ASCEND: Advanced Synthesis, Circuit Exploration, and Design Optimization for NCL Circuits

Alexander Bodoh
Dept. of Electrical and Computer Engineering
Florida Polytechnic University, Lakeland, FL, USA
abodoh8714@floridapoly.edu

Ashiq A. Sakib
Dept. of Electrical and Computer Engineering
Southern Illinois University Edwardsville, IL, USA
asakib@siue.edu

***Abstract*—Null Convention Logic (NCL) is a popular asynchronous design paradigm that, in contrast to its synchronous counterpart, demonstrates robustness, ultra-low power performance, and ease of design reuse, while avoiding the complex clock management issues. Despite these advantages, the lack of mature computer-aided design tools has limited the widespread adoption of the domain across the industry. To address this gap, we present herein a novel framework, ASCEND, to facilitate the automated synthesis of NCL circuits with enhanced optimization capabilities. ASCEND includes multiple NCL protocol and equivalence verification schemes, as well as supports the synthesis of a pareto-optimal design in terms of throughput or area, contingent upon the target requirement. The performance and efficacy of the framework have been validated using multiple IWLS and ISCAS benchmark circuits.**

Keywords—Null Convention Logic, Electronic Design Automation, Logic Synthesis, Design Optimization, Asynchronous Circuits

I. INTRODUCTION

The conventional synchronous domain has been dominating the digital IC design industry for years, owing to its compliance with the growing consumer demands for miniaturized, reliable, and fast-operating electronic devices. The clocked domain has, however, reached a point where the potential for further advancements is significantly restricted by a variety of design constraints, with clock management being the foremost concern. Several clock-related issues, such as clock skew, clock distribution, and clock jitter, arise that require substantial mitigation, making the design significantly expensive in terms of area and power consumption. For instance, addressing clock skew demands complex timing analysis, which becomes extremely challenging due to fabrication and manufacturing variations. In addition, the significantly large capacitance associated with the clock line makes it exceedingly power hungry. Large buffer trees are employed to minimize capacitance in distribution, which further results in an increase in power as well as area overhead.

In recent times, the Quasi-Delay Insensitive (QDI) asynchronous clockless domain has emerged as a promising alternative, overcoming the major challenges associated with the synchronous domain [1]. Null Convention Logic (NCL) [2] is one such commercially successful QDI paradigms. NCL circuits are robust, energy efficient, provides ease of design reuse, and demonstrates less electromagnetic interference. The inherent robustness and resilience against process, voltage, and temperature (PVT) variations render them an excellent choice for a variety of distinctive applications, such as energy-harvesting- and/or self-powered IoT applications, low-power and low-maintenance applications, radiation intensive adverse environmental applications, among others. Despite the fact that NCL circuits have garnered a growing industrial interest, their widespread adoption has been mostly impeded by the following: 1) the lack of mature electronic design automation (EDA) tools to facilitate automated synthesis, optimization, mapping, testing, and verification, 2) a lack of human resources with the requisite design expertise, and 3) increased area footprint due to architectural constraints [3]. The synchronous-like framework of NCL has motivated researchers to utilize similar design flows and processes to automate the synthesis of such circuits. NCL_D [4] and NCL_X [5] are two earlier commercial NCL design flows, which use standard synchronous computer-aided design (CAD) tools for NCL logic synthesis. The NCL_D flow is more restrictive due to the imposed delay-insensitivity (DI) constraints, which restrict the potential for optimization, resulting in a significantly larger area overhead. Conversely, the NCL_X flow enables a more relaxed implementation of the logic unit, thereby improving the circuit's overall delay and area utilization. However, this is accomplished at the cost of a larger and slower completion detection unit. Numerous NCL optimization schemes were developed in the subsequent years, both at the gate and module levels, with the primary focus to reduce area by striking an optimal balance between restrictive and non-restrictive implementations [6-8]. UNCLE [9] is a more recent EDA tool that integrates several optimization schemes, including some of the ones previously referenced, into the flow to synthesize optimized NCL circuits.

Herein, we propose a novel EDA framework, Advanced Synthesis, Circuit ExploratioN, and Design Optimization for NCL (ASCEND), to facilitate the automated synthesis of NCL circuits with enhanced optimization capabilities. Following are the major contributions and unique features of ASCEND:

i) integrates the first ever enhanced optimization scheme, which is primarily focused on the synthesis of throughput-optimized NCL circuits within a unified design flow;

ii) intelligently combines and builds upon several existing NCL optimization schemes, yielding smaller and faster circuits;

iii) incorporates multiple functions to enhance the optimization flexibility by allowing for less restrictive and independent optimization of logic and delay insensitivity;

iv) empowers the user to make a well-informed decision regarding the synthesis of a pareto-optimal design in terms of throughput or area, contingent upon the target requirement;

v) includes multiple optional testing and debugging features

This work was supported by the National Science Foundation (NSF) under Grant CCF-2153373.

to validate the functional correctness, observability, and input-completeness of the synthesized NCL circuits.

The remainder of the paper is organized as follows: Section II presents a brief NCL overview; Section III details the ASCEND design flow; Section IV demonstrates the tool and presents the synthesis results; and Section V concludes the paper and discusses the scope of future work.

II. NCL Overview

A. NCL Framework

Fig. 1 depicts the NCL framework. Each stage within an NCL pipeline consists of two sets of registers, one combinational logic (C/L) unit, and one completion detection (C/D) unit. NCL mostly employs a dual-rail one-hot (*1-of-2*) data encoding scheme, where each dual-rail variable can have one of the three valid values: DATA0 ($\{rail^1, rail^0\} = 01$), DATA1 ($\{rail^1, rail^0\} = 10$), and NULL ($\{rail^1, rail^0\} = 00$), which correspond to Boolean logic '0', Boolean logic '1', and a spacer value that indicates the unavailability of DATA, respectively. In a properly functioning circuit, the two rails of a given bit must never be asserted simultaneously, i.e., $\{rail^1, rail^0\} \neq 11$. A quad-rail logic (*1-of-4*) is less prevalent, but it is also feasible. NCL primarily employs a four-phased handshaking protocol to regulate and synchronize the flow of input wavefronts. In each stage, the registration and the C/D unit utilize local request (*Ki*) and acknowledgment (*Ko*) signals to maintain an alternating sequence of NULL and DATA. The NULL spacer helps in distinguishing two distinct DATA wavefronts in the absence of a global clock reference. NCL gates, frequently referred to as *threshold* gates, are utilized to implement NCL registers, C/L unit, and C/D unit. A set of 27 fundamental threshold gates collectively implements all unique functions consisting of up to four variables. A threshold gate can be unweighted or weighted. An unweighted gate is represented as *THmn*, where *m* and *n* indicate the gate's threshold and the number of inputs, respectively, where $1 \leq m \leq n$. The threshold determines the minimum number of inputs required to be asserted to assert the gate output. A weighted gate is represented as $THmnWw_1 \ldots w_N$, where w_N is the integer weight value of input *N*. NCL gates have state holding capability, referred to as *hysteresis*, which ensures that that once a gate is asserted, it will not be deasserted until all inputs are deasserted. This is a key feature to preserve the delay insensitivity within the pipeline.

B. Input Completeness, Observability, and Relaxation

The NCL C/L unit must be *input-complete* and *observable* to preserve delay-insensitivity [1]. In an input-complete NCL C/L unit, the circuit's outputs are allowed to transition from NULL-to-DATA (DATA-to-NULL) only after all its inputs have transitioned from NULL-to-DATA (DATA-to-NULL). Seitz's "weak conditions" for delay-insensitive signaling impose a less stringent requirement [10], allowing certain outputs in a multi-output C/L circuit to transition to DATA (NULL) without requiring a complete DATA (NULL) set at the inputs. This is permissible as long as it is not possible to compute a complete DATA (NULL) set at the output before the arrival of a complete input set. Observability requires each gate transition to be detectable at the output, i.e., any gate that undergoes a transition must also cause a transition in at least one primary output. Note that an input-complete NCL circuit ensures the observability of each primary input. Nowick introduced the concept of *relaxation* in [6], illustrating that it is feasible for an NCL C/L to replace certain input-complete functions with their equivalent *relaxed* (input-incomplete) functions and still remain input-complete and observable as a whole. Figs. 2a and 2b show the NCL implementation of an input-complete (strict) and input-incomplete (relaxed) 4-input NCL AND function, respectively. While the strict implementation of the AND function requires 93 transistors and two levels of gate delays, the relaxed implementation requires 30 transistors and one level of gate delay. This indicates the potential of relaxation to significantly reduce the delay and area overhead in NCL circuits.

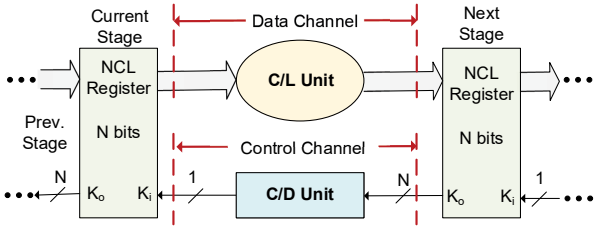

Fig. 1. NCL Framework [3].

III. ASCEND Design Flow

The ASCEND design flow is outlined in Fig. 3. The input to the tool is a Hardware Description Language (HDL) implementation of the specification synchronous/Boolean circuit. The specification first goes through an initial RTL synthesis process to generate an optimized netlist, *Bool34_opt*, consisting of up to 4-input fundamental Boolean functions only (i.e., OR, NOR, AND, NAND, XOR, XNOR, NOT), targeting an NCL gate library. The ABC tool [11] is utilized to perform the mapping and optimization in this stage. The tool then diverges into two paths, one prioritizes the synthesis of a throughput optimized NCL circuit, while the other prioritizes area optimization, as illustrated next.

A. Area and Throughput Optimized NCL Synthesis

a) ASCEND Enhanced Throughput Optimization: In our novel ASCEND Enhanced Throughput optimization, the algorithm leverages multiple optimization strategies, namely intelligent gate merging, explicit completeness, and iterative relaxation, yielding a faster NCL circuit than any of the existing methods. Integrating this throughput optimization scheme into the design flow is a major contribution, which advances ASCEND compared to previous design flows, and has been internally developed and extensively validated, as detailed in [12]. This is a three-stage process, as summarized below:

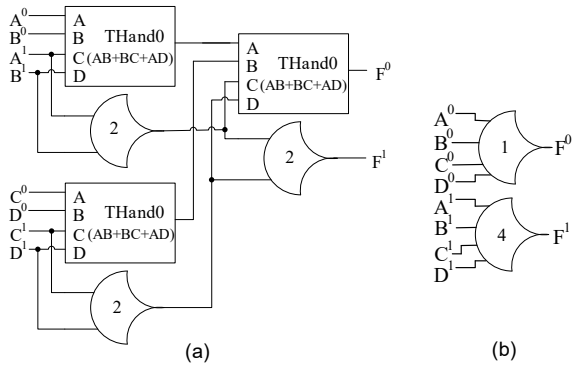

Fig. 2. a) Strict 4-input NCL AND; b) relaxed 4-input NCL AND function.

i. The first stage is referred to as the *iterative relaxation* stage, during which a modified version of Nowick's relaxation algorithm, targeting latency optimization, is applied iteratively. All the functions in *Bool34_opt* are considered as *strict* (i.e., input-complete) and the circuit's critical paths are identified prior to applying the iterative relaxation algorithm. The objective of the algorithm is to reduce the critical path gate delay by relaxing as many 3 and 4-input functions in the critical path(s) as possible during each iteration. The *strict* implementation of any 3 or 4-input function in NCL, with the exception of XOR and XNOR, necessitates a gate delay 1-level longer than its equivalent *relaxed* implementation. A categorical weight assignment scheme is implemented to prioritize relaxation of critical path functions with 3- and 4-inputs. The optimal set of functions that can be relaxed in the identified critical path(s) is then determined using a unate covering problem algorithm. Subsequently, the critical paths are recalculated, and the steps are iterated upon until no further relaxation is possible.

ii. The second stage is referred to as the *intelligent merging* stage, during which the partially relaxed circuit from stage 1 is further optimized for throughput, utilizing our novel iterative merging algorithm. The algorithm initially identifies clusters of Boolean functions in the critical path(s) that may be merged, then subsequently evaluates the tradeoffs associated with these possible merges, taking into account several constraints particular to NCL. The merged functions are relaxed to further reduce the delay in critical path(s), which can compromise the circuit's input-completeness and observability. To prevent this, signals that are not observable through any path in the C/L circuit are explicitly made observable through the C/D unit by generating completion signals from each unobservable function. Since this synthesis step prioritizes throughput optimization, a merging that necessitates the generation of explicit signals is only permissible if it reduces the critical path delay or does not increase the completion unit's delay due to the added signals. After each merging, the critical path(s) are recalculated, and the procedure is repeated until there are no mergeable functions.

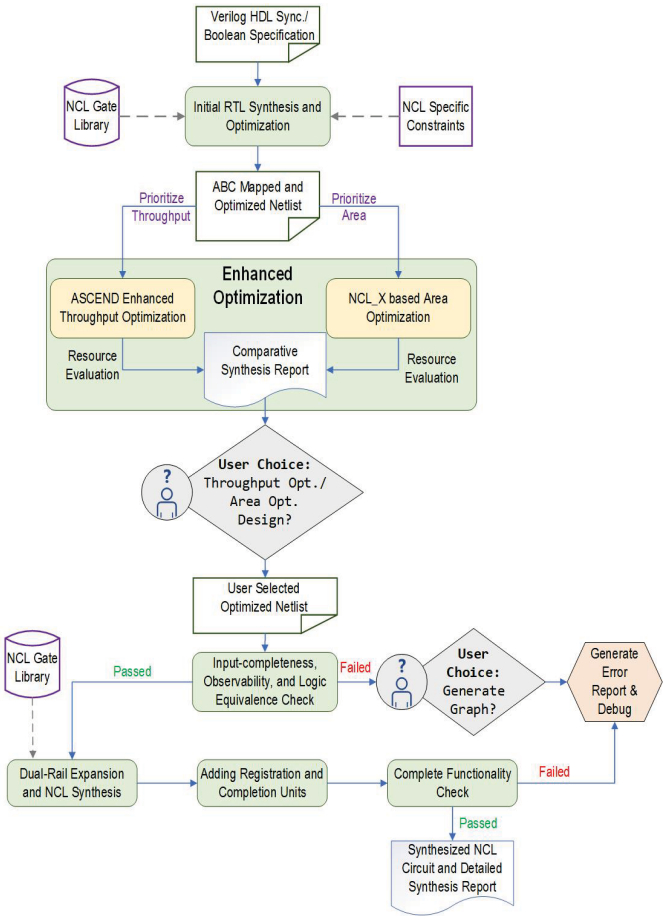

Fig. 3. ASCEND Design Flow

iii. This third stage is the *area optimization* stage. While the first two stages target relaxation and merging on the critical path functions for throughput optimization, this stage applies the techniques to merge and relax as many remaining functions as possible from non-critical paths, to optimize for area.

b) NCL_X based Area Optimization: This is the alternate synthesis path that targets the synthesis of area-optimized NCL circuits. Kondratyev's NCL with explicit completeness architecture, NCL_X [5], is integrated herein within the design flow. In this phase of synthesis, all the functions in *Bool34_opt* are implemented in a relaxed manner, causing the C/L to be neither input-complete nor observable. This relaxed implementation significantly reduces the area and delay as compared to regular NCL implementations. However, this cannot outperform the ASCEND throughput optimized synthesis in terms of delay, as can be seen from Table I. This is because the overall delay of an NCL circuit is determined by both the C/L unit as well as the C/D circuitry. In NCL_X, a large number of explicit signals are required to be generated in each stage, which are combined by an additional C/D unit per stage, to make the C/L functions observable and the overall circuit input-complete. This significantly increases the completion delay of the synthesized circuit, which in turn increases the overall circuit delay.

B. Evaluation and Verification of the Synhtesized Logic with Graphical Debugging Features

Post logic synthesis, the resource evaluation metrics, such as the number of input-complete gates, number of relaxed gates, overall critical path delay, total transistor counts, etc., of both optimizations are presented to the user for a well-informed selection based on the target application and requirements. Additionally, throughout the entire logic synthesis phase, the initial specification circuit undergoes a significant transformation. This may result in the implementation (i.e., synthesized circuit) appearing significantly different from its specification counterpart. The tool conducts a logic equivalence check at this point, which is based on extensive simulation, to guarantee that the specification and the implementation are logically equivalent. The user is informed of the verification's success or failure via text transcripts. This also enables the designers to detect any corner case bugs within the processes. In addition, the synthesized circuit comprises a variety of functions, some of which are *strict* and others that are *relaxed*. A bug in the synthesis process can impact circuit's input-completeness or observability property, hence compromising its delay-insensitivity. To detect such bugs, the tool includes an automated verification scheme that validates the input-completeness and observability of the synthesized NCL circuits. In addition, the tool can generate a graphical image, highlighting

the input-complete gates, relaxed gates, gate-to-gate connections, critical paths from primary inputs to primary outputs, etc., for debugging, in case the verification fails.

C. Dual-Rail Expansion and Inclusion of Registration and Completion Units

A dual-rail expansion is performed post logic synthesis, where each wire is converted to a dual-rail signal and each gate function gets replaced by the corresponding NCL implementation using threshold gates. The registers and the completion detection circuitry are added at this point. The complete NCL circuit is presented as a Verilog HDL model. A final functional check is performed to verify the synthesized NCL implementation's functionality. The check performs tests to guarantee the correct propagation of NULL and DATA through the stages, the generation of correct request and acknowledgment signals by the stages during NULL/DATA propagation, and the generation of non-illegal dual-rail signals, among others.

IV. Tool Demonstration and Synthesis Results

ICSAS'85 [13] and IWLS [14] combinational benchmark circuits were utilized to demonstrate the performance of the proposed ASCEND tool. The throughput and area optimized NCL versions of all the benchmark circuits were effectively synthesized by the tool. Due to page constraints, Table I presents the synthesis results of 6 of the 64 benchmark circuits of varying size and complexity. The *NCL Funcs.* column shows the total number of NCL gate functions in the synthesized circuit, with the percentage of relaxed functions calculated as the value within the parenthesis. The *Total Delay* column represents the combined worst case number of gate delays in the circuit's C/L and C/D logic, i.e., *Total Delay = C/L delay+ C/D delay*. The speed of an NCL circuit is estimated in terms of gate delays as *2 × Total Delay* [12]. The *#Transistors* column indicates the number of transistors required to implement each NCL circuit's completion and combinational logic. Table I suggests that the ASCEND enhanced throughput optimization can produce NCL circuits that are ~8% faster than NCL_X on average, with *term1* exhibiting 13.3% faster operation. NCL_X based area optimized synthesis requires 152, 2,856, 1,185, and 6,653 fewer transistors for *term1*, *frg2*, *c5315*, and *too_large* circuits, respectively, as compared to the throughput optimized synthesis. However, it is interesting to note that NCL_X based synthesis requires 1,526 and 3,880 more transistors for *c1355* and *c7552*, respectively.

TABLE I. Synthesis Results

Circuit	ASCEND Enhanced Throughput Optimization			NCL_X based Area Optimization		
	NCL Funcs.	Total Delay	#Transistors	NCL Funcs.	Total Delay	#Transistors
term1	250 (39%)	13	7060	192	15	6908
c1355	170 (89%)	13	6282	170	14	7808
frg2	905 (34%)	14	27534	681	15	24678
c5315	1170 (44%)	25	34421	876	27	33236
c7552	1121 (48%)	26	34300	992	28	38180
too_large	2667 (60%)	15	81973	2059	16	75320

This suggests that the integration of our novel ASCEND enhanced throughput optimization scheme within the synthesis flow, enables our tool to produce NCL circuits that are always faster and even smaller in certain instances. This underscores the efficacy of our tool, which generates a comprehensive synthesis report that includes all evaluation metrics for both optimizations, enabling users to make more informed tradeoffs.

V. Conclusions

In this paper, we have presented and demonstrated ASCEND, a novel EDA framework, to facilitate the automated synthesis of NCL circuits with multi-faceted enhanced optimization capabilities. The tool offers outstanding prospects for future research to advance development, assisting current researchers in their endeavors to create industry-standard support tools for optimized NCL asynchronous circuits. Currently, work is in progress in the following areas: i) integrating transistor level SPICE simulation data for NCL primitive cells to precisely evaluate the delay and energy utilization; ii) replacing simulation-based testing with NCL formal verification schemes; and iii) developing an RTL Netlist viewer for the post-optimization synthesized NCL circuits.

Acknowledgment

This work was supported by the National Science Foundation (NSF) under Grant CCF-2153373.

References

[1] S. C. Smith and J. Di, Designing Asynchronous Circuits using NULL Convention Logic (NCL). San Rafael, CA, USA: Morgan & Claypool, 2009.

[2] 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 (ASAP)*, 1996, pp. 261–273.

[3] D. Khodosevych and A. A. Sakib, "Evolution of NULL convention logic based asynchronous paradigm: An overview and outlook," in *IEEE Access*, vol. 10, pp. 78650–78666, 2022.

[4] M. Ligthart, K. Fant, R. Smith, A. Taubin, and A. Kondratyev, "Asynchronous design using commercial HDL synthesis tools," *in ASYNC*, 2000, pp. 114–125.

[5] A. Kondratyev and K. Lwin, "Design of asynchronous circuits using synchronous CAD tools," in *IEEE Des. Test Comput*., vol. 19, no. 4, pp. 107–117, Jul. 2002.

[6] 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.

[7] C. Jeong and S. M. Nowick, "Block-level relaxation for timing-robust asynchronous circuits based on eager evaluation," in *ASYNC.*, Apr. 2008, pp. 95–104.

[8] W. Toms and D. Edwards, "Prime indicants: A synthesis method for indicating combinational logic blocks," in *ASYNC*., 2009, pp. 139–150.

[9] R. B. Reese, S. C. Smith, and M. A. Thornton, "Uncle—An RTL approach to asynchronous design," in *ASYNC.*, May 2012, pp. 65-72.

[10] C. L. Seitz, "System timing," in *Intro. to VLSI Systems.* Reading, MA, USA: Addison-Wesley, 1980, pp. 218–262.

[11] R. Brayton and A. Mishchenko, "ABC: An academic industrial-strength verification tool," in *Proc. 22nd Int. Conf. Comput. Aided Verification*, Edinburgh, U.K., 2010, pp. 24–40.

[12] D. Khodosevych, A. C. Bodoh, A. A. Sakib and S. C. Smith, "Combining relaxation with NCL_X for enhanced optimization of asynchronous Null convention logic circuits," in *IEEE Access*, vol. 11, pp. 104688-104699, 2023.

[13] ISCAS-85 High Level Models. Accessed: May, 2024. [Online]. Available:http://web.eecs.umich.edu/~jhayes/iscas.restore/benchmark.html.

[14] S. Yang, "Logic synthesis and optimization benchmarks user guide: Version 3.0," *Microelectron. Center North Carolina (MCNC)*, NC, USA, Tech. Rep. 1991-IWLS-UG-Saeyang, 1991.