous digital designs, which spans over asynchronous logic, circuit/architecture optimization, low-power designs, resilience, and formal verification. He is a member of IEEE-HKN and Phi Kappa Phi. He is a Handling Editor for *Journal of Circuits, Systems, and Computers*.