

# A Hardware Threat Modeling Concept for Trustable Integrated Circuits

Jia Di<sup>1</sup> and Scott Smith<sup>2</sup>

<sup>1</sup>Computer Science and Computer Engineering Department  
University of Arkansas, [jdi@uark.edu](mailto:jdi@uark.edu)

<sup>2</sup>Electrical and Computer Engineering Department  
University of Missouri-Rolla, [smithsco@umr.edu](mailto:smithsco@umr.edu)

## Abstract

Similar to the effects of software viruses, hardware can also be compromised by introduction of malicious logic into circuits to cause unwanted system behaviors. This can be done by changing or adding internal logic, in such a way that it is undetectable using traditional testing and verification tools and techniques. Therefore, the user of the circuit needs to decide whether it can be trusted, i.e., it only performs functions defined in the original circuit specification (no more and no less), before employing it in the system. In this paper, a preliminary methodology is proposed to model potential hardware threats in order to determine a circuit's trustability and provide guidance to malicious-logic checking tools.

## 1. INTRODUCTION

The impact of software viruses has been felt by the entire computerized world, through loss of productivity, loss of system resources or data, or mere inconvenience on a massive scale. Hardware, especially integrated circuits (ICs), was considered safe and attack-free, in contrast to its software counterpart. However, as technologies advance and markets expand, hardware is becoming as vulnerable as software. Malicious logic, similar to the viruses in software, could be inserted into the design like Trojan horses, in that they can lie dormant and undetectable until activated, but then cannot be effectively defeated. Take Electronic Voting Machines (EVMs) as an example: EVMs are claimed to be tamper-proof and error-free because the embedded software is hard-coded in memory, such that it cannot be altered. However, this claim is only true if the underlying hardware is not corrupt. An EVM consists of a microcontroller and other digital hardware components. If any of these components is compromised due to the inclusion of malicious logic, e.g., moving one vote from the "YES" group to the "NO" group if a voting sequence of "YYYYNNN" is observed, then the voting result is no longer trustable. Such erroneous behavior is very difficult to check through design verification due to the infinite number of possible input patterns.

These days most complex digital systems are not designed from scratch; instead they use many 3<sup>rd</sup> party Intellectual Property (IP) blocks. Hence, one or more 3<sup>rd</sup> party IP blocks could contain malicious logic that may affect the entire system. Furthermore, all non-trivial digital

designs rely heavily on Computer-Aided Design (CAD) tools. These CAD tools themselves may be contaminated or malevolently configured, causing them to insert malicious logic into the circuits. Figure 1 shows the supply chain structure of semiconductor ICs similar to that in [1]. Two types of ICs are included: Application Specific ICs (ASICs) and Field Programmable Gate Arrays (FPGAs). It can be seen from Figure 1 that although some steps are trusted due to the fact that they are under the designer's control, the fabrication phase for ASICs, the configuration phase for FPGAs, and the design phase for both, are not trustable. Malicious logic can be inserted in the circuit during any untrustable phase. The malicious logic can lead to various unwanted scenarios like causing the system to output data to the wrong port or address (information leakage), monitoring and modifying the system's output data (tampering), or disabling the system by changing the system's internal timing or control, e.g., holding the clock or bus (denial of service). All these can be done by changing or adding internal logic in such a way that it is very unlikely to be detected by traditional testing and verification tools and techniques [1]. These threats must be identified and the corresponding attacks must be mitigated to ensure the IC is trustable before the applications employing these chips are placed into operation. Programmable logic devices like FPGAs and Complex Programmable Logic Devices (CPLDs) do not have the fabrication phase. After the design phase, the synthesized netlist will be transferred into a configuration file and be downloaded into the device. Malicious logic may be inserted into the configuration file. For run-time reconfigurable FPGAs whose configuration can be changed online, the situation is even worse since the damage can be done in real time.

Threat modeling is a method of assessing and documenting the security risks associated with an application. Threat modeling looks at a system from an adversary's perspective to anticipate attack goals. For hardware, threat modeling consists of analyzing the circuit structure, identifying the potential threats, enumerating the possible attacks, and determining the trustability, which will be introduced in detail in Section 3.

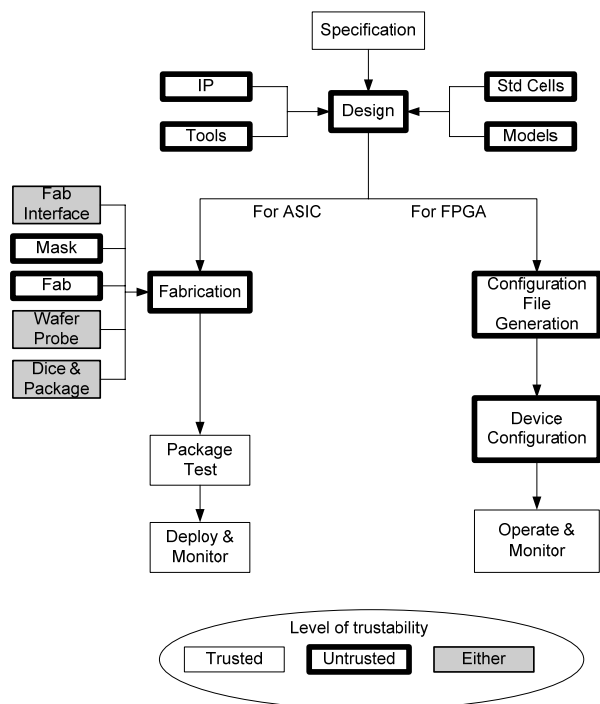


Figure 1: Supply chain structure of semiconductor ICs

## 2. RELATED WORK

Much threat modeling-related research has been performed [2-10]. Attack Graph, which was developed for networking systems, was introduced in [3]. Attack Graph can analyze the risks to a specific network asset, or examine the universe of possible consequences following a successful attack. Another method named Attack Tree was proposed in [4]. Attack trees provide a formal, methodical way of describing the security of systems, based on varying attacks. Both methods are widely used to analyze software and networking system security. A Privilege Graph approach was proposed in [5] to quantitatively assess the operational security of computing systems. A probabilistic logic modeling technique of network reliability for hybrid network architectures was introduced in [6]. A new way of looking at security violations, called insecurity flow, was proposed in [7]. In contrast to those protection-based models, this model is only concerned with whether or not an insecurity can or cannot make it past the protection. Systematic approaches to model threats can be found in [2], [8], and [9]. In [2], threat modeling for software was illustrated in detail. It shares a solid, practical approach to threat modeling and risk mitigation. The contents include all important aspects about software threat modeling, e.g., the rationale behind threat modeling, the modeling process, the implementation of the model, and samples of practical threat models. [8] is more like an abstract of [2] with clear explanation and wider scope of description. A unified conceptual framework for security auditing from a risk management perspective through the generation of threat models was proposed in [9]. This Trike methodology is still under heavy development. Although all these methods are

very useful in their areas of applications, they were developed for either software or networking systems, thus cannot be applied directly to hardware.

Two other models, namely, STRIDE and DREAD, have also been developed and implemented in practice [10]. The STRIDE model is used to categorize the identified threats. STRIDE stands for six effects of realizing a threat: Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege. The DREAD method is used to characterize the risk associated with vulnerability. DREAD stands for five categories of risks: Damage potential, Reproducibility, Exploitability, Affected users, and Discoverability. Both methods are very useful in threat and risk analysis. However, these two methods were also not developed for hardware. Some categories are not applicable to hardware threats, e.g., Spoofing, Repudiation, Elevation of privilege; some can be applied but have different meanings, e.g., Exploitability, Affected users, Discoverability, while some hardware issues are not covered, e.g., online amendability of the threats by the malicious-logic checking tools for run-time reconfigurable FPGAs.

## 3. HARDWARE THREAT MODELING

### 3.1 Establishment of Threat Models for Digital ICs

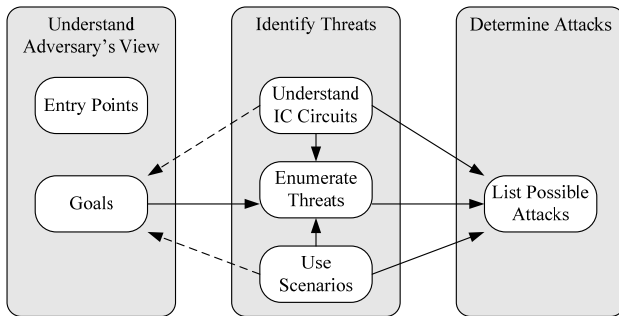
#### 3.1.1 Uniqueness of hardware threats compared to software and network counterparts

Hardware is fundamentally different from software and networking systems in terms of trustability concerns. The uniqueness of hardware includes the following:

- For ASICs, after deployment the design is fixed to both attackers and protectors. This means that there will be no dynamic attack to the hardware. In other words, if the ASIC is guaranteed to be trustable after fabrication, no more malicious logic can be added to it. On the other hand, if the malicious logic has been added to the design before then, it cannot be mitigated either;
- The way malicious logic works is to directly incorporate itself into the original design and perform illegal operations without relying on the data being processed in the circuit. Therefore, cryptography and authentication, which are widely used in software and networking system protection, will not help;
- Hardware attackers do not need to have physical access to the design. The malicious logic can be added through either soft IPs or CAD tools performing synthesis or layout; and
- Since a hardware design is fixed after fabrication/configuration, **all possible threats/attacks can be enumerated in a step-by-step process**. Furthermore, virtually no piece of hardware can bypass the malicious-logic check for the same reason.

### 3.1.2 Modeling Process and Algorithm

The preliminary hardware threat modeling process is comprised of three high-level steps similar to that in [2], as shown in Figure 2: Understanding the Adversary's View, Identifying Threats, and Determining Attacks. The first two steps are further divided into logical substeps. However, threat modeling is not strictly linear [2]. For example, if the same IC is used for different applications, the external scenarios may change. This will cause the Use Scenarios substep to be modified, thus affecting other substeps, including the Goals in the previous step.



**Figure 2: Threat modeling process**

#### *Understand adversary's view*

Basically, this step aims to answer the question “what do the attackers want?” This is critical because the ultimate purpose of threat modeling is to prevent attackers from fulfilling their goals. In order to achieve this, the attackers’ goals must be understood.

*Entry points* – Entry points are the entrances where the attackers can interface with the target. Unlike in software/network systems, where the entry points are usually difficult to enumerate and subject to change, entry points for hardware are much fewer and relatively fixed. For example, entry points for ASICs in the design phase include the IP cores and synthesis and layout CAD tools;

*Goals* – Hardware attackers’ goals are usually tied to the IC’s applications. Most goals can be classified into one or more of three categories:

- Information leakage – attackers extract information directly from an IC, passively or actively, as an individual component, and/or as a deployed element of an integrated system. Information to be protected includes the IP associated with a chipset and its design, data associated with both the hardware and deployed software, and data embedded or downloaded to the IC either prior to or during operation [1];
- Tampering – attackers eavesdrop on or modify the data associated with the IC once it is deployed in operation, independently or as part of an integrated subsystem, by prolonged inspection and monitoring; and
- Denial of service – attackers modify the internal circuit structure of an IC to cause the circuit to

malfunction or shut down under certain operating conditions.

#### *Identify threats*

Based on the attackers’ goals summarized in the previous step, this step aims to identify all possible threats of the IC for the attackers to fulfill their goals. For example, if one of the attackers’ goals is to acquire a certain cryptographic key stored inside the IC, the corresponding threats could include accessing the on-chip RAM, controlling the internal bus, and modifying the output address.

*Understand IC circuit structure* – The goal of this substep is to understand the IC’s functionality and its internal circuit structure, which will help identify the threats. A C program will read the design file (HDL, synthesized netlist, or post-layout netlist) and reconstruct the circuit applying a special data structure. Besides the connection information, this data structure will also contain a field to indicate the vulnerability of each component, i.e., its probability of being compromised. For example, the highly vulnerable components include bus drivers, counters, timers, storage elements, power management unit, etc., while decoders, clock distribution buffers, ALU, etc. are less vulnerable since malicious logic added to these components are likely to be found by verification tools. This vulnerability field will facilitate the threat/attack enumeration in latter steps.

*Use scenarios* – This substep is to determine the potential applications of the target IC. Two questions that need to be investigated are: how the IC will be used and how the IC will *not* be used. Both are very helpful to threat modeling. The former suggests the types of potential threats that need to be identified, e.g., clock tree and power management unit in flight control ICs. The latter indicates what threats do not need to be considered for this IC, e.g., parallel communication port in an IC used for the 2-wire I<sup>2</sup>C bus communication environment. This substep is able to direct the threat/attack enumeration and improve efficiency.

*Enumerate threats* – As mentioned above, it is possible to enumerate all potential threats in an IC because once fabricated/configured, the circuit structure is fixed. In this substep, all threats will be enumerated based on the understanding of attackers’ goals, internal circuit structure, and external scenarios. A tree-like structure will be incorporated to organize the threat model. Each node corresponds to a new type of threats. Leaves are the determined attacks that will be introduced next.

#### *Determine attacks*

*List possible attacks* – This step aims to answer the question “how can the attackers do it?” by enumerating and listing all possible attacks corresponding to each threat identified in the previous step. During the process, both internal circuit structure and external scenarios will be taken into account. For example, if the attacker wants to access the on-chip RAM to get the cryptographic key, the possible attacks include adding additional control logic, most likely

consisting of multiplexers and latches, to the memory address register and chip select signal generator. These attacks will be enumerated associated with the corresponding threats. Since the target IC could be very complex and difficult to analyze, optimization techniques need to be applied to the enumeration process, e.g., group the similar attacks and build/maintain an attack library to speed up the analysis for future ICs.

### 3.2 Application of Hardware Threat Model for Trustability Analysis and Determination

After the hardware threat model has been generated, questions like “is this IC trustable?” or “how secure is this IC?” must be answered. In order to do this, an indicator to characterize the relative trustability of the target IC compared to other ICs needs to be defined. If this indicator is quantifiable, it will provide a direct trustability evaluation of the IC to help define the trustable/untrustable boundary.

#### 3.2.1 Considerations to determine the severity of threats/attacks

A quantified threat model should be able to rank the possible threats/attacks by their severity in order to guide malicious-logic checking tools to work efficiently. Efficiency is especially important for the embedded processors running online malicious-logic checking tools inside run-time reconfigurable FPGAs, since the computational resources are strictly limited. The following considerations need to be addressed while analyzing the threats/attacks:

- How serious the consequences will be if the threat/attack is not detected and mitigated;
- How difficult to detect this threat/attack by regular design verification tools, and online/offline malicious-logic checking tools;
- How difficult to mitigate this threat/attack by online/offline malicious-logic checking tools; and
- How easily this threat/attack can be implemented/inserted into the design, and what constraints, if any, are associated with it.

#### 3.2.2 Analysis mechanism

A detailed mechanism for analyzing the severity of hardware threats/attacks is yet to be defined. The procedure will be to first enumerate all threat effects and classify them into several categories, e.g., Denial of service, Information leakage, etc. Next, according to the functionality of the IC and its application, assign a number to each category to identify the severity, e.g., for central control applications the Denial of service is more critical, while in secure communication applications Information leakage is more important. Then for each threat/attack, weighted numbers will be assigned to all considerations listed in the previous section to represent the possibilities/severities. Finally, these numbers will be used to generate a quantified trustability indicator, calculated from specific mathematical equations. Note that the indicator will not be just a single number; instead, it will include a systematic numerical analysis containing a quantified evaluation of each possible threat/attack. Therefore, this indicator will serve well in

determining the trustability as well as the trustable/untrustable boundary of the IC for its applications.

## 4. CONCLUSION

Preliminary work on hardware threat modeling is presented in this paper. Considering the uniqueness of hardware malicious logic attacks, hardware threat modeling is able to enumerate all potential threats/attacks, rank them in the order of severity, and evaluate the trustability indicator of the target IC, thus providing guidance to malicious-logic checking tools in order to improve their effectiveness and efficiency.

## REFERENCE

- [1] DARPA Trust RFI, Draft Trust Program Information, URL: <http://www.fbo.gov/spg/ODA/DARPA/CMO/SN06%2D25/listing.html>
- [2] F. Swiderski and W. Snyder, Threat Modeling, Microsoft Press, 2004
- [3] C. Phillips and L. P. Swiler, “A graph-based system for network-vulnerability analysis,” Proceedings of the 1998 Workshop on New Security Paradigms, pp. 71-79, 1998
- [4] B. Schneier, “Attack Trees – Modeling Security Threats,” Dr. Dobb’s Journal, Dec. 1999.
- [5] M. Dacier, Y. Deswarte, M. Kaaniche, “Quantitative Assessment of Operational Security: Models and Tools,” Information System Security: facing the information society of the 21st century, pp. 177-186, 1996
- [6] G. D. Wyss, H. K. Schriener, and T. R. Gaylor, “Probabilistic Logic Modeling of Network Reliability for Hybrid Network Architectures,” Proceedings of the 21st IEEE Conference on Local Computer Networks, 1996
- [7] I. S. Moskowithz and M. H. Kang, “An insecurity flow model,” Proceedings of the 6th New Security Paradigms Workshop, Sep. 1997, pp. 61-74
- [8] Peter Torr, “Demystifying the threat-modeling process,” IEEE Security and Privacy, Vol. 3, NO. 5, pp. 66-70, 2005
- [9] P. Saitta, B. Larcom, and M. Eddington, “Trike v.1 Methodology Document [Draft],” URL: [http://dymaxion.org/trike/Trike\\_v1\\_Methodology\\_Document-draft.pdf](http://dymaxion.org/trike/Trike_v1_Methodology_Document-draft.pdf)
- [10] M. Howard and D. LeBlanc, Writing Secure Code, 2nd edition, Microsoft Press, 2002