Pre-Silicon Security Verification and Validation: A Formal Perspective

Xiaolong Guo
University of Central Florida
Orlando, FL 32816
guoxiaolong@knights.ucf.edu

Raj Gautam Dutta
University of Central Florida
Orlando, FL 32816
rajgautamdutta@knights.ucf.edu

Yier Jin
University of Central Florida
Orlando, FL 32816
yier.jin@eecs.ucf.edu

Farimah Farahmandi
University of Florida
Gainesville, FL 32611
farimah@cise.ufl.edu

Prabhat Mishra
University of Florida
Gainesville, FL 32611
prabhat@cise.ufl.edu

Invited

ABSTRACT

Reusable hardware Intellectual Property (IP) based System-on-Chip (SoC) design has emerged as a pervasive design practice in the industry today. The possibility of hardware Trojans and/or design backdoors hiding in the IP cores has raised security concerns. As existing functional testing methods fall short in detecting unspecified (often malicious) logic, formal methods have emerged as an alternative for validation of trustworthiness of IP cores. Toward this direction, we discuss two main categories of formal methods used in hardware trust evaluation: theorem proving and equivalence checking. Specifically, proof-carrying hardware (PCH) and its applications are introduced in detail, in which we demonstrate the use of theorem proving methods for providing high-level protection of IP cores. We also outline the use of symbolic algebra in equivalence checking, to ensure that the hardware implementation is equivalent to its design specification, thus leaving little space for malicious logic insertion.

1. INTRODUCTION

The impact of malicious logic and design flaws in IP cores threatens to ruin the credibility of third-party vendors and places unnecessary security risks on the IP customers and end users. Existence of a malicious IP core invalidates the applicability of many of the previously proposed methods for Hardware Trojan detection [1, 6, 19, 23, 31, 32]. Most of the existing methods rely on golden models to generate the fingerprints and compare them with those measured from circuit-under-test using certain data analysis methods.

To counter the threat of untrusted third-party resources, pre-silicon trust evaluation approaches have been proposed recently [2, 16, 36]. Most of these methods try to trigger malicious logic by enhancing functional testing with extra test vectors. Authors in [36] proposed a method to generate "Trojan Vectors" into the testing patterns, hoping to activate the Hardware Trojans during the functional testing. In order to identify suspicious circuitry, unused circuit identification (UCI) [16] method analyzed the RTL code to find lines of code that are never used. However, these methods assume that the attacker uses rarely-occurring events as Trojan triggers. Using "less-rare" events as trigger will void these approaches. This was demonstrated in [35], where Hardware Trojans were designed to defeat UCI.

Admitting the limitations of enhanced functional testing methods, researchers started looking into formal solutions. Although at its early stage, formal methods have already shown their benefits over testing methods in exhaustive security verification [18, 21, 25, 39]. A multi-stage approach, which included assertion based verification, code coverage analysis, redundant circuit removal, equivalence analysis, and use of sequential Automatic Test Pattern Generation (ATPG) was adopted in [39] to identify suspicious signals for detecting Hardware Trojans. This approach was demonstrated on a RS232 circuit and the efficiency of the approach in detecting Trojan signals ranged between 67.7% and 100%. In [18, 21, 25], the PCH framework was used to verify security properties on soft IP cores. Supported by the Coq proof assistant [17], formal security properties can be formalized and proved to ensure the trustworthiness of IP cores. In this survey, we review the existing formal verification methods for soft IP cores with specific focuses on theorem proving and equivalence checking.

The rest of the paper is organized as follows: Section 2 discusses the theorem proving approach for hardware trust evaluation. In this section, the PCH framework is introduced as well as its applicability in verifying synthesizable register-transfer level (RTL) code and netlist of soft IP cores. Section 3 discusses the existing equivalence checking methods for ensuring trustworthiness of soft IP cores. Finally, Section 4 concludes the paper.

2. THEOREM PROVING FOR VALIDATION OF HARDWARE TRUST

Theorem provers are used to prove or disprove properties of systems expressed as logical statements. Since 1960s, several automated and interactive theorem provers have been developed and used for proving properties of hardware and

DAC '15, June 07 - 11, 2015, San Francisco, CA, USA

http://dx.doi.org/10.1145/2744769.2747939

software systems. However, verifying large and complex systems using theorem provers require excessive effort and time. Despite these limitations, theorem provers have currently drawn a lot of interest in verification of security properties on hardware. Among all the formal methods, they have emerged as the most prominent solution for providing high level protection of the underlying designs. In this section, we introduce the PCH framework, which uses an interactive theorem prover for verifying security properties on soft IP cores.

2.1 Proof-Carrying Hardware Framework

Proof-Carrying Hardware is an approach for ensuring trustworthiness of hardware [9, 10, 24, 25]. The PCH method is inspired from the proof-carrying code (PCC), which was proposed by G. Necula [30]. Using the PCC mechanism, untrusted software developers/vendors certify their software code. During the certification process, software vendor develops *safety proof* for the safety policies provided by software customers. The vendor then provides the user with a PCC binary file, which includes the formal proof of the safety properties encoded with the executable code of the software. The customer becomes assured of the safety of the software code by quickly validating the PCC binary file in a proof checker. Efficiency of this approach in reducing validation time at the customer end led to its adoption in different applications.

Following the concept of PCC, authors in [9–12] proposed the Proof-Carrying Hardware (PCH) framework for dynamically reconfigurable hardware platforms. In the PCH framework, authors used runtime combinational equivalence checking (CEC) for verifying equivalence between the design specification and the design implementation. A boolean satisfiability (SAT) solver was used to generate resolution proof for unsatisfiability of the combinational miter circuit, represented in a conjunctive normal form (CNF). The proof traces were combined with the bitstream into a proof-carrying bitstream by the vendor and given to the customer for validation. However, the approach did not consider exchange of a set of security properties between the customer and the vendor. Rather it considers safety policy, which included agreements on a specific bitstream format, on a CNF to represent combinational functions, and the propositional calculus for proof construction and verification.

2.2 Proof-Carrying Based RTL Verification

In [24, 25], another PCH framework was proposed, which overcame the limitations of the previous framework and expanded it for verification of security properties on soft IP cores. The new PCH framework is dedicated for security properties verification on synthesizable register-transfer level (RTL) IP cores. In the framework, Hoare-logic style reasoning is used to prove the correctness of the RTL code and implementation was carried out using the Coq proof assistant [17]. As Coq supports automatic proof checking, it can help IP customers to validate proof of security properties with minimum effort. Moreover, usage of the Coq platform by both IP vendors and IP consumers ensures that same deductive rules could be used for validating the proof. However, Coq does not recognize commercial hardware description languages (HDLs) and security properties expressed in a natural language. To solve this problem, semantic translation of HDLs and informal security specifications to calculus of inductive construction (CIC) was done. Based on this PCH framework, a new trusted IP acquisition and delivery protocol was proposed (See Figure 1), in which IP consumers provided both functional specifications and a set of security properties to IP vendors. IP vendors then developed the HDL code based on the functional specifications. The HDL code and security properties were then translated to CIC. Subsequently, proofs were constructed for security theorems and the transformed HDL code. The HDL code and proof for security properties were combined into a trusted bundle and delivered to the consumer. Upon receiving the trusted bundle, IP consumers first generate the formal representation of the design and security properties in CIC. The translated code, combined with formal theorems and proofs were quickly validated using the proof checker in Coq platform.

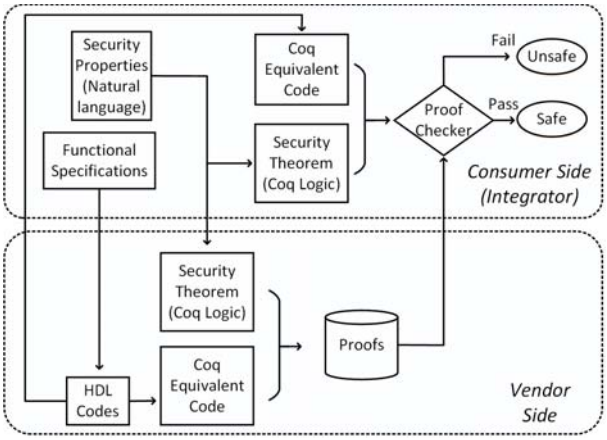

Figure 1: Working process of the PCH framework

Within the PCH framework defined in [25], the most important component is the set of security properties. A complete set of properties can ensure trustworthiness of the design by detecting malicious logic if present in the IP core. As different soft IP cores often share similar security properties, an expandable centralized repository of security properties with theorem-proof pairs will be a desirable solution for reducing the verification effort and protect the design from different types of hardware Trojan attacks. Any hardware designer will be able to pick a set of security properties from the property library rather than developing properties from scratch. The selected set of properties will be able to render many modes of attack significantly difficult to implement, thereby ensuring the trustworthiness of the delivered IP cores.

As a first step toward building such a property library, data secrecy properties were considered [20,21]. These properties help in tracking the internal information flow. Subsequently, the PCH framework formally prove that no sensitive information is leaked through the primary output or the Trojan side channels. The proof-carrying based static information flow scheme was demonstrated in [20]. In this method, each signal of the formal circuit had two values: logic value and signal sensitivity. The semantics of the formal circuit representation was updated to support both the logic signal propagation and signal sensitivity operation. Accordingly, a new formal model was developed for assigning appropriate sensitivity tags to each signals. With this framework, if proofs can be successfully constructed for the pre-defined data secrecy property, IP consumers can trust

that the delivered IP cores will not leak sensitive information through primary outputs. This static scheme has proven effective in detecting data leakage caused by hardware Trojans and/or design faults and require less effort in constructing the proof. However, the static scheme suffers from the limitation that it cannot be directly implemented on multi-stage designs and can only check circuit trustworthiness statically. To overcome the shortcoming of the static scheme and still achieve high level protection, a dynamic information assurance scheme was developed. The new scheme supports various levels of circuit architectures, ranging from low-complexity small-scale designs to large-scale deeply-pipelined design. Similar to the static scheme, the dynamic scheme also focuses on circuits with sensitive information, such as cryptographic designs.

Within the dynamic scheme, all signals are assigned values indicating their sensitivity levels. Based on the original values of signals and the update rules defined by the signal sensitivity transition model, values of these signals are updated after each clock cycle. As the sensitivities of all circuit signals are managed in a sensitivity list, two sensitivity lists are of interest for data secrecy protection: the initial sensitivity list and the stable sensitivity list. The initial sensitivity list reflects the circuit status after initialization or powered-on mode, when only some of the input signals contain sensitive information such as plaintext, encryption keys, etc. The stable sensitivity list, on the other hand, indicates the circuit status when all internal/output signals have fixed sensitivity levels.

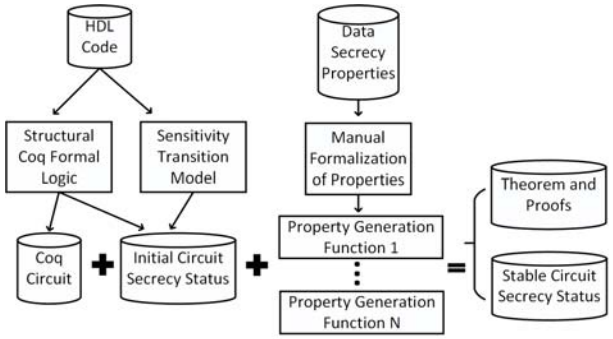

Figure 2: Trusted bundle preparation by IP vendors in the dynamic information assurance scheme

Figure 2 illustrates the preparation process of the trusted bundle defined by the dynamic information assurance scheme. A new structural Coq formal logic and a signal sensitivity transition model are developed for converting the HDL code to Coq representatives. Using the functional specifications, the IP vendor designs the HDL code and then uses the structural Coq formal logic and the signal sensitivity transition model to convert the HDL code into the language of Coq. The IP vendor also translates the agreed-upon data secrecy properties from natural language to property generation functions, which then helps in generating formal theorems.

The data secrecy property verification procedure performed by the IP consumer in the dynamic scheme is shown in Figure 3. In the first step, IP consumers check the contents of the initial signal sensitivity list and the stable signal sensitivity list. These lists represent the circuit's initial secrecy status and stabilized status. Validity of the initial list is checked to ensure that sensitivity levels are appropriately assigned to all input/output/internal signals. The stable sensitivity list contains complete information of the distribution of sensitive information across the whole circuit. Evaluating this list helps to detect hardware Trojans, which may illegally propagate sensitive information to primary outputs of the circuit.

After both the signal sensitivity lists pass the initial step, IP consumers proceed to the next step of automatic proof checking. A "PASS" output from the checker provides evidence that the HDL code do not contain any malicious channels. However, a "FAIL" results in a warning that some of the data secrecy properties are breached in the delivered IP cores. The PCH framework has been applied in cryptographic circuits such as DES, AES [20, 21].

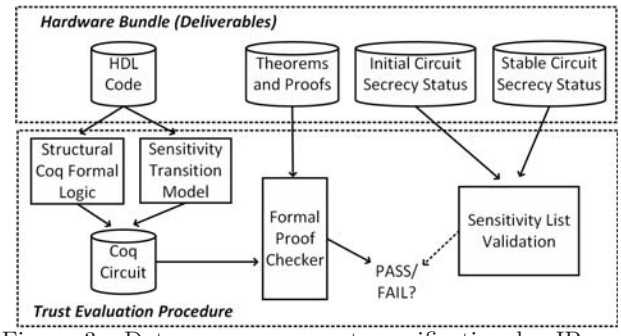

Figure 3: Data secrecy property verification by IP consumers in dynamic information assurance scheme

2.3 Proof-Carrying Based Netlist Verification

Besides the RTL code verification, the proof-carrying based information assurance scheme was extended to support gate level circuit netlist [18]. By leveraging the new gate-level framework, the authors in [18] formally analyzed the security of design-for-test (DFT) scan chains, the industrial standard testing method, and formally proved that a circuit with scan chain can violate data secrecy property. Although security concerns caused by DFT scan chains have been under investigation for decades, with various attack and defense methods being developed [8, 29, 33, 34, 37, 38], it is the first time it has been formally proved that the scan chain inserted designs are vulnerable (Note that RTL verification methods can rarely touch scan chains because scan chains are inserted in the netlist). The same framework was also applied in built-in-self-test (BIST) structure to prove that BIST structure can also leak internal sensitive information [18].

3. EQUIVALENCE CHECKING FOR HARDWARE TRUST VALIDATION

Orthogonal to the theorem prover based approaches, another promising approach is equivalence checking to ensure that the specification and implementation are equivalent. Figure 4 shows a traditional approach for performing equivalence checking using SAT solvers. If the specification and implementation are equivalent, the output of the "xor" gate should be always zero (false). If the output becomes true for any input sequence, it implies that the specification and the implementation are producing different outputs for the same input sequence. Therefore, if we construct CNF clauses of the input cone of F, we can use a SAT solver to perform

equivalence checking. If the SAT solver finds a satisfiable assignment, the specification and implementation are not equivalent. Traditional equivalence checking techniques can lead to state space explosion when large IP blocks are involved with significantly different specification and implementation. Similarly, traditional equivalence checking approaches fail for complex arithmetic circuits with larger bitwidths.

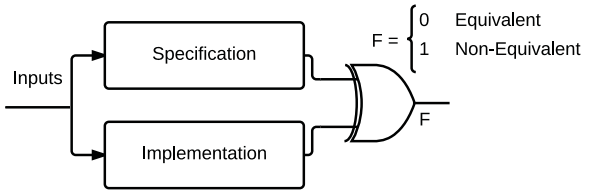

Figure 4: Equivalence Checking using SAT Solvers

A promising direction to address the state space explosion problem in verification of arithmetic circuits is to employ equivalence checking using computer symbolic algebra. Arithmetic circuits constitute a significant portion of datapath in signal processing, cryptography, multimedia applications, error root causing codes, etc. In most of them, arithmetic circuits have a custom structure and can be very large so the chances of potential malfunction is high. These bugs may cause unwanted operations as well as security problems like leakage of secret key [3]. Thus, verification of arithmetic circuits is very important.

Application of symbolic algebra for verification of combinational multipliers that support Galois field $\mathbb{F}_{2^k}$ computation (with function *P(x)* for field construction) is presented in [26]. The primary goal is to check equivalence between the specification polynomial f (with coefficient in $\mathbb{F}_{2^k}$) and gate level implementation C. The specification of arithmetic circuit and implementation are formulated as polynomials constructing a multivariate ring with coefficients from $\mathbb{F}_{2^k}$. This method uses *Gröbner basis* and *Strong Nullstellent* over Galois field to formulate the verification problem as an ideal membership testing of polynomial f in the ideal constructed by circuit polynomials (ideal I). Ideal I can have several generators, one of these generators is called Gröbner basis. First, we briefly describe Gröbner basis theory [7]. Next, we present application of Gröbner basis theory for verification of arithmetic circuits.

Let $M = {x_1}^{\alpha_1}{x_2}^{\alpha_2}...{x_n}^{\alpha_n}$ be a monomial and $f = C_1M_1 + C_2M_2 + ... + C_tM_t$ be a polynomial with $\{c_1, c_2, ..., c_t\}$ as coefficients and $M_1 > M_2 > ... > M_t$. Monomial $lm(f) = M_1$ is called leading monomial $lt(f) = C_1M_1$ is called leading term of polynomial f. Let $\mathbb{K}$ be a computable field and $\mathbb{K}[x_1, x_2, ..., x_n]$ be a polynomial ring in n variables. Then $< f_1, f_2, ..., f_s >= \{\sum_{i=1}^{n} h_i f_i : h_1, h_2, ..., h_s \in \mathbb{K}[x_1, x_2, ..., x_n]\}$ is an ideal I. The set $\{f_1, f_2, .., f_s\}$ is called generator or basis of ideal I. If $V(I)$ shows the affine variety (set of all solution of $f_1 = F_2 = ... = f_s = 0$) of ideal I, $I(V) = \{f_i \in \mathbb{K}[x_1, x_2, ..., x_n] : \forall v \in V(I), f_i(v) = 0\}$. Polynomial f_i is a member of $I(V)$ if it vanishes on $V(I)$. Gröbner basis is one of the generators of every ideal I (when I is other than zero) that has a specific characteristic to answer membership problem of an arbitrary polynomial f in ideal I. The set $G = \{g_1, g_2, ..., g_t\}$ is called Gröbner basis of ideal I, if $\forall f_i \in I, \exists g_j \in G : lm(g_j)|lm(f_i)$.

The Gröbner basis solves the membership testing problem of an ideal using sequential divisions or reduction. The reduction operation can be formulated as follows. Polynomial f_i can be reducible by polynomial g_j if $lt(f_i) = C_1M_1$ (which is non-zero) is divisible by $lt(g_i)$ and r is the remainder ($r = f_i - \frac{lt(f_i)}{lt(g_j)}.g_j$). It can be denoted by $f_i \xrightarrow{g_j} r$. Similarly, f_i can be reducible with respect to set G and it can be represented by $f_i \xrightarrow{G}_+ r$. The set G is Gröbner basis ideal I, if $\forall f \in I, f_i \xrightarrow{G}_+ 0$. Gröbner basis can be computed using Buchburger algorithm [4]. However, Buchberger algorithm is computationally intensive and it may effect the performance drastically. It has been shown in [5] that if every pair (f_i, f_j) that belongs to set $F = \{f_1, f_2, ..., f_s\}$ (generator of ideal I) has a relatively prime leading monomials ($lm(f_i).lm(f_j) = LCM(lm(f_i).lm(f_j))$) with respect to order $>$, the set F is also Gröbner basis of ideal I. Based on these observations, efficient equivalence checking between specification of an arithmetic circuit and its implementation can be performed as shown in Figure 5. The major computation steps in Figure 5 are outlined below:

- Assuming a computational field $\mathbb{K}$ and a polynomial ring $\mathbb{K}[x_1, x_2, ..., x_n]$ (note that variables $\{x_1, x_2, ..., x_n\}$ are subset of signals in the gate level implementation), a polynomial $f_{spec} \in \mathbb{K}[x_1, x_2, ..., x_n]$ representing specification of the arithmetic circuit can be derived.
- Map the implementation of arithmetic circuit to a set of polynomials that belongs to $\mathbb{K}[x_1, x_2, ..., x_n]$. The set F generates an ideal I. Note that according to the field $\mathbb{K}$, some vanishing polynomials that constructs ideal I_0 may be considered as well.
- Derive an order $>$ in a way that leading monomials of every pair (f_i, f_j) are relatively prime. Thus, the generator set F is also Gröbner basis $G = F$. As the combinational arithmetic circuits are acyclic, the topological order of the signals in the gate level implementation can be used.
- The final step is reduction of f_{spec} with respect to Gröbner basis G and order $>$. In other words, the verification problem is formulated as $f_{spec} \xrightarrow{G}_+ r$. The gate level circuit C has correctly implemented specification f_{spec}, if the remainder r is equal to 0. The non-zero remainder implies a bug or Trojan in the implementation.

Galois field arithmetic computation can be seen in Barrett reduction [28], Mastrovito multiplication and Montgomery reduction [22] which are critical part of cryptosystems. So verification of them in an efficient way is really important. In order to apply the method of Figure 5 for verification of Galios field arithmetic circuits, Strong Nullstellensatz over Galois Fields is used. Galois field is not an algebraically closed field, so its closure should be used. Strong Nullstellensatz helps to construct a radical ideal in a way such that $I(V_{\mathbb{F}^{2^k}}) = I + I_0$. Ideal I_0 is constructed by using vanishing polynomials ${x_i}^{2^k} - x_i$ by considering the fact that $\forall {x_i}^{2^k} \in \mathbb{F}_{2^k} : {x_i}^{2^k} - x_i = 0$. As a result, the Gröbner basis theory can be applied on Galois field arithmetic circuits. The method in [26] has extracted circuit polynomials by converting each gate to a polynomial and SINGULAR [15]

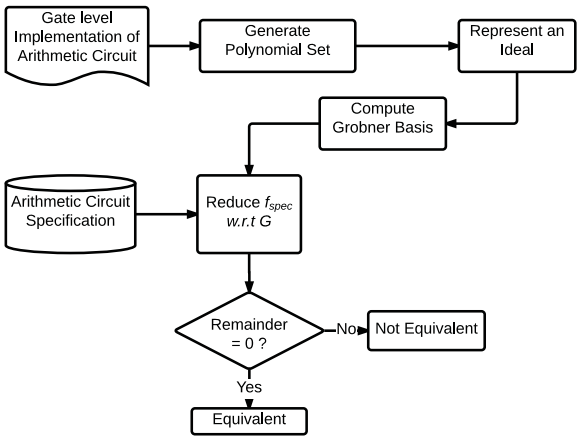

Figure 5: Equivalence checking flow.

has been used to do the $f_{spec} \xrightarrow{G}_{+} r$ computations. Using this method, the verification of Galois field arithmetic circuits like Mastrovito multipliers with up to 163 bits can be done in few hours. Some extensions of this method has been proposed in [27]. The cost of $f_{spec} \xrightarrow{G}_{+} r$ computation has been improved by mapping the computation on a matrix representing the verification problem, and the computation is performed using Gaussian elimination.

The Gröbner basis theory has been used to verify arithmetic circuits over ring $\mathbb{Z}[x_1, x_2, \ldots, x_n]/2^N$ in [14]. Instead of mapping each gate to a polynomial, the repetitive components of the circuit are extracted and the whole component is represented using one polynomial (since arithmetic circuit over ring $\mathbb{Z}[x_1, x_2, \ldots, x_n]/2^N$ contain carry chain, the number of polynomials can be very large). Therefore, the number of circuit polynomials are decreased. In order to expedite the $f_{spec} \xrightarrow{G}_{+} r$ computation, the polynomials are represented by Horner Expantion Diagrams. The reduction computation is implemented by sequential division. The verification of arithmetic circuit over ring $\mathbb{Z}[x_1, x_2, \ldots, x_n]/2^N$ up to 128 bit can be efficiently performed using this method. An extension of this method has been presented in [13] that is able to significantly reduce the number of polynomials by finding fanout-free regions and representing the whole region by one single polynomial. Similar to [27], the reduction of specification polynomial with respect to Gröbner basis polynomials is performed by Gaussian elimination resulting in verification time of few minutes. In all of these methods, when the remainder r is non-zero, it shows that the specification is not exactly equivalent with the gate level implementation. Thus, the non-zero remainder can be analyzed to identify the hidden malfunctions or Trojans in the system.

4. CONCLUSION

Growing reliance on hardware IPs, often gathered from untrusted third-party vendors, severely affects the security and trustworthiness of SoC computing platforms. A major concern with the hardware IPs acquired from external sources is that they may come with deliberate malicious implants to incorporate undesired functionality, undocumented test/debug interface working as hidden backdoor, or other integrity issues. SoC integrators typically tend to treat these IPs as black box and rely on the IP vendors on their structural/functional integrity. As various reports suggest that such a reliance can compromise the hardware security and trust. This paper surveyed existing formal methods for hardware IP trust validation: theorem proving and equivalence checking. We presented theorem proving approaches using proof-carrying hardware to enable high-level protection of IP cores. We also outlined equivalence checking techniques to guarantee that there are no malicious implants in the IP blocks by ensuring that the IP implementation faithfully represents the IP specification - nothing more, nothing less. We believe that the existing methods are promising but a lot more research effort is required to enable trusted SoC design using potentially untrusted components.

5. ACKNOWLEDGEMENT

This work was supported in part by the National Science Foundation grants (CCF-1218629, CNS-1319105 and CNS-1441667) and Semiconductor Research Corporation grant (2014-TS-2554).

6. REFERENCES

[1] D. Agrawal, S. Baktir, D. Karakoyunlu, P. Rohatgi, and B. Sunar. Trojan detection using IC fingerprinting. In *IEEE Symposium on Security and Privacy*, pages 296–310, 2007.

[2] M. Banga and M. Hsiao. Trusted RTL: Trojan detection methodology in pre-silicon designs. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 56–59, 2010.

[3] E. Biham, Y. Carmeli, and A. Shamir. Bug attacks. *Advances in Cryptology*, page 221–240, 2008.

[4] B. Buchberger. Ein algorithmus zum auffinden der basiselemente des restklassenringes nach einem nulldimensionalen polynomideal. *University of Innsbruck*, 1965.

[5] B. Buchberger. A criterion for detecting unnecessary reductions in the construction of a groebner bases. In *EUROSAM*, 1979.

[6] R. Chakraborty, F. Wolff, S. Paul, C. Papachristou, and S. Bhunia. MERO: A statistical approach for hardware Trojan detection. In *Cryptographic Hardware and Embedded Systems*, pages 396–410, 2009.

[7] D. Cox, J. little, and D. O'shea. Ideal, varieties and algorithm: An introduction to computational algebraic geometry and commutative algebra. In *Springer*, 2007.

[8] J. Da Rolt, G. Di Natale, M. L. Flottes, and B. Rouzeyre. Are advanced dft structures sufficient for preventing scan-attacks? In *VLSI Test Symposium (VTS)*, pages 246–251, 2012.

[9] S. Drzevitzky. Proof-carrying hardware: Runtime formal verification for secure dynamic reconfiguration. In *International Conference on Field Programmable Logic and Applications*, pages 255–258, 2010.

[10] S. Drzevitzky, U. Kastens, and M. Platzner. Proof-carrying hardware: Towards runtime verification of reconfigurable modules. In *ReConFig*, pages 189–194, 2009.

[11] S. Drzevitzky, U. Kastens, and M. Platzner. Proof-carrying hardware: Concept and prototype tool flow for online verification. *International Journal of Reconfigurable Computing*, vol. 2010, 2010.

[12] S. Drzevitzky and M. Platzner. Achieving hardware security for reconfigurable systems on chip by a proof-carrying code approach. In *6th International Workshop on Reconfigurable Communication-centric Systems-on-Chip*, pages 1–8, 2011.

[13] F. Farahmandi and B. Alizadeh. Groebner basis based formal verification of large arithmetic circuits using gaussian elimination and cone-based polynomial extraction. In *Microprocessors and Microsystems - Embedded Hardware Design*, pages 83–96, 2015.

[14] F. Farahmandi, B. Alizadeh, and Z.Navabi. Effective combination of algebraic techniques and decision diagrams to formally verify large arithmetic circuits. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 338–343, 2014.

[15] P. G. S. Greuel, G.-M. *2012. SINGULAR 3.1.3 A Computer Algebra System for Polynomial Computations. Centre for Computer Algebra.* http://www.singular.uni-kl.de.

[16] M. Hicks, M. Finnicum, S. T. King, M. M. K. Martin, and J. M. Smith. Overcoming an untrusted computing base: Detecting and removing malicious hardware automatically. In *Proceedings of IEEE Symposium on Security and Privacy*, pages 159–172, 2010.

[17] INRIA. The coq proof assistant, 2010. `http://coq.inria.fr/`.

[18] Y. Jin. Design-for-security vs. design-for-testability: A case study on dft chain in cryptographic circuits. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2014.

[19] Y. Jin and Y. Makris. Hardware Trojan detection using path delay fingerprint. In *IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 51–57, 2008.

[20] Y. Jin and Y. Makris. Proof carrying-based information flow tracking for data secrecy protection and hardware trust. In *VLSI Test Symposium (VTS)*, pages 252–257, 2012.

[21] Y. Jin, B. Yang, and Y. Makris. Cycle-accurate information assurance by proof-carrying based signal sensitivity tracing. In *IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 99–106, 2013.

[22] C. Koc and T. Acar. Montgomery multiplication in $GF(2^k)$. In *Designs, Codes and Cryptography*, volume 14, pages 57–69, 1998.

[23] C. Lamech, R. Rad, M. Tehranipoor, and J. Plusquellic. An experimental analysis of power and delay signal-to-noise requirements for detecting Trojans and methods for achieving the required detection sensitivities. *IEEE Trans. on Information Forensics and Security*, 6(3):1170–1179, 2011.

[24] 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*, pages 12–17, 2011.

[25] E. Love, Y. Jin, and Y. Makris. Proof-carrying hardware intellectual property: A pathway to trusted module acquisition. *IEEE Transactions on Information Forensics and Security*, 7(1):25–40, 2012.

[26] J. Lv, P. Kalla, and F. Enescu. Efficient groebner basis reductions for formal verification of galois field multipliers. In *Proceedings Design, Automation and Test in Europe Conf. (DATE)*, pages 899–904, 2012.

[27] J. Lv, P. Kalla, and F. Enescu. Efficient groebner basis reductions for formal verification of galois field arithmetic circuits. In *IEEE Transactions on CAD (TCAD)*, volume 32, pages 1409 – 1420, 2013.

[28] a. J. F. M. Knežević, and K. Sakiyama and I. Verbauwhed. Modular reduction in $GF(2^n)$ without pre-computational phase. In *Proceedings of the International Workshop on Arithmetic of Finite Fields*, pages 77–87, 2008.

[29] R. Nara, N. Togawa, M. Yanagisawa, and T. Ohtsuki. Scan-based attack against elliptic curve cryptosystems. In *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, pages 407–412, 2010.

[30] G. C. Necula. Proof-carrying code. In *POPL '97: Proceedings of the 24th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, pages 106–119, 1997.

[31] R. Rad, J. Plusquellic, and M. Tehranipoor. Sensitivity analysis to hardware Trojans using power supply transient signals. In *HOST*, pages 3–7, 2008.

[32] R. M. Rad, X. Wang, M. Tehranipoor, and J. Plusquellic. Power supply signal calibration techniques for improving detection resolution to hardware Trojans. In *ICCAD*, pages 632–639, 2008.

[33] J. Rolt, A. Das, G. Natale, M.-L. Flottes, B. Rouzeyre, and I. Verbauwhede. A new scan attack on rsa in presence of industrial countermeasures. In W. Schindler and S. Huss, editors, *Constructive Side-Channel Analysis and Secure Design*, volume 7275 of *Lecture Notes in Computer Science*, pages 89–104. Springer Berlin Heidelberg, 2012.

[34] G. Sengar, D. Mukhopadhyay, and D. Chowdhury. Secured flipped scan-chain model for crypto-architecture. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(11):2080–2084, 2007.

[35] C. Sturton, M. Hicks, D. Wagner, and S. King. Defeating UCI: Building stealthy and malicious hardware. In *2011 IEEE Symposium on Security and Privacy (SP)*, pages 64–77, 2011.

[36] F. Wolff, C. Papachristou, S. Bhunia, and R. S. Chakraborty. Towards Trojan-free trusted ICs: Problem analysis and detection scheme. In *DATE*, pages 1362–1365, 2008.

[37] B. Yang, K. Wu, and R. Karri. Scan based side channel attack on dedicated hardware implementations of data encryption standard. In *ITC*, pages 339–344, 2004.

[38] B. Yang, K. Wu, and R. Karri. Secure scan: A design-for-test architecture for crypto chips. *IEEE Transactions on CAD*, 25(10):2287–2293, 2006.

[39] X. Zhang and M. Tehranipoor. Case study: Detecting hardware trojans in third-party digital ip cores. In *HOST*, pages 67–70, 2011.