Enhancing Trojan Detection by Finding LTL and Taint Properties in RTL Circuit Designs: A Case Study

Juan Portillo
Department of Electrical and Computer Engineering
University of Texas at San Antonio
1 UTSA Circle
San Antonio, TX, USA
juan.portillo@my.utsa.edu

Eugene John
Department of Electrical and Computer Engineering
University of Texas at San Antonio
1 UTSA Circle
San Antonio, TX, USA
eugene.john@utsa.edu

***Abstract*— The advent of multi-vendor participation for circuit design has introduced new hardware security concerns, mainly, exploitation of code bugs and insertion of malicious circuits (Hardware Trojans). In this work, we detect register transfer language (RTL) Trojan designs using Linear-time Temporal Language (LTL) and Taint Propagation (TP) to verify components and signal paths on suspicious circuits. We demonstrate that LTL methods can detect denial of service (DoS), data leak, and change of functionality attacks while TP methods can detect DoS, and data leak attacks. Some Trojans can also be described as an LTL and a Taint problem which allows multiple tools to be used for security validation and improves the odds of detecting Trojan vulnerabilities.**

Keywords— 3PIP; SoC; Security; Design Defects; Hardware Vulnerabilities; Hardware Trojans; Formal Verification; Formal Testing; LTL; Taint Propagation;

I. INTRODUCTION

Modern day chip integrators use third-party intellectual property (3PIP) to help lower non-reoccurring engineering (NRE) costs and speed up time-to-market (TTM). Integrating 3PIP from third-party vendors carries risks which include incomplete or improper testing by the third-party vendor or intentionally inserted malicious logic by the third-party vendor which can be exploited for malicious use by anyone with knowledge of the vulnerability. It is left to the integrator to verify the system-on-chip (SoC). Insufficient verification may lead to the presence of code bugs or intentionally inserted hardware Trojans in the 3PIP [1] [2] [3].

Currently, it is infeasible for the integrator to exhaustively verify the correct operation of an entire 3PIP using functional verification under time constrained conditions due to covering an exponential number of test input values and logical states even after redundant patterns have been eliminated [4]. While formal verification provides an automated way to exhaustively verify a single test property, sometimes a solution may not converge since there may be an exponential number of states to test for that property [5]. Exhaustive verification using a formal analysis approach presents challenges for a SoC if some solutions do not converge in reasonable time frames or within the verification system's memory constraints. Formal verification still relies on the manual creation of test properties for each component within a 3PIP which makes exhaustive 3PIP verification impractical. Therefore a feasible approach to formal verification may be to reduce the scope of security properties to system specifications [6].

Previous hardware (HW) Trojan detection techniques rely on some form of functional verification [5] [7] [8] [9]. Techniques which focus on detecting the trigger mechanism include FANCI [10] and VeriTrust [8]. Other techniques formally verify properties which describe invalid modifications to registers [11] or perform model checking on specifications and paths [12] [13]. Rathmair et al. [13] detects hardware Trojans by formally proving properties which describe the absence of all defined Trojan behavior. In Ref [14], Rajendran et al. detects key leak payloads using formal verification of security properties.

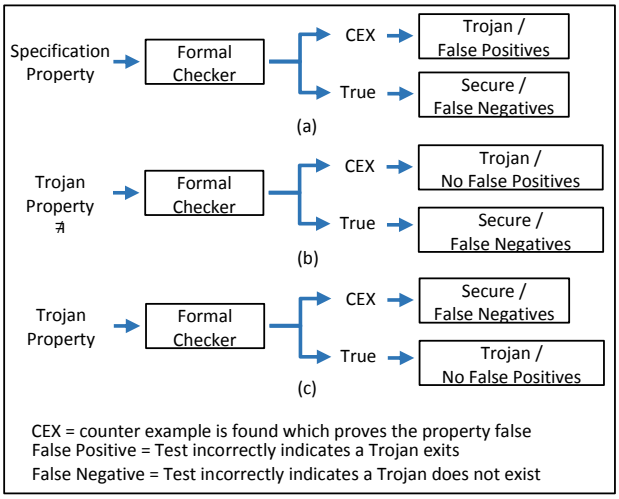
Model checking (formal analysis) of a circuit can detect code bug or Trojan vulnerabilities by a) proving that security properties that describe valid specification behavior always occur [12], b) proving that security properties which describe known or expected Trojan behavior do not occur [13], or c) proving security properties that describe Trojan behavior do occur [14]. Using method (a) may not necessarily guarantee that a 3PIP is free of vulnerabilities due to incomplete specifications or inaccurate security properties [15] and may result in false negatives if the proof holds true or false positives if a counter-example (CEX) to the proof is found. Method (b) faces the same challenges as (a) as well as contending with a larger space for property coverage [16] and may result in false negatives if the proof is false but no false positives since the property has defined the exact behavior for the vulnerability. Method (c) faces the same challenges as (a) and (b) and will also result in false negatives if the proof holds true but will result in no false positives for the same reason as method (b) (Figure 1). Vulnerability detection tools from [17] and [18] show that proving only security properties which describe valid circuit behavior is sufficient for Trojan detection to a degree of certainty. Satisfying a security property does not guarantee that a 3PIP is free of vulnerabilities [15], but increasing the quality of the security property test improves vulnerability detection rates [19].

Krieg et al. [12], proposed design partitioning for making exhaustive verification on the entire design feasible. Ref. [19] breaks down large SoC designs into security sensitive assets that can be formally verified to provide security assurance to a degree of certainty. Leveraging the power of formal analysis reveals unspecified behavior that is undesirable since it is not functionally

978-1-5386-2652-8/17 $31.00 © 2017 IEEE
DOI 10.1109/CSCI.2017.13

Authorized licensed use limited to: ETRI. Downloaded on July 03,2025 at 02:05:17 UTC from IEEE Xplore. Restrictions apply.

Figure 1: Property definition approaches and interpretation of proof outcomes



or structurally valid [4] or is a code [17] [19] [20] or HW Trojan vulnerability [11] [12] [13] [14] [18] [19].

In this research, we apply linear-time temporal language (LTL) and taint propagation (TP) properties for component checking (CC) and path equivalence checking (PEC). LTL ensures chip integrity and availability and TP ensures confidentiality. In some cases LTL can also provide chip confidentiality. This provides enough coverage to detect three major classes of register transfer language (RTL) Trojan payloads as defined by TrustHub [21]. These two approaches, LTL and TP, taken together, improve the odds of Trojan-free designs. The main contribution of this paper is to demonstrate that formal tools for Trojan detection can be enhanced by approaching the same Trojan circuit as an LTL 'and' a Taint problem. Although not demonstrated in this paper, this technique can also be applied to code bugs since they share similar traits to Trojan vulnerabilities; namely they can have the same effect as a Trojan payload.

The rest of this paper is organized as follows: Section 2 provides a background on existing Trojan detection techniques and the motivation for the proposed technique, Section 3 describes the methods used for demonstrating the proposed technique, Section 4 discusses the results of applying CC and PEC on several test circuits, and Section 5 concludes the work with a summary and discussion of the results.

II. BACKGROUND

Design defects and malicious HW Trojans can be detected at the RTL level. Defects may allow a malicious end user a way to exploit security weakness in the final end user product [20]. HW Trojans can arise due to a rogue designer in the third-party design house who inserts a malicious line of code which can be triggered by the end-user using malicious third-party software (SW) on the OS or by internal or external data or control signals [22].

Detecting hardware Trojans has been the focus of intense research in the past. Some of the previous research includes use of general verification methods such as functional verification [5] [7] [8] [9] or code coverage [5] to detect Trojan trigger mechanisms. Other researchers used highly customized methods to detect HW Trojan trigger mechanisms [10]. Methods using formal analysis include work by Rajendran et al. [11] which detect modification of critical registers and pseudo critical registers and detect the presence of bypass registers. This approach is equivalent to finding the Trojan payload. They were able to detect a variety of payloads including changes to functionality, denial of service (DoS), and data leaks. However, focusing on corruption of registers may limit detection of payloads due to invalid paths. One such Trojan which leaks a secret key causes the formal tool to time out before the trigger, a 128-bit counter, reaches its final value. In [14] Rajendran et al. expands on their work in [11] to detect crypto key leaks using formal verification of security properties that solve for the presence of key bits or the complement of key bits at output ports. The focus is on finding complex payloads such as a key bit stream encoded by a linear feedback shift register (LFSR).

Krieg et al. [12] detect a Trojan in a universal asynchronous receiver/transmitter (UART) circuit using formal analysis with model checking which performs verification of the UART FSM by proving assertions defined by the specification of the circuit behavioral model, and structural checking which detects changes to the interconnect at the gate level. Rathmair et al. [13] detects Trojans by formally testing for the absence of security properties which describe Trojan behavior and demonstrates detection of a Trojan payload which modifies data on the memory address interface by proving security properties which describe valid and invalid paths. Rathmair et al. [13] also proposes a Trojan Assurance Level (TAL) as a measure of threats due to Trojans. In [16], Rathmair et al. expand on this metric as a measure of how well Trojan properties provide security assurance for a circuit.

Previous work by Cabodi et. al. [23] uses TP (a class of CTL) based formal analysis for assuring key secrecy, crypto isolation, system availability, verification of authorized use, and data integrity. Cabodi et al. [24] ensures confidentiality, data integrity, and authentication of different embedded systems. They compare path properties, which describe valid paths such as those derived from a specification, against CTL based Taint Properties, which describe undesired behavior such as corrupted user code as a source which propagates to a destination through invalid paths.

Malik et. al [25] uses CTL and LTL to ensure circuit integrity and Dynamic Taint Analysis (aka Taint Propagation) and Secure Type Systems to ensure confidentiality and integrity. The circuit design is abstracted at the instruction level and a specification language is introduced so that Taint Analysis can be used for security verification of a system which includes FW and HW.

Our research demonstrates that threats to confidentiality, such as key leaks, can also be assured by using a LTL to define properties based on a known specification which can be proved with a formal analysis tool to verify the security of a white boxed RTL circuit model. Based on our observations, if a single Trojan can be described using LTL and TP, then the odds of detecting the Trojan increase, and thus, increases confidence in 3PIP security verification.

III. METHODS

The benchmark circuits used in this research represent an entire white boxed 3PIP containing a Trojan circuit. Trojan components and paths are known beforehand since exhaustive and complete circuit verification is not within the scope of this paper. The Trojans represent a sample of three major types of attacks with different trigger and payload designs. LTL or TP analysis was performed on the module depending on the security property of interest (Table 1). The LTL based properties were created using System Verilog Assertions (SVA) in Cadence

Authorized licensed use limited to: ETRI. Downloaded on July 03,2025 at 02:05:17 UTC from IEEE Xplore. Restrictions apply.

Figure 2: Example using LTL based formal analysis for Component Checking (CC)

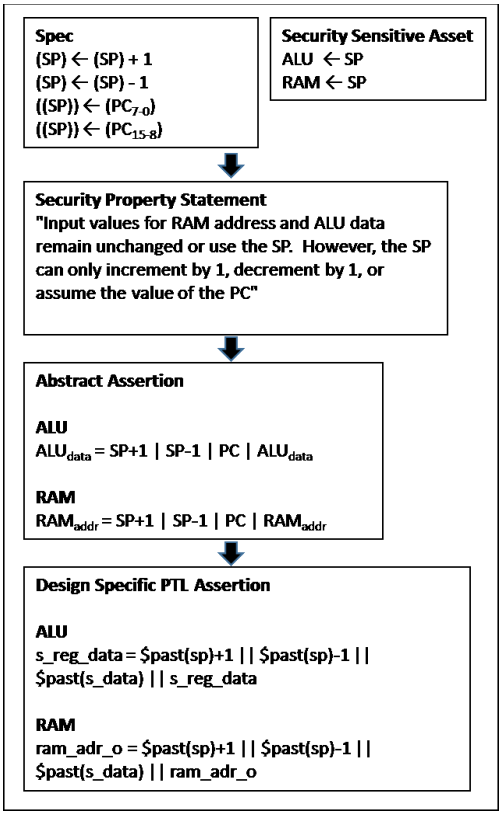
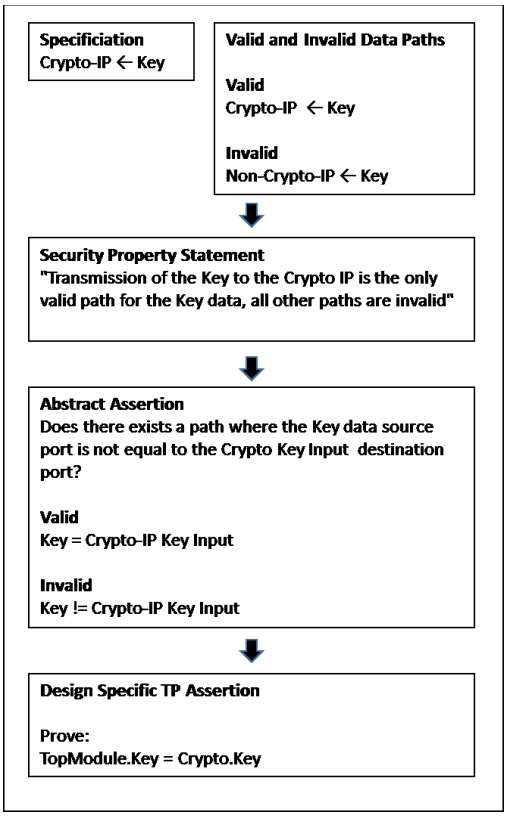


Figure 3: Example using LTL and TP based formal analysis for Path Equivalence Checking (PEC)



Jasper Gold Formal Property Verification (FPV) and TP based properties were created using a graphic user interface (GUI) in Cadence Jasper Gold Secure Path Verification (SPV) [26] to test CC or PEC techniques. The circuits used in this research, Trojan descriptions, and their detection methods are summarized in Table 1.

Formal analysis using LTL or TP allows detection of Trojans without knowledge of the specific Trojan trigger or payload mechanism. This is possible since formal analysis based methods convert the propositions (assertions) and the given circuit model into a mathematical expression and exhaustively validates the assertions using algebraic proofs [4]. Since the assertion is based on a specification, the formal tool effectively validates the specification against the circuit model in an exhaustive manner. This ensures that the specification is not violated as opposed to exhaustively searching for a trigger or payload. If the specification is violated then the formal tool produces a counter example which can be traced, with some manual effort by the verification engineer, to the Trojan trigger or payload.

In the case for LTL based formal analysis for CC, the formal tool will test the assertion (Section 3.2, Equation 1) until a counter example is found or all possible logic values for the given component have been searched. TP based formal analysis for PEC will test the assertion (Section 3.4, Equation 2) until a counter example is found or all possible data flows for a given source have been examined. In the case for LTL based formal analysis for PEC, the formal tool will test the assertion (Section 3.4, Equation 3) until a counter example is found or all possible conditions for the given path equivalency definitions have been applied. For any of these methods, if a counter example is found then the specification has been violated, thus indicating the presence of a defect or Trojan. If the test reaches completion without finding a counter-example then the assertion has been exhaustively searched and passes, thus indicating the absence of a Trojan.

A. Component Checking Example

The MC8051-T800 microcontroller circuit with a HW Trojan insertion is verified for the presence of a HW Trojan vulnerability using CC (Figure 2). ALU and RAM are dependent on a properly functioning stack pointer (SP), however, a Trojan modifies the SP to incorrectly increment by two. A security property written in SVA defines all valid operations for the SP. All other outcomes indicate a security vulnerability.

B. Generalizing Component Checking

Component checking (CC) ensures valid circuit behavior by detecting the presence of vulnerabilities. The valid circuit behavior is defined using the circuit specifications to design a security property which can be formally proven. A counter-example to the security property during formal analysis indicates there is a security vulnerability. Rathmair et al. [13] uses Equation (1) to describe how security properties can be used to find vulnerabilities in a circuit. Equation (1) states that each security property *O* for each corresponding specification *f* must be proven on the circuit model *M* to ensure a vulnerability-free design. In this equation we interpret *f* as a non-Trojan specification.

$$M \models O_f \, \forall \, f \qquad (1)$$

Authorized licensed use limited to: ETRI. Downloaded on July 03,2025 at 02:05:17 UTC from IEEE Xplore. Restrictions apply.

Table 1: List of Trojan trigger, payload mechanisms and their corresponding detection techniques

Circuit	Trigger	Payload	PTL	TP	Detection Method	Trust-Hub Paylod Category	Detected ?
MC8051-T800	depends on nexstate from controller FSM module = FETCH and opcode from ROM = MOV_D_DATA and a predefined address in ROM = xFF; sequential and combinational	modifies Stack Pointer; data integrity	✓		CC	denial of service	Yes
MC8051-T700	when state from controller FSM module = FETCH and instr from memory controller = MOV_D_DATA; sequential and combinational	modifies ROM data with zeros at input to memory controller at addresses specified by output of memory controller; data integrity		✓	PEC	denial of service	Yes
AES-T700	data input 'state' equals some predefined value; combinational	key leak; data confidentiality	✓	✓	PEC	leak information	Yes
AES-T1200	after encryption is performed a predefined number of times; sequential	key leak; data confidentiality	✓	✓	PEC	leak information	Yes
AES-T800	after some sequence of input 'state'; sequential	key leak; data confidentiality	✓	✓	PEC	leak information	Yes
MC8051-T400	after some sequence of four instructions; sequential	prevent counter for interrupt controller from starting by modifying path for tcon bit of counter 0 and counter 1; data integrity	✓		PEC	change functionality	Yes

The procedure for using LTL for CC can be generalized as follows:

- Define valid operations for the vulnerable architectural component i.e. Stack Pointer, Program Counter, etc.
- Define a relation between the vulnerable architectural component and security sensitive asset i.e. ALU, RAM, TIMER.
- State the bounds and limitations on the relationship between the vulnerable architectural component and the security sensitive asset.
- Define an abstract Boolean expression for the security property.
- Define a circuit specific Boolean expression to design the LTL assertion by refining the abstract assertion using specific references from the RTL design.

C. Path Equivalence Checking Example

The AES-T700 circuit implements the AES cryptographic algorithm. The corrupted version of this circuit has a HW Trojan vulnerability and is tested for the presence of any hardware vulnerability using PEC (Figure 3). Only the AES crypto module is authorized to receive the secret key. The security critical path is any path which involves the secret key input. PEC detects a key leak by verifying that only paths defined by the SVA security property as valid paths are allowed to transmit the key and that all other paths which distribute the secret key are invalid.

D. Generalizing Path Equivalence Checking

Path equivalence checking (PEC) ensures that data paths that are used to transfer information are not modified. The path can be between any two RTL modules or logical elements within a module. A PEC security property tests for equivalence of the source and destination of interconnects. Since we apply PEC using TP to find vulnerabilities due to invalid modifications to signal paths, we introduce Equation (2), which defines only paths between A and B_Y as valid for all paths starting from the *Nth* source.

$$A_N = B_{Y,N} \ \forall N \tag{2}$$

Equation (2) is sufficient for TP based assertions. We now require an equation which describes how to use LTL based assertions to find vulnerbiliteis due to modificaotins of signal paths. Rathmiar et al. [13] introduces such an equation for ensuring data integrity. We show that this equation can also be applied to ensure data confidentiality. Based on the equation from [13] we state Equation (3), where we define valid (V) source (A) and destination (B) pairs (N) and invalid (I) source and destination pairs.

$$(A_N = B_{N,V}) \ \& \ (A_N \neq B_{N,I}) \ \forall N \tag{3}$$

The procedure for using LTL or TP for PEC can be generalized as follows:

- Define the valid information flow requirements for data paths from the specification
- Identify security sensitive information and associated and non-associated data paths i.e. inter-module connections
- State the bounds and limitations on the relationship between security sensitive information and associated and non-associated data paths
- Define an abstract assertion statement by forming a Boolean expression for the security property
- Define a circuit specific assertion by refining the abstract assertion using circuit specific references from the RTL design for proving using LTL or TP formal methods

Authorized licensed use limited to: ETRI. Downloaded on July 03,2025 at 02:05:17 UTC from IEEE Xplore. Restrictions apply.

IV. RESULTS

LTL and TP detects even those events that lie outside the specification. The two techniques also help to enforce design specifications which may have been inadvertently omitted during the design phase and cover the corner cases which may have been missed during standard pre-silicon verification. All six Trojans in this study where detected using either 1) LTL or 2) TP. The Trojans represented three major types of attacks including; Change of functionality (integrity, availability), data leak (confidentiality), and DoS (integrity, availability) using Trojan payloads from six TrustHub [21] benchmarks. Three Trojans (AES-T700/T1200/T800) having the same payloads but different activation mechanisms were detected to show that LTL and TP methods are independent of the Trojan trigger design in data leak attacks. LTL properties were used to detect DoS, data leaks, and change of functionality. TP was used to detect DoS, and data leaks. Most significantly, LTL can be applied on an untrusted component (MC8051-T800) as well as an untrusted data path (MC8051-T400). LTL and TP are independent of the specific payload as long as there is a valid specification defined for the path (PEC) or component (CC) under test (Table 1). All formal proofs converged within a few minutes.

A. Component Checking (CC)

Component checking was used to detect more than one type of Trojan payload using LTL based verification in Jasper Gold FPV.

1) Availability

LTL based formal methods using the CC approach detected a DoS attack due to a Trojan in circuit MC8051-T800. The formal verification tool finds a counterexample to the security property which defines valid operations for the stack pointer function. The tool returns the out-of-spec stack pointer operation and the Trojan logic that produces this error.

B. Path Equivalence Checking (PEC)

Path equivalence checking was used to detect change of functionality, data leak, and DoS Trojans using a combination of SVA in Jasper Gold FPV and manual assertions in Jasper Gold SPV.

1) Integrity

LTL based formal methods using the PEC approach detected change of functionality due to a Trojan in circuit MC8051-T400. The verification tool detects an invalid modification of the interrupt counter variable *tcon* along the valid path staring from the *tcon* bit assignment and ending at its input to the timer counter. While this path is a valid circuit path, it does not pass the security property which tests for data equivalence from source to destination.

2) Confidentiality

TP based formal methods using the PEC approach detected malicious data leaks due to a Trojan in circuits AES-T700/T1200/T800. After all valid key-carrying paths are defined in the security property test, the verification tool detected an invalid key-carrying path from the port of the top level AES module to the port of a sub-module not authorized to receive the key. This sub-module is the Trojan payload which encodes the key using an LFSR and leaks the obfuscated key data to a side channel.

LTL based formal analysis was used with the PEC approach to detect circuits which leaked a key to an unauthorized sub-module. Key leaks in circuits AES-T700, AES-T1200, and AES-T800 where found by testing assertions which checked if the I/O of modules of different hierarchies where properly connected as defined by the specification.

3) Availability

TP based formal methods using the PEC approach detected a DoS attack due to a Trojan in circuit MC8051-T700. The verification tool indicates that the Trojan logic does not follow design rules for the path under test and produces an error.

V. CONCLUSION

In this research we show that using formal tools for Trojan detection can be extended by finding ways to describe a single Trojan circuit as an LTL and a Taint Propagation problem. Security verification is achieved by proving security properties defined by a specification rather than trying to find a specific trigger or payload. Six test circuits, each with one Trojan vulnerability, were analyzed using formal verification, resulting in the detection of all 6 Trojans. Several Trojans having the same payload but different trigger mechanisms were tested to show that Trojans can be detected regardless of the trigger mechanism.

Taken together, LTL and TP can be used to detect three major classes of RTL design vulnerabilities. In some cases components and paths can be verified using either LTL or TP formal methods to detect the same type of attack, such as, DoS. In addition, these vulnerability detection techniques are independent of the Trojan design implementation.

REFERENCES

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

[2] M. Rostami, F. Koushanfar and R. Karri, "A primer on hardware security: Models, methods, and metrics," *Proceedings of the IEEE,* vol. 102, no. 8, pp. 1283-1295, Aug. 2014.

[3] A. Basak, S. Mal-Sarkar and S. Bhunia, "Secure and trusted SoC: Challenges and emerging solutions," in *Microprocessor Test and Verification (MTV), 2013 14th International Workshop on*, Austin, TX, 2013, pp. 29-34.

[4] D. L. Perry and H. D. Foster, Applied Formal Verification, New York, USA: McGraw-Hill, 2005.

[5] X. Zhang and M. Tehranipoor, "Case study: Detecting hardware Trojans in third-party digital IP cores," in *International Symposium on Hardware Oriented Security and Trust*, San Diego, CA, 2011, pp. 67-70.

[6] E. Love, Y. Jin and Y. Makris, "Enhancing security via provably trustworthy hardware intellectual property," in *Hardware-Oriented Security and Trust (HOST), 2011 IEEE International Symposium on*, San Diego, CA, 2011, pp. 12-17.

[7] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin and J. M. Smith, "Overcoming an untrusted computing

Authorized licensed use limited to: ETRI. Downloaded on July 03,2025 at 02:05:17 UTC from IEEE Xplore. Restrictions apply.

based: Detecting and removing malicious hardware automatically," in *2010 IEEE Symposium on Secuirty and Privacy*, Oakland, CA, 2010, pp. 159-172.

[8] J. Zhang, F. Yuan, L. Wei, Z. Sun and Q. Xu, "VeriTrust: Verification for hardware trust," in *IEEE/ACM Design Automation Conference*, Austin, TX, 2013, pp. 1-8.

[9] M. Banga and M. Hsiao, "Trusted RTL: Trojan detection methodology in pre-silicon designs," in *International Symposium on Hardware-Oriented Security and Trust*, Anaheim, CA, 2010, pp. 56-59.

[10] A. Waksman, M. Suozzo and S. Sethumadhavan, "FANCI: Identification of stealthy malicious logic using boolean functional analysis," in *ACM Conference on Computer and Communications Security*, Berlin, Germany, 2013, pp. 697-708.

[11] J. Rajendran, V. Vedula and R. Karri, "Detecting malicious modifications of data in third-party intellectual property cores," in *Design Automation Conference*, San Francisco, CA, 2015, pp. 1-6.

[12] M. R. Krieg and F. Schupfer, "A process for the detection of design-level hardware Trojans using verification methods," in *High Performance Computing and Communications, 6th Intl Symp on Cyberspace Safety and Security, 11th Intel Conf on Embedded Software and Syst (HPCC, CSS, ICESS)*, Paris, 2014, pp. 729-734.

[13] M. Rathmair, F. Schupfer and C. Krieg, "Applied formal methods for hardware Trojan detection," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, Melbourne VIC, 2014, pp. 169-172.

[14] J. Rajendran, A. M. Dhandayuthapany, V. Vedula and R. Karri, "Formal security verification of third party intellectual property cores for information leakage," in *2016 29th International Conference on VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID)*, Kolkata, 2016, pp. 547-552.

[15] Y. Jin and Y. Makris, "A proof-carrying based framework for trusted microprocessor IP," *IEEE/ACM International Conference on Computer-Aided Design,* pp. pp. 824-829, 2013.

[16] M. Rathmair and F. Schupfer, "Metrics for Formal Property Checking Against Undesired Circuit Behavior in Embedded Systems," in *ANALOG*, Bremen, Germany, 2016, pp. 1-6.

[17] P. Subramanyan and D. Arora, "Formal verification of taint-propagation secuirty properties in a commercial SoC design," *Design, Automation and Test in Europe (DATE),* pp. pp. 1-2, 2014.

[18] W. Hu, J. Oberg, J. Barrientos, D. Mu and R. Kastner, "Expanding gate level information flow tracking for multilevel security," *IEEE Embedded Systems Letters,* vol. 5, no. 2, pp. 25-28, June 2013.

[19] J. Portillo, S. Narasimhan and E. John, "Building trust in 3PIP using asset based security property verification," in *IEEE VLSI Test Symposium*, Las Vegas, NV, 2016, pp. 1-6.

[20] D. W. Palmer and P. K. Manna, "An efficient algorithm for identifying security relevant logic and vulnerabilities in RTL designs," in *Hardware-Oriented Security and Trust (HOST)*, Austin, TX, 2013, pp. 61-66.

[21] H. Salmani, M. Tehranipoor and R. Karri, "On design vulnerability analysis and trust benchmarks development," in *2013 IEEE 31st International Conference on Computer Design (ICCD)*, Asheville, NC, 2013, pp. 471-474.

[22] R. Karri, J. Rajendran, K. Rosenfeld and M. Tehranipoor, "Trustworthy hardware: Identifying and classifying hardware Trojans," *Computer,* vol. 43, no. 10, pp. 39-46, Oct. 2010.

[23] G. Cabodi, P. Camurati, S. F. Finocchiaro, C. Loiacono, F. Savarese and D. Vendraminetto, "Secure embedded architectures: Taint properties verification," in *2016 International Conference on Development and Application Systems (DAS)*, Suceava, Romania, 2016, pp. 150-157.

[24] G. Cabodi, P. Camurati, S. F. Finocchiaro, C. Loiacono, F. Savarese and D. Vendraminetto, "Secure Path Verification," in *2016 1st IEEE International Verificatoin and Security Workshop (IVSW)*, St. Feliu de Guixols, 2016, pp. 1-6.

[25] S. Malik and P. Subramanyan, "Invited: Specification and modeling for Systems-on-Chip security verification," in *2016 53rd ACM/EDAC/IEEE Design Automatoin Conference (DAC)*, Austin, TX, 2016, pp. 1-6.

[26] "Cadence Jasper Gold". www.cadence.com, accessed May 2016

Authorized licensed use limited to: ETRI. Downloaded on July 03,2025 at 02:05:17 UTC from IEEE Xplore. Restrictions apply.