Formal Verification of Completion-Completeness for NCL Circuits

Son N. Le
Electrical and Computer Engineering
North Dakota State University
Fargo, USA
son.ngoc.le@ndsu.edu

Sudarshan K. Srinivasan
Electrical and Computer Engineering
North Dakota State University
Fargo, USA
sudarshan.srinivasan@ndsu.edu

Scott C. Smith
Electrical Engineering and Computer Science
Texas A&M University Kingsville
Kingsville, USA
scott.smith@tamuk.edu

***Abstract*—Ensuring completion-completeness is required for delay-insensitivity when utilizing bit-wise completion to pipeline NCL circuits comprised of input-incomplete logic functions. Hence, this work presents an automated formal method to detect NCL circuits that are not completion-complete.**

Keywords— asynchronous circuits, formal verification, formal methods, equivalence checking, NULL Convention Logic

I. Introduction

NULL Convention Logic (NCL) [1] is a Quasi-Delay Insensitive (QDI) asynchronous design paradigm that has been shown to be able to operate correctly in more extreme environments than its synchronous counterpart [2]. NCL circuits do not utilize a clock signal for synchronization; instead NCL utilizes multi-rail logic, such as dual-rail, along with a 4-phase handshaking protocol to achieve delay-insensitivity. A dual-rail signal, D, consists of two wires, D^0 and D^1, which may assume any value from the set {DATA0 (i.e., 0b01), DATA1 (i.e., 0b10), and NULL (i.e., 0b00)}. The DATA0 state ($D^0 = 1$ and $D^1 = 0$) corresponds to a Boolean logic 0, the DATA1 state ($D^0 = 0$ and $D^1 = 1$) corresponds to a Boolean logic 1, and the NULL state ($D^0 = 0$ and $D^1 = 0$) corresponds to the empty set meaning that the value of D is not yet available. The two rails are mutually exclusive, such that both rails can never be asserted simultaneously; this state is defined as an ILLEGAL state. NCL systems contain at least two DI registers, one at both the input and at the output. Two adjacent register stages interact through their request and acknowledge handshaking signals, K_i and K_o, respectively, to prevent the current DATA wavefront from overwriting the previous DATA wavefront, by ensuring that the two DATA wavefronts are always separated by a NULL wavefront. An asserted handshaking signal represents request for DATA (rfd), while it being deasserted represents request for NULL (rfn). Handshaking can be performed using either full-word or bit-wise completion [3]. Full-word completion requires that the acknowledge signals from each bit in register$_i$ be conjoined together by the completion component, whose single-bit output is connected to all request lines of register$_{i-1}$. On the other hand, bit-wise completion only sends the completion signal from bit b in register$_i$ back to the bits in register$_{i-1}$ that took part in the calculation of bit b. This method may potentially require fewer logic levels than that of full-word completion, thus increasing throughput.

In order to achieve delay-insensitivity, NCL circuits must be input-complete and observable [4]. Input-completeness requires that all outputs of a combinational circuit may not transition from NULL to DATA until all inputs have transitioned from NULL to DATA, and that all outputs of a combinational circuit may not transition from DATA to NULL until all inputs have transitioned from DATA to NULL. In circuits with multiple outputs, it is acceptable according to Seitz's "weak conditions" of delay-insensitive signaling [5], for some of the outputs to transition without having a complete input set present, as long as all outputs cannot transition before all inputs arrive. Observability requires that no orphans may propagate through a gate, where an orphan is defined as a wire that transitions during the current DATA wavefront, but is not used in the determination of the output. NCL circuits that utilize the bit-wise completion strategy along with input-incomplete logic functions/components must also be completion-complete [6] to ensure delay-insensitivity. Completion-completeness requires that completion signals only be generated such that no two adjacent DATA wavefronts can interact within any combinational logic (C/L) component. Note that completion-completeness is inherent when using full-word completion.

While [7] presents automated formal methods for ensuring that NCL circuits utilize correct handshaking, and are input-complete and observable, this paper describes an automated formal method to ensure that NCL circuits are also completion-complete.

II. Previous Work

The need for completion-completeness was demonstrated in [6] by showing a number of example NCL circuits that utilized proper handshaking connections and were input-complete and observable, but still were not delay-insensitive, since they allowed two adjacent DATA wavefronts to interact within a C/L component. Take for example the partial product generation circuit for X(1:0) × Y(1:0) utilizing bit-wise completion, as shown in Fig. 1. AND functions Y(1) • X(1) and Y(0) • X(0) are input-complete, as shown in Fig. 2(a), while the other two AND functions are input-incomplete, as shown in Fig. 2(b), such that the entire circuit is input-complete (i.e., all outputs cannot become DATA until all inputs are DATA). To show that this circuit is not completion-complete, and therefore not delay-insensitive, let X_i and Y_i be 00_2 and 11_2, respectively, which would result in $PP_i = 0000_2$; and let $X_{i+1}(0)$ = DATA1 and $Y_{i+1}(1)$ = DATA0, which would result in $PP_{i+1}(1)$ = DATA0, where the subscript, i, refers to the

XXX-X-XXXX-XXXX-X/XX/$XX.00 ©20XX IEEE

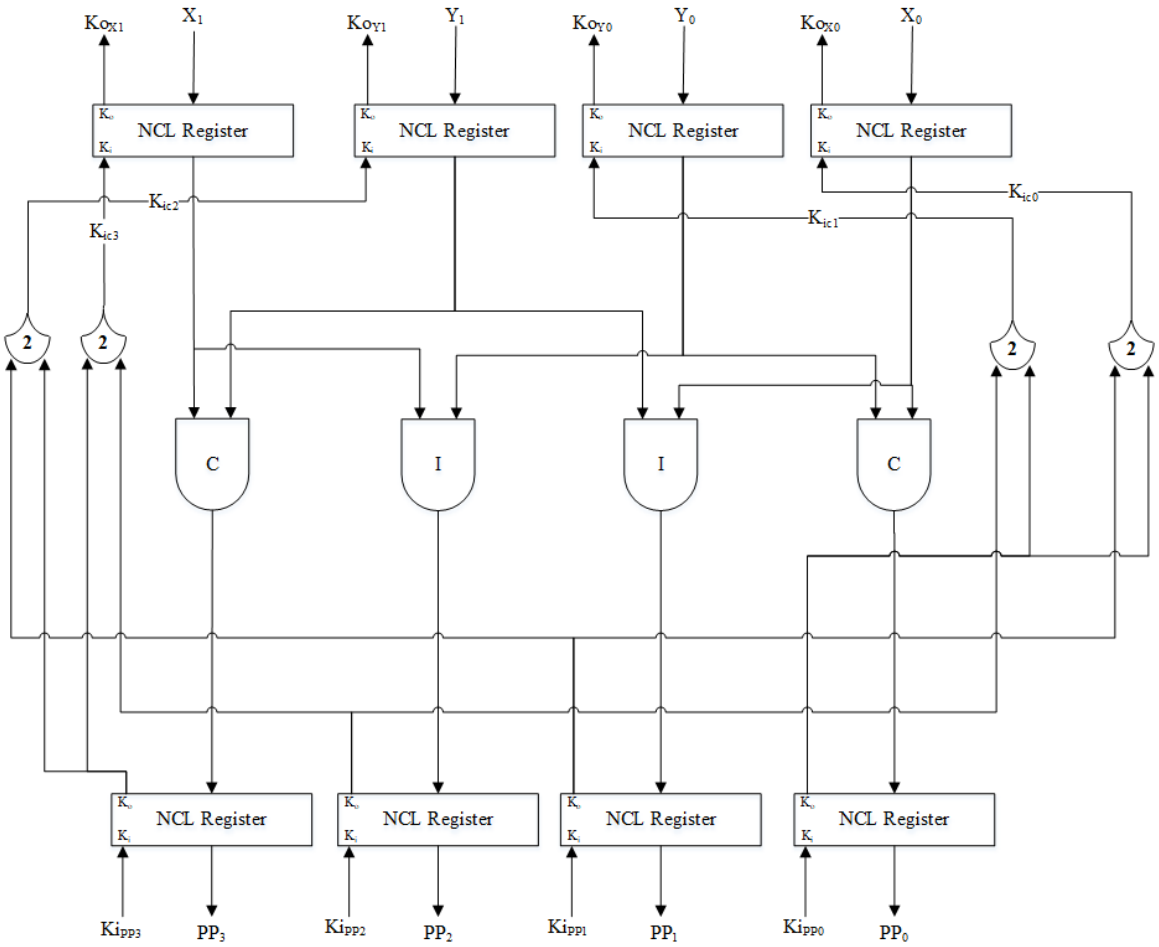

Fig. 1. Completion-Incomplete NCL Circuit

wavefront. Now, assume that the signals transition as follows, starting from the NULL state (i.e., all dual-rail signals are NULL and all K_i and K_o signals are rfd): X_i changes to DATA (i.e., 00_2), $Y_i(0)$ changes to DATA (i.e., DATA1), and $Y_i(1)$ remains NULL. This causes $PP_i(2:0)$ to become 000_2, as expected, while $PP_i(3)$ remains NULL, which in turn causes K_{ic0} and K_{ic1} to become rfn, allowing NULL to flow through these two input registers. This in turn causes $PPi(1:0)$ to become NULL, assuming their respective K_i signals are rfn, which transitions K_{ic0} to rfd, allowing $X_{i+1}(0)$ = DATA1 to flow through its register. $Y_i(1)$ now finally transitions to DATA1, which causes two adjacent DATA wavefronts, $X_{i+1}(0)$ and $Y_j(1)$, to interact within the combinational logic, which violates the completion-completeness criterion; and this produces $PP_{i+1}(1)$ = DATA1, which is incorrect.

In addition to showing how to manually determine if an NCL circuit is completion-complete, [6] also presented a variety of methods to make NCL circuits completion-complete, so that they would be delay-insensitive. For this example, either the two input-incomplete AND functions could be replaced with input-complete versions, or the completion logic sets would need to be modified. The work herein presents an automated method to formally verify that an NCL circuit is completion-complete.

III. Completion-Completeness Verification

The proposed completion-completeness verification is as follows. The NCL circuit is partitioned into stages, and each stage is handled independently. If a stage has p inputs, then p proof obligations are required for that stage. Below, we describe the generic proof obligation (PO) template that must be applied to each input of a circuit stage. The approach described using this template can be applied to any arbitrary NCL circuit. The POs are formulated such that they can be automatically checked using a Satisfiability Modulo Theories (SMT) solver [8].

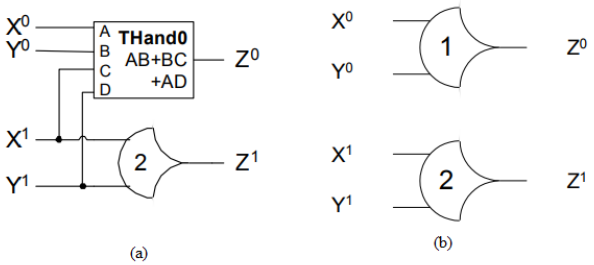

Fig. 2. (a) Input-Complete and (b) Input-Incomplete NCL AND Components

Completion-Completeness Proof Obligation: Without loss of generality, an NCL circuit stage is assumed to have m threshold gates, p dual-rail inputs, and q dual-rail outputs, and include a p-bit input register and q-bit output register, as shown in Fig. 1 for p=q=4. To formulate the proof, three separate symbolic steps of the NCL circuit are required, denoted as Steps A, B, and C, respectively. ${g_{A/B/C/D}}^1, \dots, {g_{A/B/C/D}}^m$, are Boolean variables that represent the current state of the threshold gates for the corresponding symbolic step. ${i_{A/B/C}}^1, \dots, {i_{A/B/C}}^p$, are the symbolic values applied to the circuit inputs for the corresponding symbolic step; and ${i_{A/B/C}}^k$ represents the circuit input that is being verified. ${o_{A/B/C/D}}^1, \dots, {o_{A/B/C/D}}^q$, are the output values acquired during the corresponding step of the circuit with current state and input values mentioned above. ${Ko_{A/B/C/D}}^1, \dots, {Ko_{A/B/C/D}}^p$ and ${Kic_{A/B/C/D}}^1, \dots, {Kic_{A/B/C/D}}^p$ represent the K_o outputs and K_i inputs, respectively, of the NCL input register for the corresponding step, as labeled in Fig. 1. ${Ki_{A/B/C}}^1, \dots, {Ki_{A/B/C}}^q$ corresponds to the K_i inputs to the output register for the respective steps. Note that K_o, K_{ic}, and o are all threshold gate outputs, and are therefore accounted for in variable g.

The predicates used to construct the completion-completeness PO are shown in Table I. p_0 indicates that all threshold gate output values are 0 for Step A, which indicates that the circuit is in the NULL state before a DATA transition. p_1 indicates that all circuit inputs during Step A are NULL. p_2 indicates that all circuit K_i inputs are 1, which indicates that the circuit is in a rfd state. p_3 symbolically steps the circuit stage under test using dual-rail inputs $({i_A}^1, \dots, {i_A}^p)$, K_i inputs $({Ki_A}^1, \dots, {Ki_A}^q)$ and threshold gate values $({g_A}^1, \dots, {g_A}^m)$, and stores the gate output values to $({g_B}^1, \dots, {g_B}^m)$. p_4 indicates that the circuit stage input being tested for completion-completeness, ${i_B}^k$, remains NULL, and all other inputs are DATA for Step B. p_5 symbolically steps the circuit stage under test using the Step B inputs, and stores the gate output values to $({g_C}^1, \dots, {g_C}^m)$. p_6 indicates that all circuit inputs during Step C are set to NULL. p_7 is used to constrain the K_i inputs for Step C, such that if a particular circuit output is DATA, then its corresponding K_i input is constrained to 0, indicating rfn; and if the circuit output is NULL, then its corresponding K_i input is constrained to 1, indicating rfd. p_8 symbolically steps the circuit stage under test using the Step C inputs, and stores the gate output values to $({g_D}^1, \dots, {g_D}^m)$. p_9 checks the K_{ic} and K_o values of the input register to ensure that they are correct for input i^k being constrained to NULL (i.e., K_{ic} and K_o for input register$_k$ should both be 1, since i^k never transitioned from NULL; and the rest of the input register bits' K_{ic} and K_o should not both be 1, as this would allow the subsequent DATA wavefront to pass through into the C/L, thus violating the completion-completeness criteria). The completion-completeness proof obligation is constructed as follows:

$$\{p_0 \wedge p_1 \wedge p_2 \wedge p_3 \wedge p_4 \wedge p_5 \wedge p_6 \wedge p_7 \wedge p_8\} \rightarrow p_9$$

At a high level, these predicates restrict the input under test so that it stays NULL and is therefore requesting DATA for all three symbolic steps (A, B, and C). The other inputs are not constrained, and can transition from NULL to DATA and back to NULL, as allowed by their respective handshaking signals. If the circuit is completion-complete, then the unconstrained inputs can transition from DATA to NULL, but not back to DATA before the constrained input transitions to DATA and then to NULL. An unconstrained input that could transition back to DATA indicates that the circuit is not completion-complete. Essentially, p_9 indicates that none of the unconstrained inputs could transition back to DATA, which is what the SMT solver is checking. This can be observed when looking back to the Fig. 1 example in Section II. The property is violated when the constrained input, *Y(1)* is tested, as both K_{ic0} and K_{oX0} are 1 after NCLStep C, such that $X_{i+1}(0)$ would be allowed to pass through into the C/L. If the solver is able to prove the PO, then this indicates that the circuit is completion-complete w.r.t. input k. If however, there is a violation, then the solver will provide a counter example to the proof obligation, which can then be used to trace the source of the completion-completeness violation.

TABLE I. COMPLETION-COMPELTENESS PREDICATES

p_n	PREDICATE
p_0	$\bigwedge_{n=1}^{n=m} ({g_A}^n = 0)$
p_1	$\bigwedge_{n=1}^{n=p} ({i_A}^n = 0b00)$
p_2	$\bigwedge_{n=1}^{n=q} ({Ki_A}^n = {Ki_B}^n = 1)$
p_3	$({g_B}^1, \dots, {g_B}^m) = NCLStep({i_A}^1, \dots, {i_A}^p, {g_A}^1, \dots, {g_A}^m, {Ki_A}^1, \dots, {Ki_A}^q)$
p_4	$\bigwedge_{n=1}^{n=p} ({i_B}^n = \begin{cases} 0b00, n = k \\ (0b01 \vee 0b10), n \neq k \end{cases})$
p_5	$({g_C}^1, \dots, {g_C}^m) = NCLStep({i_B}^1, \dots, {i_B}^p, {g_B}^1, \dots, {g_B}^m, {Ki_B}^1, \dots, {Ki_B}^q)$
p_6	$\bigwedge_{n=1}^{n=p} ({i_C}^n = 0b00)$
p_7	$\bigwedge_{n=1}^{n=q} ({Ki_C}^n = \begin{cases} 0, & {o_B}^n = (0b01 \vee 0b10) \\ 1, & {o_B}^n = 0b00 \end{cases})$
p_8	$({g_D}^1, \dots, {g_D}^m) = NCLStep({i_C}^1, \dots, {i_C}^p, {g_C}^1, \dots, {g_C}^m, {Ki_C}^1, \dots, {Ki_C}^q)$
p_9	$\bigwedge_{n=1}^{n=p} \begin{cases} ({Ko_D}^n = Kic_C = 1), n = k \\ \sim({Ko_D}^n \bar{\wedge} Kic_C), n \neq k \end{cases}$

TABLE II. COMPLETION-COMPLETENESS VERIFICATION BENCHMARKS

N	COMPLETION-COMPLETE RUN TIME (SEC)	COMPLETION-INCOMPLETE RUN TIME (SEC)
4	2.785	.186
8	2.785	.186
12	7.131	.315
16	15.315	.525
20	30.135	.797
24	49.909	1.142
28	78.887	1.628
32	115.078	2.092
36	169.248	2.745
40	229.685	3.278
44	314.026	3.967
48	429.539	4.776
52	564.778	5.739
56	709.051	6.835
60	1042.379	8.469
64	1170.315	10.883

IV. RESULTS

For the verification results presented herein, partial-product generation of N-bit × N-bit unsigned dual-rail NCL multipliers were used as benchmarks, where $4 \leq N \leq 64$. The verification proof obligations were checked using the Z3 SMT solver [9] running on an Intel® Core™ i5-6600k CPU with 16GB of RAM, operating at 3.50 GHz; however, any SMT solver could be used. The results are listed in Table II, where the first column is *N*, corresponding to an N-bit × N-bit dual-rail NCL unsigned multiplier partial product generation circuit. The second column is the verification time in seconds of completion-complete multipliers that are constructed using only input-complete AND2 components, shown in Fig. 2(a). The third column is the verification time in seconds of completion-incomplete multipliers, where the input-complete AND2 components are replaced with their input-incomplete version, shown in Fig. 2(b), for partial products $X_i Y_j$, where $i \neq j$. These are used to test the time to detect circuits that are input-complete but not completion-complete. The time reported for each completion-complete circuit is the total time to verify that all inputs are completion-complete; and the time reported for each completion-incomplete circuit is the total time until one input is found to be completion-incomplete. Z3 detected all completion-incomplete circuits and provided a counter example.

In addition to the multiplier partial product generation circuits, the other two circuits described in [6] where tested as well: a) the final stage of an unsigned multiplier with completion-complete and completion-incomplete GEN_S7 components, and b) the six 2-input AND function circuit. The developed automated completion-completeness verification method correctly verified the completion-complete versions and flagged the completion-incomplete circuits.

V. CONCLUSIONS AND FUTURE WORK

This paper presents the first automated methodology for formal verification of completion-completeness of NCL circuits. The results are very promising, as even a 64 × 64 multiplier partial product generation circuit could be fully verified in 19.5 minutes. The limitation to verification using the computer described above was not verification timeout of more than one day, but storage limitation as circuit size grew. Techniques to further improve efficiency and scalability could be explored as future work.

Acknowledgement: This paper is based upon work supported by the National Science Foundation under Grant No. CCF-1717420.

REFERENCES

[1] K. M. Fant and S. A. Brandt, "NULL Convention Logic: A Complete and Consistent Logic for Asynchronous Digital Circuit Synthesis," *International Conference on Application Specific Systems, Architectures, and Processors*, pp. 261-273, August 1996.

[2] J. Di and S. C. Smith, "Asynchronous Digital Circuits," in *Extreme Environment Electronics*, pp. 663 – 673, CRC Press, November 2012.

[3] S. C. Smith, R. F. DeMara, J. S. Yuan, M. Hagedorn, and D. Ferguson, "Delay-Insensitive Gate-Level Pipelining," *Elsevier's Integration, the VLSI Journal*, Vol. 30/2, pp. 103-131, October 2001.

[4] S. C. Smith and J. Di, "Designing Asynchronous Circuits using NULL Convention Logic (NCL)," *Synthesis Lectures on Digital Circuits and Systems*, Morgan & Claypool Publishers, Vol. 4/1, July 2009.

[5] C. L. Seitz, "System Timing," in *Introduction to VLSI Systems*, Addison-Wesley, pp. 218-262, 1980.

[6] S. C. Smith, "Completion-Completeness for NULL Convention Digital Circuits Utilizing the Bit-wise Completion Strategy," *International Conference on VLSI*, pp. 143-149, June 2003.

[7] A. A. Sakib, S. Le, S. C. Smith, and S. K. Srinivasan, "Formal Verification of NCL Circuits," in *Asynchronous Circuit Applications*, pp. 309-338, IET, December 2019.

[8] D. Monniaux, "A survey of Satisfiability Modulo Theory" [online]. Available: https://hal.archives-ouvertes.fr/hal-01332051/document [Accessed April 25, 2020].

[9] L. M. de Moura and N. Bjørner, "Z3: An efficient SMT solver," *in TACAS*, ser. Lecture Notes in Computer Science, C. R. Ramakrishnan and J. Rehof, Eds., vol. 4963, Springer, 2008, pp. 337–340.