Hardware Trojan Design and Detection in Asynchronous NCL Circuits

Kushal K. Ponugoti
Electrical & Computer Engineering
North Dakota State University
Fargo, ND, USA
kushalkumar.ponugoti@ndsu.edu

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

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

***Abstract*—Hardware Trojans that degrade performance, change functionality, leak information, or halt service, have been problematic in synchronous circuits. Asynchronous circuits are gaining popularity due to their lower power consumption, and better immunity against PVT (process, voltage, temperature) variations. In this paper, we show some designs to realize hardware Trojans in NULL Convention Logic (NCL), which is a Quasi-Delay Insensitive (QDI) asynchronous digital circuit paradigm. We also present a formal verification methodology to detect such Trojans, and validate our technique on an RSA decryption circuit.**

***Index Terms*—Hardware Trojans, Asynchronous Circuits, NULL Convention Logic (NCL), Formal Verification**

I. Introduction

Use of third-party Intellectual Property (IP) in modern electronic design has become a serious security issue, as the IP may be compromised by inserted hardware Trojans, which are a malicious and intentional change in the circuit design to alter functionality. The taxonomy of hardware Trojans, benchmark circuits, and detection methodologies have been extensively described in [1]–[4] for synchronous circuits, where a global clock controls the flow of data. However, distributing a single clock signal to the entire circuit is becoming problematic due to clock skew, large area requirements, and excess power dissipation for the clock tree and associated clock drivers [5].

Asynchronous design is a viable alternative, where the global clock is replaced by local handshaking signals. NULL Convention Logic (NCL) [6] is a popular Quasi-Delay Insensitive (QDI) asynchronous design paradigm that has been utilized in a number of commercial applications. 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, DATA1, NULL. 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, and will not occur in a properly operating circuit. Hence, when NCL circuits are verified and tested, illegal inputs are not typically considered.

The major contributions of this paper are: 1) we show that it is possible to design NCL hardware Trojans that are only triggered by inputting illegal values, which leads to a new class of Trojans that can impact QDI paradigms that employ dual-rail or other multi-rail encodings; and 2) we propose a formal verification method to detect these illegal Trojans, which is tested on an RSA decryption circuit.

The rest of the paper is organized as follows. Background on hardware Trojans and NCL circuits is described in Section II. Section III discusses related work on hardware Trojans in asynchronous circuits, and detection methods for such Trojans. Section IV describes a few examples of how illegal state NCL Trojans can be designed. Section V provides a formal verification technique to detect illegal state NCL Trojans. Verification results are presented in Section VI; and Section VII discusses conclusions and future work.

II. Background

A. Hardware Trojans

A hardware Trojan typically consists of a trigger circuit and a payload circuit. The Trojan can be designed to be triggered by external input combinations, rare internal states, or a combination of both. When the Trojan trigger circuit is activated, it invokes the payload circuit that can be designed to degrade system performance, leak security sensitive data, change intended functionality, or halt system operation. Much effort goes into the design of the Trojan so that it is not detected during testing. Therefore, formal verification methods need to be specifically designed to target Trojan detection.

B. NCL Background

As mentioned earlier, NCL uses multi-rail encoding, such as dual-rail. NCL circuits are implemented using 27 basic gates with hysteresis, such that after the gate output is asserted, it is only de-asserted after all inputs are de-asserted [5]. 24 of the NCL gates are threshold gates, denoted as THmnWw_1w_2, …, w_R, where n is the number of inputs; m is the gate's threshold; w_1w_2, …, w_R, each > 1 and $\leq m$, are optional integer weights of $input_1$, $input_2$, …, $input_R$, respectively; and the gate output is asserted whenever the sum of the asserted inputs meets or exceeds the gate's threshold value. Each of the n inputs is connected to the rounded portion of the gate; the output

978-1-7281-6044-3/20/$31.00 ©2020 IEEE

emanates from the pointed end of the gate; and the gate's threshold value, m, is written inside of the gate. An input with weight W_R is depicted by connecting that input to the gate W_R times. NCL systems consists of QDI registers at both the input and output, and may include additional internal registers. Two adjacent register stages interact through handshaking to ensure that two adjacent DATA wavefronts are always separated by a NULL wavefront, to prevent the subsequent DATA wavefront from overwriting the previous DATA wavefront until that has been consumed.

NCL circuits must satisfy two properties: Input-completeness and Observability [5]. 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, some of the outputs may transition without having a complete input set present, as long as all outputs cannot transition before all inputs are available. Take for example the NCL AND function in Fig. 1. The version in Fig. 1a is not input-complete, as output Z will transition to DATA0 if either input is DATA0, even if the other input is NULL. However, the version in Fig. 1b is input-complete, as Z can only transition to DATA when both inputs are DATA.

Observability requires that every gate that transitions is necessary to transition at least one output. Every gate in the circuit must be observable in order for the circuit to be QDI, as an unobservable gate may cause the circuit to malfunction. Fig. 2a shows an unobservable XOR function, since when both X and Y are DATA0, the TH12 gate is asserted, but does not take part in asserting the output, Z = DATA0. The XOR function in Fig. 2b is observable because the assertion of either internal gate guarantees that the respective output rail is asserted, since the weight of that input, 2, is the same as the output gate threshold. Note that the Fig. 2b XOR function is not optimal, as it can be implemented using only 2 gates [5].

Ensuring input-completeness and observability of the NULL to DATA transition guarantees input-completeness and observability of the DATA to NULL transition when considering NCL circuits comprised solely of NCL gates with hysteresis; however, for relaxed NCL circuits that also include Boolean gates, input-completeness and observability of the DATA to NULL transition must also be checked [7].

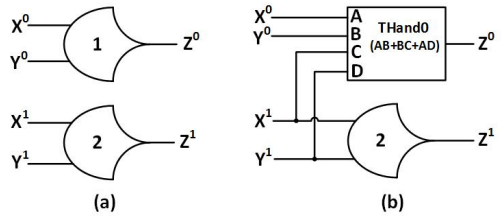

Fig. 1: NCL AND function: (a) Input-Incomplete (b) Input-complete [5]

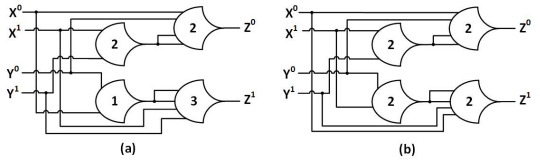

Fig. 2: NCL XOR function: (a) Unobservable (b) Observable [5]

III. Related Work

Few studies that attempt to design and detect hardware Trojans in asynchronous circuits have been reported in the literature. Inaba et al. [8] demonstrate hardware Trojans in bounded-delay MOUSETRAP-based asynchronous circuits, and utilize deep learning techniques to detect them. Four types of hardware Trojans were inserted in asynchronous FIFO-buffers by Hasan et al. [9], where the system is comprised of a memory array and a flip-flop interface between read and write controllers, each working on a different clock frequency. However, neither of these works are applicable to QDI circuits. To the best of our knowledge, this paper is the first attempt to realize hardware Trojans in QDI circuits by exploiting the illegal state.

IV. NCL Illegal Trojan Design

In this section, we show how hardware Trojans can be realized in NCL circuits by utilizing the illegal state to leak information. We first show how an illegal value can flow through an NCL circuit, using adders as example. Then, a few configurations of information leaking Trojans triggered by a propagated illegal value are presented.

Consider the input-complete and observable NCL half-adder circuit shown in Fig. 3a. The dual-rail sum output is given by $S^0 = X^0Y^0 + X^1Y^1$ and $S^1 = X^0Y^1 + X^1Y^0$. When X and Y are valid DATA values, the sum, S, and carry, C_{out}, outputs are valid DATA values. However, when either X or Y, or both, are illegal, S becomes illegal, and C_{out} may or may not be illegal. For example, when Y is DATA0 and X is illegal, C_{out} is DATA0 and S is illegal. Similar behavior can be observed in the full adder circuit shown in Fig. 3b, where S becomes illegal when any or all inputs are illegal. Processors and data encryption/decryption circuits typically

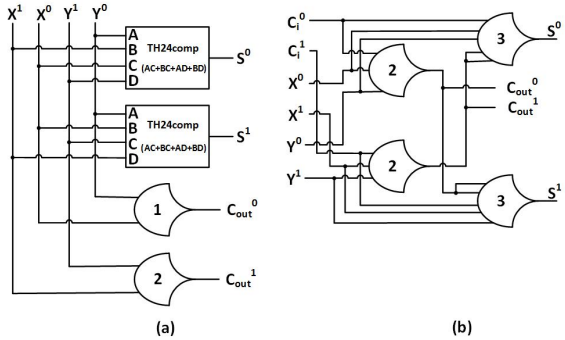

Fig. 3: (a) NCL half-adder, (b) NCL full-adder [5]

utilize multipliers, which consist of cascaded half and full adders. Security critical information moves through various circuit components, and an attacker can utilize propagated illegal values to leak information, as described next.

The circuits in Fig. 4 show some example NCL Trojans that can be employed to leak data. All three circuits utilize Boolean multiplexers to pass the normal non-secret information, B, to output, B_p, under the regular operating scenario, but pass secret information, K, only when the trigger, S, becomes illegal. For Fig. 4a, the TH22 gate, which could also be a Boolean AND gate, is never asserted when S is NULL or DATA, such that B_p always equals B. However, when S is illegal, B_p becomes secret value K instead of B, such that it can be leaked. Similar functionality can be observed in Fig. 4b by replacing the TH22 gate with a NAND gate, and swapping MUX inputs B and K. In this case, the NAND gate is always asserted when S is NULL or DATA, such that B_p always equals B. However, when S is illegal, B_p equals K instead. Another configuration is shown in Fig. 4c, where the gate responsible for generating the select signal for the multiplexers has been replaced by additional multiplexers. The behavior of this circuit is similar to the other two circuits. Like Fig. 4a and 4b, B_p always equals B, unless S is illegal, in which case B_p equals K.

V. Detection Methodology

As mentioned in Section II, NCL circuits must be verified for input-completeness and observability to ensure correct operation [7]. Luckily, these properties implicitly add a layer of security that makes it more difficult to design NCL hardware Trojans compared to synchronous circuits. Input-completeness verification does not detect any of the Trojan circuits shown in Fig. 4, as all three circuits act the same as what they replace (i.e., 2 wires that pass the 2 rails of B), as long as S in not illegal. However, observability verification flags the circuits shown in Figs. 4b and 4c as unobservable. For Fig. 4b, the NAND gate output is always 1 when S is DATA or NULL. Hence, when the overall circuit inputs are all DATA, the NAND gate is asserted and the overall circuit outputs all become DATA. Next, when all inputs change back to NULL, all circuit outputs transition back to NULL, but the NAND gate remains asserted; therefore, the NAND gate is flagged as unobservable. For Fig. 4c, the bottom left 2 MUXes

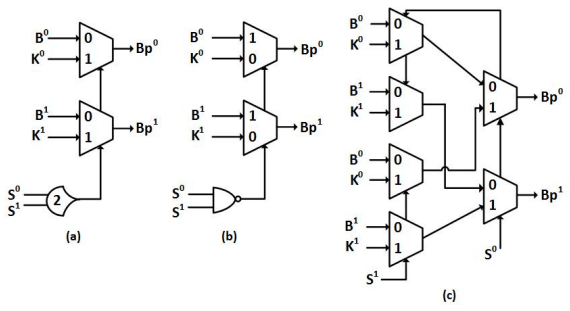

Fig. 4: NCL Trojans (a) using TH22 or AND gate, (b) using NAND gate, (c) using 6 multiplexers

controlled by S^1 are unobservable. For example, assume that S and B are both DATA1, in which case B_p is DATA1, as passed through the top left MUX pair controlled by S^0. In this case, the bottom left MUX is also asserted, but is not needed to assert B_p; hence, this MUX will be flagged as unobservable. On the other hand, the TH22 gate (or AND gate) in Fig. 4a, can never be asserted when S is valid DATA or NULL, such that observability verification skips this gate, and the circuit passes the observability check in [7]. Therefore, an additional verification method is needed to detect this Trojan, as discussed below.

If a gate can never be asserted when all circuit inputs are DATA, but can be asserted if one or more inputs are illegal, this gate should be flagged as a potential hardware Trojan. Below are the proof obligations (POs) used to detect this scenario, which must both be checked for each gate in the circuit, similar to how [7] checks each gate for observability.

Without loss of generality, assume an NCL circuit has n dual-rail inputs, $i_1, i_2, \ldots, i_n$, and k gates, with each gate output represented as $g_1, g_2, \ldots, g_k$. PO1 states that there is some combination of NULL and DATA circuit inputs that asserts gate_k. PO2 states that there is some combination of NULL, DATA, and illegal circuit inputs that asserts gate_k. If PO1 is false, but PO2 is true, then gate_k is flagged as a potential Trojan (i.e., gate_k isn't asserted when inputs are only NULL/DATA, but can be asserted when 1 or more inputs are illegal).

PO1: $\langle \forall\ i_1, i_2, \ldots, i_n \in \{\text{NULL, DATA0, DATA1}\} :: \{g_k = 1\} \rangle$

PO2: $\langle \forall\ i_1, i_2, \ldots, i_n \in \{\text{NULL, DATA0, DATA1, Illegal}\} :: \{g_k = 1\} \rangle$

VI. Results

Fig. 5 shows a sample NCL RSA decryption [10] circuit, where n and e are output as part of the public key, but d is the private key exponent that must be kept secret. The inputs are a bit (*n/e Select*) to select whether n or e should be output on the Public Key channel, and the encrypted padded message, c, to be decoded. The external interface also includes the NCL handshaking signals, Ki_{PK}, Ko_{PK}, and Ko_C. Note that the n, e, and d registers utilize read ports [11], since they are not loaded with new data every cycle; and for circuit simplicity, the d and e registers are shown padded with MSB DATA0s so that they contain the same number of bits as n. The circuit, corresponding to Node A, operates as follows. If Node B wants to send a secure message to Node A, Node A transmits n and e over an unsecured channel, such that they are publicly known. Node B uses n and e to encrypt its plaintext message, M, resulting in an encrypted message, c. Node B then transmits its encrypted message, c, over an unsecured channel to Node A, which in turn decrypts the message using public key, n, and its private key exponent, d, which is only known to Node A. Without knowing d, the encrypted message cannot be decoded.

This circuit also includes the Fig. 4a, hardware Trojan, shown in red, which causes the private key exponent, d, to be leaked on the Public Key channel when triggered by sending an encrypted message with *bit(X)* as an illegal value, and selecting e to be output via *n/e Select*. VHDL simulation of

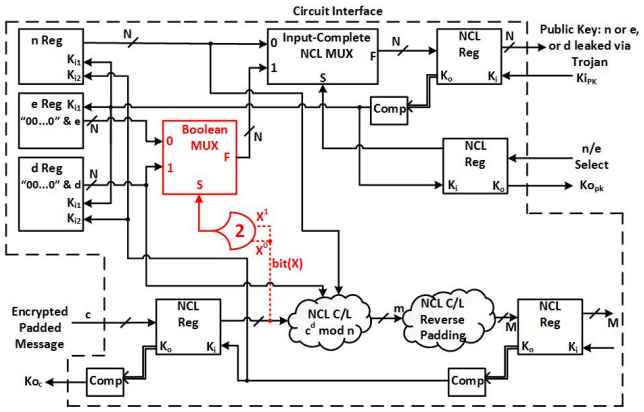

Fig. 5: NCL RSA decryption circuit with hardware Trojan inserted to leak private data

this circuit confirmed correct operation when all inputs were some combination of NULL and DATA, and that *d* was leaked in the circumstance described above. The Public Key output is input-complete with respect to (wrt) *n*, *e*, and *n/e Select*; and *M* is input-complete wrt *d* and *c*. Therefore, the circuit as a whole is input-complete. As detailed in Section V, insertion of the Fig. 4a Trojan does not compromise observability; therefore, the circuit as a whole is both input-complete and observable, and passes both of these checks described in [7]. This circuit also passes the functional verification and handshaking checks described in [7], such that it is confirmed as being properly verified. However, as described above and confirmed via VHDL simulation, the circuit is not correct due to insertion of the hardware Trojan.

The POs presented in Section V were automatically encoded for the example NCL RSA circuit, and checked using the Z3 SMT Solver [12], resulting in the red TH22 gate in Fig. 5 being flagged as a potential Trojan.

VII. Conclusions and Future Work

This paper describes how hardware Trojans can be realized in asynchronous NCL circuits using an illegal value as the trigger mechanism. While the existing NCL observability verification methodology [7] was able to detect two of the three presented Trojans, one variation escaped detection using existing NCL verification approaches. Hence, we proposed an additional verification methodology that detects this previously undetectable hardware Trojan.

Hardware Trojans can also be inserted into other types of asynchronous circuits, such as MTNCL [13] and PCHB [14]. Since MTNCL circuits are not required to be input-complete or observable, detecting MTNCL Trojans is expected to be more difficult. The Fig. 4b Trojan can be easily detected in MTNCL circuits by replacing $\{g_k = 1\}$ with $\{g_k = 0\}$ in both PO1 and PO2; however, the Fig. 4c Trojan would not be as easily detected in MTNCL circuits. As future work, we plan to explore Trojan insertion and detection in MTNCL and PCHB circuits, as well as investigate how several other types of Trojans, commonly found in synchronous circuits, could be adapted to asynchronous circuits.

VIII. Acknowledgment

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

References

[1] M. Tehranipoor and F. Koushanfar, "A survey of hardware trojan taxonomy and detection," *IEEE Design Test of Computers*, vol. 27, no. 1, pp. 10–25, 2010.

[2] B. Shakya, T. He, H. Salmani, D. Forte, S. Bhunia, and M. Tehranipoor, "Benchmarking of hardware trojans and maliciously affected circuits," *Journal of Hardware and Systems Security*, vol. 1, no. 1, pp. 85–102, 2017.

[3] R. S. Chakraborty, S. Narasimhan, and S. Bhunia, "Hardware trojan: Threats and emerging solutions," in *2009 IEEE International high level design validation and test workshop*. IEEE, 2009, pp. 166–171.

[4] S. Bhunia, M. S. Hsiao, M. Banga, and S. Narasimhan, "Hardware trojan attacks: Threat analysis and countermeasures," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1229–1247, 2014.

[5] S. C. Smith and J. Di, "Designing asynchronous circuits using null convention logic (ncl)," *Synthesis Lectures on Digital Circuits and Systems*, vol. 4, no. 1, pp. 1–96, 2009.

[6] K. M. Fant and S. A. Brandt, "Null convention logic/sup tm/: a complete and consistent logic for asynchronous digital circuit synthesis," in *Proceedings of International Conference on Application Specific Systems, Architectures and Processors: ASAP '96*, 1996, pp. 261–273.

[7] A. Sakib, S. Le, S. C. Smith, and S. Srinivasan, "Formal verification of ncl circuits," *Asynchronous Circuit Applications*, p. 309, 2020.

[8] K. Inaba, T. Yoneda, T. Kanamoto, A. Kurokawa, and M. Imai, "Hardware trojan insertion and detection in asynchronous circuits," in *2019 25th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*. IEEE, 2019, pp. 134–143.

[9] S. R. Hasan, S. F. Mossa, C. Perez, and F. Awwad, "Hardware trojans in asynchronous fifo-buffers: From clock domain crossing perspective," in *2015 IEEE 58th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2015, pp. 1–4.

[10] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[11] R. B. Reese, S. C. Smith, and M. A. Thornton, "Uncle - an rtl approach to asynchronous design," in *2012 IEEE 18th International Symposium on Asynchronous Circuits and Systems*, 2012, pp. 65–72.

[12] L. De Moura and N. Bjørner, "Z3: An efficient smt solver," in *International conference on Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340.

[13] L. Zhou, R. Parameswaran, F. A. Parsan, S. C. Smith, and J. Di, "Multi-threshold null convention logic (mtncl): An ultra-low power asynchronous circuit design methodology," *Journal of Low Power Electronics and Applications*, vol. 5, no. 2, pp. 81–100, 2015.

[14] A. J. Martin and M. Nystrom, "Asynchronous techniques for system-on-chip design," *Proceedings of the IEEE*, vol. 94, no. 6, pp. 1089–1120, 2006.