Automated verification of input completeness for NCL circuits

S. Le✉, S.K. Srinivasan and S.C. Smith

An automated formal verification approach for ensuring input completeness of NULL Convention Logic (NCL) circuits is proposed. NCL circuits have the benefit that they can operate in extreme environments where traditional synchronous circuits fail due to significant fluctuations in circuit timing. Input completeness is a critical property to ensure correct functioning of NCL circuits in extreme environments and therefore is required to be verified. Note that an NCL circuit can be functionally correct and still not be input complete, which could cause the circuit to operate correctly under normal conditions, but malfunction only when the circuit timing is substantially changed (e.g. operating in a very hot or cold environment such as outer space).

Introduction: NULL Convention Logic (NCL) circuits [1] are a type of quasi-delay insensitive asynchronous design style that has been demonstrated to function in environments characterised by high radiation exposure, and high or low temperatures or large temperature fluctuations, where synchronous circuit counterparts fail [2]. The ability of NCL circuits to function correctly in extreme environments makes them very suitable for space exploration, the power industry, the automobile industry (internal combustion engines), oil/gas exploration, medical imaging instrumentation, the laser industry, superconducting computing and energy storage systems, and low voltage or low power applications such as wireless sensor networks or Internet of Things.

Synchronisation of NCL circuits happens via the propagation of NULL and DATA waves through the circuit, utilising handshaking instead of a traditional clock signal. Dual-rail signals are used for data representation. A NULL state (absence of data) is represented by 0b00 and a DATA state is represented as either 0b01 (0 in Boolean) or 0b10 (1 in Boolean). The state 0b11 is an ILLEGAL state. To achieve delay insensitivity, all NCL circuits must satisfy two properties, input completeness and observability. In order for a combinational NCL circuit to be input complete, its outputs may not all transition from NULL to DATA until all inputs have transitioned from NULL to DATA, and conversely, may not all transition from DATA to NULL until all inputs have transitioned from DATA to NULL [3]. Note that some outputs can transition to DATA (NULL) before all inputs are DATA (NULL) as long as all outputs cannot become DATA (NULL) until all inputs are DATA (NULL) [4]. Observability ensures that every gate asserted during a DATA wavefront propagates through the circuit to cause at least one circuit output to be asserted. In this Letter, an automated formal verification approach to check input completeness of NCL circuits is proposed. The efficiency of the proposed approach is demonstrated using 37 NCL circuit benchmarks.

Related work: A manual approach to checking input completeness is outlined in [5]. To check a circuit for input completeness, an analysis has to be done on each output term. For example, in order for output Z to be input complete with respect to input A, every product term in all rails of Z (in SOP format) must contain any rail of A. This ensures that Z cannot be DATA until A is DATA, and if Z is constructed solely out of NCL gates with hysteresis, the gate hysteresis ensures that Z cannot transition from DATA to NULL until A transitions from DATA to NULL. Hence, Z is input complete with respect to A. However, this method cannot ensure input completeness of relaxed NCL circuits [6], where not all gates contain hysteresis. Also, scalability is a problem with this approach, as the number of product terms that need to be verified grows exponentially as the number of inputs increase.

Kondratyev *et al.* [7] provide a formal verification approach for observability verification, which entails determining all input combinations that assert $gate_i$, then forcing $gate_i$ to remain de-asserted while checking that none of those input combinations result in all circuit outputs becoming DATA. This check is performed for all gates to ensure circuit observability; and if also applied to each circuit input (i.e. replace $gate_i$ with $input_i$ in the observability check explanation), will guarentee input completeness. In contrast, our approach directly verifies input completeness for all circuit inputs using only two proof obligations (POs).

There have also been several formal verification approaches to check safety and liveness of NCL circuits [8, 9]; however, safety and liveness verification does not guarantee input completeness, which has to be verified independently.

Input-completeness verification: The proposed approach for input-completeness verification is as follows. Two POs have been formulated, one for NULL to DATA transition and one for DATA to NULL transition. The POs are generic and can be applied to any NCL combinational circuit, and can be automatically checked using a decision procedure such as a satisfiability modulo theories (SMT) solver. The PO for the NULL to DATA transition is described next.

NULL to DATA proof obligation: Without loss of generality, an NCL circuit is assumed to have m threshold gates, p dual-rail inputs, and q dual-rail outputs. $g_A^1 \dots g_A^m$ are Boolean variables that represent the current state of the threshold gates. $i^1 \dots i^p$ are the symbolic values applied to the circuit inputs. $o^1 \dots o^q$ are the circuit output values obtained using a symbolic step of the circuit by applying the current state and inputs given above. The predicates used in the NULL to DATA PO are as follows. p_0: $\wedge_{n=1}^{n=p} \neg(i^n = 0b11)$ represents that none of the dual-rail inputs are illegal. p_1: $\wedge_{n=1}^{n=m} (g_A^n = 0)$ indicates that all threshold gates have an initial value of 0, which represents the NULL state of the circuit before a DATA transition. p_2: $\vee_{n=1}^{n=p} (i^n = 0b00)$ represents that at least one input is NULL; and p_3: $\vee_{n=1}^{n=q} (o^n = 0b00)$ represents that at least one output is NULL. The PO for NULL to DATA input completeness is given below, and states that if the inputs are all legal, the threshold gates are all initialised to 0, and one or more of the inputs are NULL, then after stepping the circuit, one or more of the outputs must be NULL. Since the input values are symbolic, when the above PO is checked using a decision procedure, it verifies the property for all possible input combinations that satisfy the PO hypothesis

$$\{\, p_0 \;\wedge\; p_1 \;\wedge\; p_2 \,\} \rightarrow p_3$$

DATA to NULL proof obligation: For the DATA to NULL PO, all possible valid combinations of threshold gate values that the circuit can have after transitioning to DATA need to first be computed. To do this, the circuit is first symbolically stepped by initialising all threshold gates to 0 and applying valid (not ILLEGAL) DATA inputs (identified as step A), which corresponds to a NULL to DATA transition. The values of the gates after step A will correspond to all possible valid combinations of the gates before a DATA to NULL transition. Next, the circuit is again symbolically stepped (called step B) to correspond to the DATA to NULL transition by using the values of the gates at the end of step A. Step B is used to verify input completeness for the DATA to NULL transition. For the PO, let $g_A^1 \dots g_A^m$ be the gate values for step A, $g_B^1 \dots g_B^m$ be the gate values for step B, $i_A^1 \dots i_A^p$ be the step A inputs, $i_B^1 \dots i_B^p$ be the inputs for step B, and $o_B^1 \dots o_B^q$ be the outputs for step B. *NCLStep* represents a step of the circuit with the specified inputs.

The predicates required for the PO are as follows. p_4: $\wedge_{n=1}^{n=p} ((i_A^n = 0b01) \vee (i_A^n = 0b10))$ represents that step A inputs correspond to valid DATA. p_5: $\wedge_{n=1}^{n=m} (g_A^n = 0)$ indicates that the gates are initialised to 0 for step A. p_6:$(g_B^1, \dots, g_B^m) = NCLStep(i_A^1, \dots, i_A^p)$ assigns $g_B^1 \dots g_B^m$, the values of the gates at the end of step A. p_7: $\wedge_{n=1}^{n=p} ((i_B^n = i_A^n) \vee (i_B^n = 0b00))$ indicates that each input for step B can either have transitioned to NULL or retained its value from step A. p_8: $\vee_{n=1}^{n=p} (i_B^n = i_A^n)$ indicates that at least one of the step B inputs has retained its previous DATA value. p_9: $\vee_{n=1}^{n=q} ((o_B^n = 0b01) \vee (o_B^n = 0b10))$ indicates that at least one of the step B outputs is still DATA (i.e. not NULL). The PO itself is given below

$$\{p_4 \;\wedge\; p_5 \;\wedge\; p_6 \;\wedge\; p_7 \;\wedge\; p_8 \,\} \rightarrow p_9$$

Results: Verification of the two POs can be performed using an SMT solver. The benchmarks used for verification were $N \times N$ unsigned dual-rail NCL multipliers ranging from 3 to 20 bits, as well as the ISCAS-85 C432 27-channel interrupt controller [10]. To perform verification, both the circuit and the POs needed to be encoded in the SMT-LIB [11] language. This was performed automatically using a developed tool that took as input the netlist of the circuit and generated both the circuit model and PO specifications in SMT-LIB format. PO checking was performed using the z3 SMT solver [12]. Verification experiments were run on a 3.5 GHz Intel Core i5 6600k processor with 16 GB of DDR4 RAM.

Table 1: Z3 runtime for benchmark NCL unsigned multipliers and NCL ISCAS circuit (*all times listed in seconds*)

Circuit	N to D	Buggy N to D	D to N	Buggy D to N
umult3	0.024	0.062	0.030	0.033
umult4	0.030	0.031	0.048	0.039
umult5	0.056	0.037	0.116	0.061
umult6	0.161	0.092	0.289	0.240
umult7	0.414	0.246	1.209	1.009
umult8	1.398	0.651	5.877	1.561
umult9	4.591	2.409	18.238	7.266
umult10	18.705	3.734	103.419	22.172
umult11	63.169	21.529	363.063	223.627
umult12	223.371	9.714	1620.489	6.572
umult13	801.975	21.590	7014.603	84.911
umult14	4796.622	26.271	30,935.954	43.472
umult15	10,974.461	417.004	TO	802.518
umult16	76,300.605	534.320	TO	45.577
umult17	TO	160.857	TO	204.847
umult18	TO	3566.523	TO	21.659
umult19	TO	4930.595	TO	70.909
umult20	TO	5381.748	TO	108.503
r-umult3	0.024	0.024	0.030	0.029
r-umult4	0.030	0.031	0.047	0.037
r-umult5	0.048	0.054	0.104	0.047
r-umult6	0.135	0.127	0.427	0.057
r-umult7	0.399	0.433	1.539	0.064
r-umult8	1.209	1.530	6.830	0.097
r-umult9	4.928	4.926	27.827	0.119
r-umult10	15.636	16.050	118.184	0.111
r-umult11	58.690	3.967	963.396	0.153
r-umult12	205.409	188.072	2996.760	0.213
r-umult13	761.903	777.390	16,969.252	0.183
r-umult14	2591.337	77.150	77,112.377	0.233
r-umult15	12,592.790	9637.624	TO	0.308
r-umult16	41,717.063	29,484.895	TO	0.811
r-umult17	TO	2922.783	TO	1.120
r-umult18	TO	12,453.725	TO	1.656
r-umult19	TO	TO	TO	1.642
r-umult20	TO	TO	TO	2.117
ISCAS-85	0.062	0.068	0.074	0.070

The verification results for the benchmark circuits are shown in Table 1. umultN represents an $N \times N$ NCL unsigned multiplier comprised completely of NCL gates. r-umultN represents a relaxed $N \times N$ NCL unsigned multiplier that contain Boolean gates when hysteresis is not needed. The first column in the table gives the circuit name. The second and third columns give the verification times for correct and buggy versions of the circuit for the NULL to DATA PO, respectively. The fourth and fifth columns give the verification times for correct and buggy versions of the circuit for the DATA to NULL PO, respectively. TO (timed out) denotes that the verification simulation time was greater than one day. The multipliers were designed using input-complete AND functions to generate the X_iY_i partial products and input-incomplete AND functions to generate the rest of the partial products (i.e. X_iY_j; $i! = j$) [13]. Buggy versions of the non-relaxed multipliers (i.e. comprised solely of NCL gates with hysteresis) were obtained by selecting a dual-rail-input at random and replacing its corresponding partial product generating input-complete AND function with the input-incomplete version. The other multiplier components, half-adders (HAs) and full-adders (FAs) are inherently input complete (i.e. all outputs cannot be determined without all inputs) and therefore cannot be made to be input incomplete when designed solely using NCL gates with hysteresis. Relaxed NCL multipliers were designed by replacing the TH22 gate within the input-incomplete AND functions and HAs with Boolean AND gates. Buggy versions of the relaxed multipliers were obtained by replacing one of the following gates with its Boolean version: the THand0 gate within an X_iY_i partial product generating AND function, a TH24comp gate within a HA, or a TH34w2 or TH23 gate within a FA. The ISCAS C432 circuit was designed by utilising as many input-incomplete functions as possible while still maintaining input completeness. Its buggy version was obtained by replacing the input-complete three-input NAND component used to calculate RC, with its input-incomplete version. For all bugs, z3 produces a counterexample that can be used to trace the bug.

Conclusion: An approach to automated verification of input completeness for NCL circuits is presented. It ensures input completeness of combinational NCL circuits comprised solely of gates with hysteresis, as well as relaxed NCL circuits that contain some gates without hysteresis; whereas the previous manual approach for ensuring input completeness [5] is not applicable to relaxed NCL circuits. It also ensures input completeness for all inputs simultaneously, whereas [7] must check each input separately. The proposed approach is efficient; however, scalability can be improved and will be a topic for future work.

Acknowledgment: This paper is based on work supported by the National Science Foundation under grant no. CCF-1717420.

Submitted: *28 June 2018* E-first: *30 August 2018*
doi: 10.1049/el.2018.6068
One or more of the Figures in this Letter are available in colour online.

S. Le, S.K. Srinivasan and S.C. Smith (*Electrical and Computer Engineering, North Dakota State University, Fargo, ND 58105, USA*)

✉ E-mail: son.ngoc.le@ndus.edu

References

1 Fant, K.M., and Brandt, S.A.: 'Null convention logictm: a complete and consistent logic for asynchronous digital circuit synthesis'. Proc. of Int. Conf. on Application Specific Systems, Architectures and Processors, 1996. ASAP 96, Chicago, IL, USA, August 1996, pp. 261–273
2 Di, J., and Smith, S.C.: 'Asynchronous digital circuits' in Mead, C.A., and Conway, L.A. (Eds.), 'Extreme environment electronics' (CRC Press, 2012), pp. 663–673
3 Smith, S.C., DeMara, R.F., Yuan, J.S., *et al.*: 'Optimization of NULL convention self-timed circuits', *Integr. VLSI J.*, 2004, **37**, (3), pp. 135–165. Available at http://dx.doi.org/10.1016/j.vlsi.2003.12.004
4 Seitz, C.L.: 'System timing', 'Introduction to VLSI systems' (Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1979), pp. 218–262
5 Smith, S.C., and Di, J.: 'Designing asynchronous circuits using null convention logic (NCL)', *Synth. Lect. Digit. Circuits Syst.*, 2009, **4**, (1), pp. 1–96. Available at https://doi.org/10.2200/S00202ED1V01Y200907DCS023
6 Jeong, C., and Nowick, S.M.: 'Optimization of robust asynchronous circuits by local input completeness relaxation'. 2007 Asia and South Pacific Design Automation Conf., Yokohama, Japan, January 2007, pp. 622–627
7 Kondratyev, A., Neukom, L., Roig, O., *et al.*: 'Checking delay-insensitivity: 104 gates and beyond'. Proc. Eighth Int. Symp. on Asynchronous Circuits and Systems, Manchester, UK, April 2002, pp. 149–157
8 Wijayasekara, V.M., Srinivasan, S.K., and Smith, S.C.: 'Equivalence verification for null convention logic (NCL) circuits'. 2014 IEEE 32nd Int. Conf. on Computer Design (ICCD), Seoul, South Korea, October 2014, pp. 195–201
9 Wijayasekara, V.M., Rollie, A.T., Hodges, R.G., *et al.*: 'Abstraction techniques to improve scalability of equivalence verification for NCL circuits', *Electron. Lett.*, 2016, **52**, (19), pp. 1594–1596
10 'ISCAS-85 c432 27-channel interrupt controller'. Available at http://web.eecs.umich.edu/~jhayes/iscas.restore/c432.html
11 Barrett, C., Fontaine, P., and Tinelli, C.: 'The SMT-LIB standard: version 2.6'. Tech. Rep., Department of Computer Science, The University of Iowa, 2017. Available at www.SMT-LIB.org
12 De Moura, L., and Bjørner, N.: 'Z3: an efficient SMT solver'. International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Budapest, Hungary, 29 March–6 April 2008, pp. 337–340. Available at http://dl.acm.org/citation.cfm?id=1792734.1792766
13 Smith, S.C., DeMara, R.F., Yuan, J.S., *et al.*: 'Delay-insensitive gate-level pipelining', *Integr. VLSI J.*, 2001, **30**, (2), pp. 103–131